Electronică digitală
- Curs -

Gheorghe TOACŞE [1]

April 3, 2005

[1]TRANSILVANIA University Braşov, Electronics and Computers
Email: toacseg@vega.unitbv.ro

Cuprins

Capitolul 1

PORŢI LOGICE

1.1 SUPORTUL LOGIC PENTRU SISTEMELE DIGITALE

Practica sistemelor digitale simple se poate face pe bază de raţionamente logice elementare. Dar, deoarece sistemele digitale au devenit foarte complexe, şi această tendinţă continuă, pentru analiza, proiectarea/sinteza şi realizarea acestora este necesar un suport formal abstract şi un suport circuistic. Suportul formal abstract se construieşte, prin extensie, pe baza algebrei logice bivalente iar suportul circuistic se construieşte pornind de la poarta logică. Abordarea întrepătrunsă a acestor două componente constituie subiectul acestui capitol.

1.1.1 Axiomele şi teoremele algebrei Booleene

În **logica aristoteliană**, care trebuie înţeleasă ca o ştiinţă a demonstraţiei, al cărui obiect îl constituie stabilirea condiţiilor corectitudinii gândirii, se operează pe bază de raţionament cu propoziţii (**logica propoziţiilor**), care pot fi: fie adevărate, fie false. De exemplu, dacă presupunem că pentru statura unei persoane sunt admise numai două atribute - înalt şi scund - iar pentru vreme numai două atribute - rece şi cald - atunci sunt naturale următoarele patru propoziţii simple: *vremea este caldă*, *vremea este rece*, *Radu este înalt* şi *Radu este scund*. Considerând că din fiecare pereche de atribute unul este adevărat şi celălalt fals rezultă că propoziţiile formate cu aceste atribute pot fi fie adevărate fie false; o propoziţie nu poate fi simultan şi falsă şi adevărată (în cazul nostru considerăm că prima şi a treia propoziţie simplă sunt adevărate, iar celelalte două sunt false). Dar, cu aceste propoziţii simple pot fi realizate propoziţii compuse, de exemplu: *vremea este caldă şi Radu este înalt*, *vremea este rece sau Radu este scund*. Propoziţiile compuse pot fi, la fel, adevărate sau false in funcţie de valoarea de adevăr/fals a propoziţiilor componente şi de *modul de compunere* a acestora în propoziţia compusă. Modul de compunere, în acest caz corespunde conectorului AND/ŞI pentru prima propoziţie compusă, respectiv conectorului OR/SAU pentru a doua propoziţie compusă. Dacă prima propoziţie are valoarea de adevăr, când vremea este caldă şi Radu este înalt, evident că a doua propoziţie compusă are valoare de fals. Dar dacă prima propoziţie compusă o transformăm în *nu*

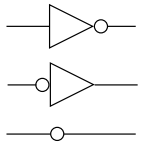
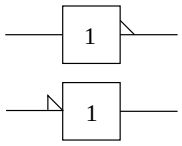
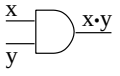
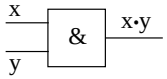
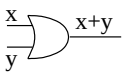
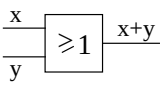
este adevărat: că Radu este înalt şi vremea este caldă atunci această nouă propoziţie are o valoare de fals ca şi a doua propoziţie compusă? Evident că da. Dar dacă mai introducem şi o a treia pereche de propoziţii simple *zăpada este albă*, *zăpada este neagră* şi formăm propoziţii compuse, după diferite moduri de compunere, din câte una din fiecare pereche anterioară atunci mai putem afirma cu uşurinţă care dintre propoziţiile compuse sunt echivalente? Destul de greu. În general, la nivelul normal de dotare intelectuală, un individ cu greu poate realiza un raţionament corect, fără un suport mecanic, când operează simultan cu mai mult de trei variabile. Pentru a învinge această incapacitate avem nevoie de un instrument care să "mecanizeze" raţionamentele stufoase.

Matematicianul englez George Boole (1815-1854) a elaborat o algebră (algebra Booleană) ale cărei axiome şi teoreme pot fi utilizate pentru a transforma (transfera) logica aristoteliană a propoziţiilor din domeniul raţionamentului oral într-un limbaj formal, operant cu simboluri (logica formală). Această logică formală poate fi aplicată ca un instrument operant şi pentru descrierea conectării, în sisteme, a elementelor fizice (mecanice, electrice, hidropneumatice) care prezintă in funcţionarea lor doar două stări distincte.

Algebra Booleană (algebră logică binară, algebră logică bivalentă) operează pe o mulţime binară **B**:

$$\begin{aligned} B &= \{x | x = 0, 1\} \\ 0 &= \text{elementul nul} \\ 1 &= \text{elementul unitate (universal).} \end{aligned} \tag{1.1}$$

Tabelul 1.1 Operatorii booleeni; definiţie şi simboluri grafice

OPERATORUL LOGIC	SIMBOL	TABELUL DE ADEVAR	Reprezentarea grafica conform standardului IEC/ANSI−91−1984 varianta veche	varianta noua
Complementarea logica(negatia) **NOT / NON**	’ —	x | x’ : 0 | 1 ; 1 | 0		
Conjunctia, Produsul logic **AND / SI**	⋀ ⋂ •	x y | x∩y x∧y x·y : 0 0 | 0 ; 0 1 | 0 ; 1 0 | 0 ; 1 1 | 1		
Disjunctia, Suma logica **OR / SAU**	⋁ ⋃ +	x y | x∪y x∨y x+y : 0 0 | 0 ; 0 1 | 1 ; 1 0 | 1 ; 1 1 | 1		

ANSI (American National Standard Institute)

În această carte elementele mulţimii binare reprezintă valorile logice de adevăr sau de fals sau cifrele sistemului de numeraţie în baza doi - **bit** (**bi**nary digi**t** - cifră binară).

Există concepute şi algebre definite pe mulţimi care conţin mai mult de două elemente, **algebre polivalente** – **logică polivalentă**, **logică cu n-valori**; cazuri în care mulţimea de definiţie este formată din n numere întregi. O generalizare pentru logica cu n-valori este **logica fuzzy**, însă în logica fuzzy variabilele iau valori în mod continuu pe intervalul [0,1]. [Matei ′93], [Cocan ′01].

Definiţia 1.1 Fie M o mulţime nevidă. O aplicaţie f definită pe M cu valori în M

$$f : M \to M$$

se numeşte **lege de compoziţie internă**. $\diamond$

Algebra Booleană defineşte pe mulţimea B următoarele trei legi de compoziţie internă (la care, obişnuit, ne vom referi şi prin termenul de operator logic):

1. **Complementarea** sau **negaţia (NOT/NON)**;
2. **Conjuncţia** sau **produsul logic (AND/ŞI)**;
3. **Disjuncţia** sau **suma logică (OR/SAU)**.

Tabelul 1.2 Axiomele şi teoremele algebrei Booleene

Denumirea	Forma cu operatorul produs ($\cdot$)	Forma cu operatorul sumă ($+$)
Axioma 1: Mulţimea $B = \{0,1\}$ este închisă în raport cu operatorii $+$ şi $\cdot$ (Inchiderea)	$x \in B,\ y \in B \Rightarrow x \cdot y \in B$	$x \in B,\ y \in B \Rightarrow x + y \in B$
Axioma 2: Asociativitatea	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$	$x + (y + z) = (x + y) + z$
Axioma 3: Comutativitatea	$x \cdot y = y \cdot x$	$x + y = y + x$
Axioma 4: Existenţa elementului neutru	$x \cdot 1 = 1 \cdot x = x$	$x + 0 = 0 + x = x$
Axioma 5: Distributivitatea	$x \cdot (y + z) = x \cdot y + x \cdot z$	$x + y \cdot z = (x + y) \cdot (x + z)$
Axioma 6: Existenţa complementului	$x \cdot \overline{x} = 0$	$x + \overline{x} = 1$
Teorema 1: Idempotenţa sau tautologia	$x \cdot x = x$	$x + x = x$
Teorema 2: Legea lui 0 şi a lui 1	$x \cdot 0 = 0$	$x + 1 = 1$
Teorema 3: Dubla negaţie (Involuţia)	$\overline{\overline{x}} = x$	$\overline{\overline{x}} = x$
Teorema 4: Absorbţia Absorbţia inversă	$x \cdot (x + y) = x$ $\overline{x} \cdot (x + y) = \overline{x} \cdot y$ $x \cdot (\overline{x} + y) = x \cdot y$	$x + x \cdot y = x$ $\overline{x} + x \cdot y = \overline{x} + y$ $x + \overline{x} \cdot y = x + y$
Teorema 5: Teorema lui De Morgan	$\overline{x \cdot y} = \overline{x} + \overline{y}$	$\overline{x + y} = \overline{x} \cdot \overline{y}$

În *Tabelul 1.1* sunt prezentate pentru fiecare lege de compoziţie booleană aplicaţia definită sub forma unui tabel de adevăr precum şi simbolurile grafice de reprezentare. Deşi există recomandarea pentru utilizarea noilor reprezentări grafice s-a încetăţenit şi continuă folosirea vechilor reprezentări grafice (americane).

Definiţia 1.2 O expresie Booleană $f(x_1, x_2, \ldots, x_n, 0, 1, \cdot, +)$, compusă prin intermediul celor trei operatori AND, OR, NOT prezintă o **expresie dual ă $\mathbf{f^D}$** care se obţine din expresia lui f prin substituţiile următoare: $AND \rightarrow OR$, $OR \rightarrow AND$, $0 \rightarrow 1$, $1 \rightarrow 0$

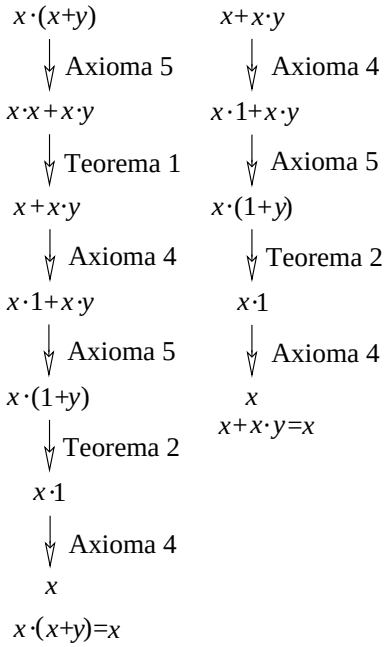
$$f^D(x_1, x_2, \ldots, x_n, 0, 1, \cdot, +) = f(x_1, x_2, \ldots, x_n, 1, 0, +, \cdot) \tag{1.2}$$

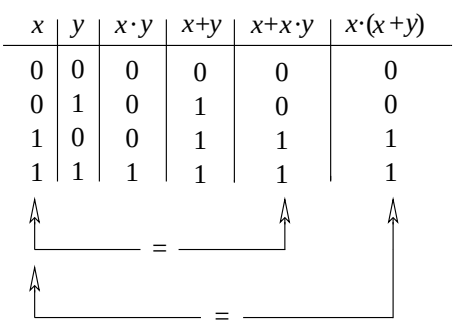
◇

Dacă o axiomă, teoremă sau expresie booleană este adevărată atunci şi forma sa dual ă este adevărată.

Axiomele şi teoremele algebrei Booleene sunt prezentate, sistematic, în *Tabelul 1.2*. Pentru fiecare dintre acestea sunt expuse cele două forme, cea în formă produs şi cea în formă sumă, fiecare fiind duala celeilalte. În acest tabel sunt şase axiome şi cinci teoreme.

Corectitudinea unei expresii booleene se poate verifica analitic (prin utilizarea axiomelor şi teoremelor din *Tabelul 1.2*) sau prin calcularea valorilor logice ale expresiilor din cele două părţi ale semnului egalităţii. Folosind aceste două modalităţi în continuare se va verifica corectitudinea expresiei teoremei absorbţiei (Teorema 4).





x	y	$x \cdot y$	$x+y$	$x+x \cdot y$	$x \cdot (x+y)$
0	0	0	0	0	0
0	1	0	1	0	0
1	0	0	1	1	1
1	1	1	1	1	1

Continuăm demonstraţia analitică pentru variantele teoremei de absorbţie inversă (Teorema 4), dar acum nu se mai indică succesiune numărul axiomelor şi teoremelor aplicate.

$\overline{x}+y=\overline{x}+y\cdot 1=\overline{x}+y(\overline{x}+x)=\overline{x}\cdot 1+y\cdot\overline{x}+y\cdot x=\overline{x}\cdot(1+y)+y\cdot x=\overline{x}+x\cdot y$
$x+y=x+y\cdot 1=x+y(\overline{x}+x)=x\cdot 1+y\cdot\overline{x}+y\cdot x=x\cdot(1+y)+\overline{x}\cdot y=x+\overline{x}\cdot y$

1.1.2 Algebre polivalente

Algebra Booleană a devenit un suport formal pentru sistemele fizice care utilizează elemente cu două stări distincte. Algebra Booleană, care am introdus-o anterior şi la care ne vom referi prin $\mathbf{B}(2)$, este cel mai simplu membru al unei familii de algebre Booleene $\mathbf{B}(q)$ bazate pe noţiunea abstractă de latice (o latice este o mulţime nevidă L înzestrată cu două operaţii "$\vee$", "$\wedge$" care satisfac proprietăţile de idempotenţă, comutativitate, asociativitate şi absorbţie). Astfel, se poate construi o algebră Booleană $B(q)$ pentru orice număr q care este o putere a lui doi, $q=2^k$. Deci există familia de algebre Booleene $B(2)$, $B(4)$, $B(8)$,..., $B(2^k)$. Pentru $k=1$, $q=2^1$ rezultă $B(2)$ definită pe $\{0,1\}$ care este chiar algebra Booleană prezentată anterior. Pentru $k=2$, $q=2^2=4$ rezultă $B(4)$ definită pe mulţimea $\{0,a,b,1\}$ cu următoarele tabele de definiţie ale operatorilor: conjuncţie, disjuncţie şi deplasarea ciclică.

$\cdot$	0	a	b	1
0	0	0	0	0
a	0	a	0	a
b	0	0	b	b
1	0	a	b	1

Conjuncţia

$+$	0	a	b	1
0	0	a	b	1
a	a	a	1	1
b	b	1	b	1
1	1	1	1	1

Disjuncţia

x	x'
0	a
a	b
b	1
1	0

Deplasarea ciclică

Dintre toate algebrele $B(q)$ numai $B(2)$ este **funcţional completă**, deci poate fi suport pentru implementarea sistemelor logice. O algebră este funcţional completă dacă pentru o funcţie de un număr de variabile se generează doar o singură reprezentare/expresie.

Dezvoltarea tehnologiei electronice a dus la realizarea unor elemente care pot realiza mai multe stări distincte. Era normal, ca pentru elementele cu mai multe stări, să se găsească un suport formal cu valori multiple adică algebre polivalente, sau q-valente, mulţimea de definiţie pentru aceste algebre fiind formată dintr-un număr de q elemente distincte. Construcţia algebrelor polivalente a urmat două căi. Prima, a fost o **generalizare a operatorilor/(conectivi) booleeni** AND, OR şi NOT spre operatori corespunzători – **conjuncţia**, **disjuncţia** şi **deplasarea ciclică** – obţinându-se algebrele Postiene (introduse de E.L. Post, 1921). A doua cale, dezvoltată de B.A. Bernstein (1928), a fost o **abordare prin algebra claselor de resturi**, operatorii fiind adunarea şi înmulţirea modulo q.

O **algebră Postiană q-valentă**, $P(q)$, este definită pe mulţimea $\{0,1,2,\ldots,q-1\}$, adică pe intervalul de numere întregi $[0,q-1]$, iar operatorii sunt definiţi în felul următor:

- conjuncţia: $x\cdot y=\min(x,y)$;
- disjuncţia: $x+y=\max(x,y)$;
- deplasarea ciclică: $x'=x\oplus 1$ modulo q.

Pentru algebrele Postiene $P(2)$ şi $P(4)$ rezultă următoarele tabele de adevăr ale celor trei operatori:

- $\mathbf{P}(2)$

$\cdot$	0	1
0	0	0
1	0	1

Conjuncţia binară

$+$	0	1
0	0	1
1	1	1

Disjuncţia binară

x	x'	x''
0	1	0
1	0	1

Deplasarea ciclică binară

- $\mathbf{P}(4)$

$\cdot$	0	1	2	3
0	0	0	0	0
1	0	1	1	1
2	0	1	2	2
3	0	1	2	3

Conjuncţia cuaternară

$+$	0	1	2	3
0	0	1	2	3
1	1	1	2	3
2	2	2	2	3
3	3	3	3	3

Disjuncţia cuaternară

x	x'	x''	x'''	x''''
0	1	2	3	0
1	2	3	0	1
2	3	0	1	2
3	0	1	2	3

Deplasarea ciclică cuaternară

Rezultă o identitate între $B(2)$ şi $P(2)$, ($\boldsymbol{B(2) \equiv P(2)}$). Algebrele Postiene pot fi o replică pentru algebrele Booleene, dar operarea în aceste algebre, ca aplicaţii pentru funcţiile de comutaţie, devine dificilă pe măsură ce q are valori mai mari.

Cea de a doua cale de generalizare, dezvoltată de B.A. Bernstein, a generat o altă clasă de algebre q-valente care pot fi definite pentru oricare număr întreg prin q sau pentru un număr întreg putere a lui q. Mulţimea de definiţie pentru o astfel de algebră este $\{0, 1, 2, \ldots, q-1\}$, iar operatorii sunt operaţiile aritmetice de adunare şi de înmulţire modulo q. **Structurile algebrice definite pe clasele de resturi modulo q (q număr prim) sunt referite prin câmpuri Galois şi sunt notate cu $GF(q)$** . Pentru $\boldsymbol{GF(3)}$ tabelele de adevăr prin aplicarea operatorilor produs, $\odot$, şi sumă, $\oplus$, modulo 3 sunt:

$\odot$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

$\oplus$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Dintre structurile $GF(q)$ cea pentru $q = 2$, $\boldsymbol{GF(2)}$ algebra modulo 2, denumită **algebră Reed-Muller**, prezintă interes ca suport formal în implementările unor algoritmi sau circuite de calcul sau de codificare. Operatorii algebrei Reed-Muller au următoarele tabele de adevăr:

$\odot$	0	1
0	0	0
1	0	1

$\oplus$	0	1
0	0	1
1	1	0

x	x'
0	1
1	0

Se observă că atât pentru algebra Booleană $B(2)$ cât şi pentru algebra Reed-Muller, $GF(2)$, operatorii de înmulţire logică sunt identici ($\odot = \cdot$), la fel şi operatorii de complementare. În schimb, diferă operatorii disjuncţie, aceştia fiind SAU INCLUSIV (sumă logică, $+$) în $B(2)$ şi respectiv SAU EXCLUSIV (sumă aritmetică modulo

2, $\oplus$) în $GF(2)$. Această, aparent neînsemnată diferenţă, determină semnificative diferenţe între cele două formalisme, ceea ce apare şi în metodele de proiectare şi implementare. Axiomele pe $GF(2)$ sunt:

1. Închiderea. $GF(2)$ este închisă în raport cu operatorii $\odot$ şi $\oplus$.	$A \in GF(2)$, $B \in GF(2) \Rightarrow A \odot B \in GF(2)$, $A \oplus B \in GF(2)$
2. Asociativitatea.	$A \oplus (B \oplus C) = (A \oplus B) \oplus C = A \oplus B \oplus C$ $A \odot (B \odot C) = (A \odot B) \odot C = A \odot B \odot C$
3. Comutativitatea.	$A \oplus B = B \oplus A$, $A \odot B = B \odot A$
4. Distributivitatea.	$A \odot (B \oplus C) = A \odot B \oplus A \odot C$
5. Elementul neutru.	$A \oplus 0 = A$, $A \odot 1 = A$

Din prezentarea acestor proprietăţi apare similaritatea dintre $GF(2)$ cu algebra numerelor reale (care este definită pe un câmp infinit). Dar următoarea axiomă relevă o proprietate diferită între aceste două algebre care rezultă din natura aritmeticii modulo 2.

6. Existenţa inversului. $A \oplus A = 0, A \odot A = A$

Din ultima axiomă a algebrei $GF(2)$ rezultă că $A = -A$, adică fiecare element este egal cu inversul său; aceasta înseamnă că **adunarea şi scăderea sunt identice în $GF(2)$**, ceea ce este diferit faţă de adunarea aritmetică din algebra numerelor reale. Folosind această axiomă se deduce: dacă $A \oplus B = C$ atunci $A = C \oplus B$, $B = A \oplus C$ şi $A \oplus B \oplus C = 0$.

Se pot duce relaţii pentru exprimarea operatorilor din $B(2)$ prin cei din $GF(2)$

$$\begin{aligned} A \cdot B &= A \odot B \\ A + B &= A \odot B \oplus A \oplus B \\ \overline{A} &= A \oplus 1 \end{aligned} \tag{1.3-a}$$

care se pot demonstra în felul următor. Se consideră expresia pentru sumă modulo doi $A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B$ (a se vedea funcţia $f_6(x_1, x_0)$ în 1.1.3)

$$\begin{aligned} A \oplus 1 &= A \cdot \overline{1} + \overline{A} \cdot 1 = A \cdot 0 + \overline{A} = \overline{A} \\ A + B &= \overline{\overline{A + B}} = \overline{\overline{A} \cdot \overline{B}} = ((A \oplus 1) \odot (B \oplus 1)) \oplus 1 \\ &= (A \odot B \oplus 1 \odot B \oplus A \odot 1 \oplus 1 \odot 1) \oplus 1 = (A \odot B \oplus B \oplus A \oplus 1) \oplus 1 \\ &= A \odot B \oplus B \oplus A \oplus 1 \oplus 1 = A \odot B \oplus B \oplus A \end{aligned}$$

Iar pentru exprimarea operatorilor din $GF(2)$ prin cei din $B(2)$ există relaţiile

$$\begin{aligned} A \odot B &= A \cdot B \\ A \oplus B &= A \cdot \overline{B} + \overline{A} \cdot B \\ A \oplus 1 &= \overline{A} \end{aligned} \tag{1.3-b}$$

Relaţiile 1.3 indică modalităţi de utilizare atât a formalismului din $GF(2)$ cât şi a celui din $B(2)$ pentru implementarea sistemelor în funcţie de suportul fizic (circuistica) disponibilă.

1.1.3 Funcţii Booleene

Fie $B(2) = (\{0,1\}, \cdot, +, -)$ algebra Booleană binară. Un cuvânt binar este o succesiune de biţi; un cuvânt este caracterizat prin lungimea sa, adică de numărul de biţi din succesiune.

Definiţia 1.3 Vom numi o **configuraţie binară** de n biţi, sau cuvânt binar cu lungimea de n biţi, un element al mulţimii $\{0,1\}^n$. $\diamond$

Cu doi biţi se pot forma patru cuvinte distincte cu lungimea de doi biţi (00, 01, 10, 11), cu trei biţi se pot forma opt cuvinte distincte cu lungimea de trei biţi (000, 001, 010, 011, 100, 101, 110, 111), iar cu n biţi se pot forma 2^n cuvinte distincte fiecare cu lungimea de n biţi; deci mulţimea $\{0,1\}^n$ este constituită din 2^n cuvinte distincte cu lungimea de n biţi.

Definiţia 1.4 O funcţie Booleană [1], f , cu n intrări şi m ieşiri este o aplicaţie

$$f : \{0,1\}^n \rightarrow \{0,1\}^m \tag{1.4}$$

cu domeniul de definiţie în $X = \{0,1\}^n$ şi cu domeniul de valori în $Y \subseteq \{0,1\}^m$. $\diamond$

Relaţia 1.4, în cazul când funcţia logică are o singură ieşire, $m = 1$, cuvântul de ieşire are lungimea de un bit, se retranscrie sub forma:

$$f : \{0,1\}^n \rightarrow \{0,1\}^1 \tag{1.5}$$

Teoretic, funcţia logică cu m ieşiri se poate construi din m aplicaţii de forma 1.5 conectate în paralel. O funcţie logică cu numărul de ordine i dintr-o familie de funcţii logice de n variabile, definită conform relaţiei 1.4, va fi notată sub forma $f_i(x_{n-1}, x_{n-2}, \ldots, x_1,\ x_0)$ sau sub forma f_i^n. În acest capitol se vor studia doar funcţiile cu o singură ieşire.

Funcţia logică de zero variabile. Domeniul de definiţie pentru funcţia de zero variabile este mulţimea vidă, $\{0,1\}^0$, iar domeniul de valori este $\{0,1\}$. Rezultă că pot exista două funcţii notate cu f_0^0 şi f_1^0 care generează pe ieşire cele două valori din mulţimea $\{0,1\}$

$$\begin{aligned} f_0^0 &= 0 \\ f_1^0 &= 1 \end{aligned} \tag{1.6}$$

De fapt, putem spune că aceste funcţii sunt identice cu două constante. Într-un sistem digital cele două constante pot fi reprezentate fizic prin două tensiuni fixe: tensiunea de alimentare (V_{DD}, V_{CC}) şi tensiunea de masă V_{SS} sau 0V (liniile/barele de alimentare ale circuitului).

Funcţii logice de o singură variabilă. Configuraţiile distincte de un singur bit pe mulţimea de definiţie $\{0,1\}^1$ sunt cei doi biţi 1 şi 0. Deci funcţia $y = f(x)$, pentru fiecare valoare binară atribuită variabilei x, poate lua una din cele două valori binare $y = 1$ sau $y = 0$. Cu cele două valori posibile pentru y se pot forma patru cuvinte diferite, deci pentru o singură variabilă există patru funcţii logice distincte: f_0^1, f_1^1, f_2^1 şi f_3^1 reprezentate în tabelul din Figura 1.1.

[1] Termenul de funcţie Booleană, în această carte, este sinonim cu funcţie logică.

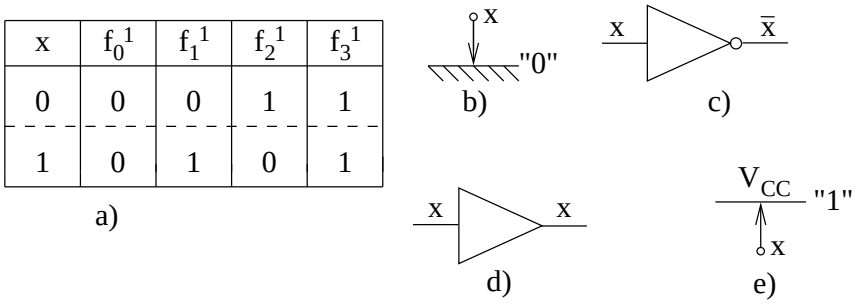


Figura 1.1 Funcţiile de o singură variabilă: a) tabelul funcţiilor de o variabilă; b) f_0^1, funcţia zero (conectarea la masă); c) f_2^1, funcţia inversor (circuitul inversor); d) f_1^1, funcţia identitate (driver, buffer); e) f_3^1, funcţia tautologie (conectarea la tensiunea de alimentare).

1. Funcţia zero $f_0(x) = 0$. Aceasta generează valoarea 0 indiferent de valoarea alocată variabilei x. Într-un sistem nu se va calcula niciodată funcţia zero deoarece valoarea acestei funcţii există, fizic punctul respectiv se leagă la tensiunea de masă; evident f_0^1 şi f_0^0 au acelaşi efect adică valoarea constantă 0.

2. Funcţia identitate, $f_1(x) = x$. Logic, această funcţie pare a fi fără utilitate; dar, practic, această funcţie este foarte utilizată sub denumirea de **driver** sau **buffer** şi într-un sistem fizic are o acţiune de a aduce/întări la anumite valori normale semnalul electric care este suport pentru variabila x. Aceste circuite care nu realizează o funcţie logică ci doar au rol de "întărire" a semnalului electric sunt referite prin circuit buffer sau circuit driver respectiv buffer sau driver.

3. Funcţia negaţie (NOT), $f_2(x) = \overline{x}$. De fapt, acesta este operatorul de complementare din *Tabelul 1.1.* Putem interpreta aspectul logic sau aritmetic al acţiunii acestei funcţii. Logic, funcţia negaţie aplicată va substitui adevărul cu fals şi falsul cu adevăr. Aritmetic, este un incrementor sau decrementor pentru numărarea în baza doi (atât la numărare în sens crescător (direct) cât şi în sens descrescător (invers), trecerea între două numere consecutive se face prin modificarea bitului cel mai puţin semnificativ, **LSB** (**L**east **S**ignificant **B**it), din unu în zero sau din zero în unu, vezi Figura 2.17-a şi 2.17-b). Suportul fizic pentru implementarea acestei funcţii este elementul inversor.

4. Funcţia tautologie, $f_3(x) = 1$. Acestă funcţie generează valoarea 1 indiferent de valoarea alocată variabilei x. Într-un sistem nu se va calcula niciodată această funcţie deoarece valoarea sa există, fizic punctul respectiv se leagă la tensiunea de alimentare V_{DD}/V_{CC}; evident f_3^1 şi f_1^0 au acelaşi efect.

Funcţii de două variabile. La o funcţie de două variabile $f(x_1, x_0)$ mulţimea de definiţie, $\{0,1\}^2$, este compusă din cele patru cuvinte de intrare de doi biţi. Pentru cele patru cuvinte de intrare se obţin pentru funcţie patru valori binare, dar cu cele patru valori binare se pot obţine $4^2 = 16$ cuvinte, deci, în total există 16 funcţii diferite de două variabile care sunt prezentate în tabelul din Figura 1.2-a.

Aceste funcţii au anumite denumiri care exprimă acţiunea realizată:

1. $f_0(x_1, x_0) = 0$, funcţia zero. Acţiunea sa este identică cu a celor două funcţii f_0^0,f_0^1.

x_1	x_2	f_0^2	f_1^2	f_2^2	f_3^2	f_4^2	f_5^2	f_6^2	f_7^2	f_8^2	f_9^2	f_{10}^2	f_{11}^2	f_{12}^2	f_{13}^2	f_{14}^2	f_{15}^2
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

a)

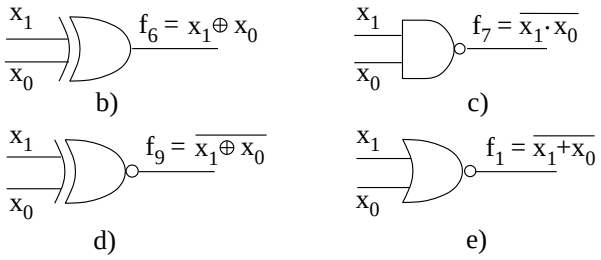


Figura 1.2 Funcţii de două variabile: a) tabelul funcţiilor de două variabile; b) simbolul grafic pentru funcţia XOR, f_6^2; c) simbolul grafic pentru funcţia NAND, f_7^2; d) simbolul grafic pentru funcţia NXOR, f_9^2; e) simbolul grafic pentru funcţia NOR, f_1^2.

2. $f_1(x_1, x_0) = \overline{x_1 + x_0}$, funcţia SAU-NEGAT/NOT-OR/NOR cu reprezentarea grafică din Figura 1.2-e. Se observă că valorile funcţiei rezultă prin negarea valorilor obţinute cu operatorul OR (*Tabelul 1.1*).

3. $f_2(x_1, x_0) = \overline{x_1} \cdot x_0$, funcţia negarea implicaţiei inverse; circuitul interdicţie (inhibare).

4. $f_3(x_1, x_0) = \overline{x_1}$, negarea lui x_1.

5. $f_4(x_1, x_0) = x_1 \cdot \overline{x_0}$, funcţia negarea implicaţiei directe; circuitul interdicţie (inhibare).

6. $f_5(x_1, x_0) = \overline{x_0}$, negarea lui x_0.

7. $f_6(x_1, x_0) = x_1 \cdot \overline{x_0} + \overline{x_1} \cdot x_0$, funcţia negarea coincidenţei, SAU-EXCLUSIV/ XOR cu simbolul grafic de reprezentare în Figura 1.2-b şi cu structura de implementare în Figura 1.3-a. Acţiunea acestei funcţii poate fi interpretată în trei modalităţi.

Prima, se observă că asupra celor două valori binare ale variabilelor x_1 şi x_0 funcţia operează ca un **sumator modulo 2** ($0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$) de unde şi notaţia consacrată $f_6(x_1, x_0) = x_1 \oplus x_0$. O altă variantă echivalentă de exprimare se obţine în felul următor: $x_1 \oplus x_0 = x_1 \cdot \overline{x}_0 + \overline{x}_1 \cdot x_0 = x_1 \cdot \overline{x}_0 + x_0 \cdot \overline{x}_0 + \overline{x}_1 \cdot x_0 + x_1 \cdot \overline{x}_1 = \overline{x}_0 \cdot (x_1 + x_0) + \overline{x}_1(x_1 + x_0) = (x_1 + x_0)(\overline{x}_1 + \overline{x}_0) = (x_1 + x_0) \cdot (\overline{x_1 \cdot x_0})$.

A doua interpretare este cea de **inversor comandat**, relaţiile 1.3. Dacă una din variabile este 1 atunci valoarea funcţiei va fi egală cu negata celeilalte variabile de intrare; $f_6(x_1, 1) = x_1 \oplus 1 = \overline{x}_1$.

A treia interpretare este cea de **negarea coincidenţei**, adică **anticoincidenţă**

(şi invers, negarea anticoincidenţei este coincidenţa, adică $\overline{x_1 \oplus x_0}$). Rezultă că se poate realiza foarte uşor un circuit pentru coincidenţa a două cuvinte de doi biţi x_1x_0, y_1y_0 raţionând în felul următor: cuvintele coincid când nu este adevărată anticoincidenţa pentru biţii de rangul unu x_1, y_1 SI nu este adevărată anticoincidenţa pentru biţii de rangul zero x_0, y_0; deci relaţia de coincidenţă este $\overline{(x_1 \oplus y_1)} \cdot \overline{(x_0 \oplus y_0)} = \overline{(x_1 \oplus y_1) + (x_0 \oplus y_0)}$. Reprezentarea acestui mod de implementare a relaţiei de coincidenţă este dată în Figura 1.3-c.

8. $f_7(x_1, x_0) = \overline{x_1 \cdot x_0}$, funcţia SI-NEGAT/NOT-AND/NAND, cu simbolul grafic de reprezentare din Figura 1.2-c. Se observă că valorile funcţiei rezultă prin negarea valorilor obţinute cu operatorul AND (*Tabelul 1.1*).

9. $f_8(x_1, x_0) = x_1 \cdot x_0$, funcţia conjuncţie, produsul logic AND, SI.

10. $f_9(x_1, x_0) = \overline{x_1} \cdot \overline{x_0} + x_1 \cdot x_0$, $f_9(x_1, x_0) = \overline{x_1 \oplus x_0}$, funcţia coincidenţă, SAU-EXCLUSIV NEGAT/NXOR cu simbolul grafic de reprezentare în Figura 1.2-d şi cu structurarea ca în Figura 1.3-b. Implementarea circuitului de coincidenţă a două cuvinte de doi biţi x_1x_0 şi y_1y_0 este prezentată în Figura 1.3-d; $\overline{(x_1 \oplus y_1)} \cdot \overline{(x_0 \oplus y_0)}$.

11. $f_{10}(x_1, x_0) = x_0$, funcţia ce nu depinde de x_1.

12. $f_{11}(x_1, x_0) = x_1 + \overline{x_0}$, funcţia implicaţie directă.

13. $f_{12}(x_1, x_0) = x_1$, funcţia ce nu depinde de x_0.

14. $f_{13}(x_1, x_0) = \overline{x_1} + x_0$, funcţia implicaţie inversă.

15. $f_{14}(x_1, x_0) = x_1 + x_0$, funcţia conjuncţie, sumă logică OR, SAU.

16. $f_{15}(x_1, x_0) = 1$, funcţia tautologie, acţiunea sa este identică cu ale funcţiilor f_1^0, f_3^1.

Funcţii de trei variabile. Pentru funcţiile de trei variabile $f(x_2, x_1, x_0)$ mulţimea de definţie, $\{0,1\}^3$, este compusă din opt ($2^3 = 8$) cuvinte binare de 3 biţi, pentru fiecare din aceste cuvinte funcţia poate avea o valoare binară 0 sau 1. Cu opt valori binare pot fi definite 2^8 funcţii diferite. Modul de formare al funcţiilor de trei variabile $f_i^3, i = 0, 1, \ldots, 255$ rezultă din *Tabelul 1.3*. Indicele i, care identifică

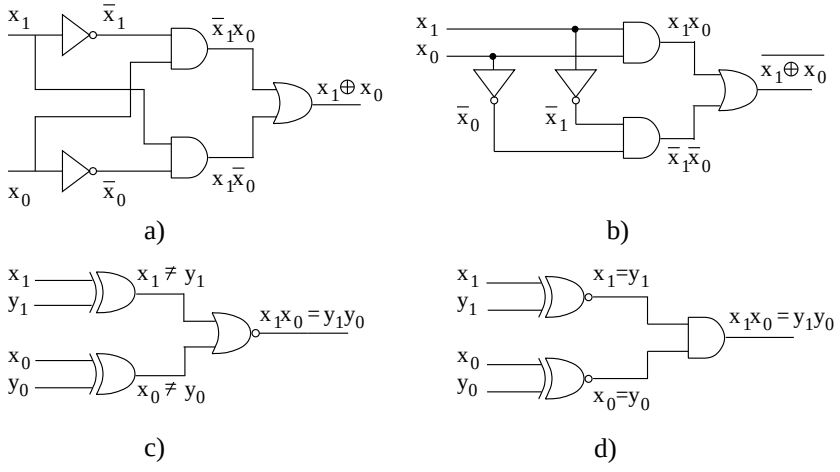


Figura 1.3 Funcţiile XOR şi NXOR: a,b) structura circuitelor XOR, respectiv NXOR; c,d) circuite de coincidenţă cu XOR şi NXOR.

funcţia f_i^3, este corespondentul zecimal al numărului binar de 8 biţi format cu valorile binare ale funcţiei. De exemplu funcţia f_{187} are următoarele valori binare 1101 1101, deoarece $187|_{10} = 1101\,1101|_2$. Bitul cel mai semnificativ, **MSB** (**M**ost **S**ignificant **B**it), din cuvântul valorilor funcţiei, corespunde configuraţiei de intrare $x_2x_1x_0 = 111$, iar **LSB** corespunde configuraţiei de intrare $x_2x_1x_0 = 000$.

Utilitatea tuturor funcţiilor f_i^3 pentru implementarea sistemelor logice este discutabilă deoarece , 16 dintre aceste funcţii sunt echivalentul funcţiilor f_i^2 iar unele din cele rămase pot fi compuse prin intermediul altor funcţii de două variabile. În general, pentru implementarea sistemelor logice sunt utilizate doar câteva din funcţiile expuse pentru cazul $n = 2$.

Pentru n variabile numărul funcţiilor logice distincte este $2^{(2^n)}$; ceea ce determină pentru $n = 4 \rightarrow 2^{16} = 65536$ funcţii, iar pentru $n = 5 \rightarrow 2^{32} = 4294967296$ funcţii(!).

Definiţia 1.5 Un **sistem complet de funcţii Booleene** este un set minimal de funcţii Booleene cu ajutorul cărora se poate exprima orice funcţie Booleană. ⋄

În paragraful 1.1.1 s-au definit cele trei legi de compoziţie pe mulţimea $\boldsymbol{B}$. Cu ajutorul celor trei operatori NOT, AND şi OR poate fi exprimată oricare funcţie logică, deci aceşti operatori formează un sistem complet.

Al doilea set de operatori care pot constitui un sistem complet este perechea NOT şi AND. Acest set poate fi substituit numai cu o singură funcţie, funcţia $f_7(x_1, x_0) = \overline{(x_1 \cdot x_0)}$, adică operatorul NAND, deoarece operatorul NOT apare ca un NAND de aceeaşi variabilă $\overline{(x \cdot x)} = \overline{x}$, iar operatorul AND rezultă ca un NAND negat, $\overline{\overline{(x_1 \cdot x_0)}} = x_1 \cdot x_0$.

Al treilea set de operatori OR şi NOT formează de asemenea un sistem complet. Şi această pereche de operatori poate fi substituită numai cu o singură funcţie, funcţia $f_2(x_1, x_0) = \overline{x_1 + x_0}$, care este operatorul NOR. Deoarece NOR este un OR negat se poate obţine uşor OR prin negarea NOR-ului, $\overline{\overline{(x_1 + x_0)}} = x_1 + x_0$, iar NOT-ul este

Tabelul 1.3 Funcţii logice de trei variabile

x_2	x_1	x_0	f_0^3	f_1^3	f_2^3		f_{17}^3		f_{187}^3		f_{255}^3
0	0	0	0	1	0		1		1		1
0	0	1	0	0	1		0		0		1
0	1	0	0	0	0		0		1		1
0	1	1	0	0	0		0		1		1
1	0	0	0	0	0		1		1		1
1	0	1	0	0	0		0		0		1
1	1	0	0	0	0		0		1		1
1	1	1	0	0	0		0		1		1
							$17\|_{10} = 0001001\|_2$		$187\|_{10} = 10111011\|_2$		

identic cu NOR-ul aplicat aceleiaşi variabile $\overline{(x + x)} = \overline{x}$.

În *Tabelul 1.4* sunt modelaţi operatorii NOT, AND, OR, NAND şi NOR fie numai cu NAND, fie numai cu NOR. Apare, deci, posibilitatea ca un sistem logic să poată fi realizat doar cu un singur operator. Consecinţa practică a acestei observaţii este enormă prin faptul că implementarea unui sistem se poate obţine prin replicarea aceluiaşi operator (acelaşi tip de circuit), în consecinţă preţul de cost scade foarte mult şi siguranţa în funcţionare creşte.

Există circuite integrate, produse de turnătoria de siliciu, dar încă neterminate, care conţin un singur (sau două) circuite logice elementare într-un număr foarte mare, referite ca arie de porţi logice (gate-array, vezi secţiunea 4.3). Pe o astfel de arie de porţi logice utilizatorul îşi poate realiza sistemul pentru aplicaţia sa prin proiectarea doar a conexiunilor între un număr de circuite logice elementare. Apoi, turnătoria de siliciu termină circuitul integrat, prin realizarea conexiunilor proiectate de utilizator, rezultând astfel un circuit cu multe avantaje, la a cărui construcţie a participat atât utilizatorul (custom) cât şi turnătoria de siliciu; sistem referit ca fiind o proiectare de tip semi-custom.

1.1.4 Forme canonice

Definiţia 1.4 exprimă noţiunea de funcţie logică de n variabile. Cu cele n variabile se pot compune, prin intermediul operatorilor AND, OR şi NOT, termenii funcţiei şi la fel tot cu aceşti operatori pot fi incluşi termenii în cadrul funcţiei. În consecinţă, un termen al unei funcţii logice poate avea variabile negate sau nenegate (NOT), care pot fi însumate logic (OR) sau înmulţite logic (AND).

Definiţia 1.6 Termenul canonic produs este produsul logic a tuturor celor n variabile ale funcţiei, negate sau nenegate. Termenul canonic produs este referit ca **minterm**. ◇

Exemplu de termen canonic produs pentru $n = 3$ poate fi $x_2 \cdot \overline{x}_1 \cdot x_0$. Un termen canonic produs nu poate fi format din mai mult de n factori (variabile). Dacă ar avea mai mult de n factori atunci ar însemna că una sau mai multe variabile ar intra în produsul logic atât negate cât şi nenegate ceea ce, conform axiomei de existenţă a complementului $x \cdot \overline{x} = 0$, *Tabelul 1.2*, ar însemna că termenul se reduce la constanta

Tabelul 1.4 Modelarea operatorilor logici pe bază de NAND sau NOR

Operatorul logic / modelat cu poarta:	NOT	AND	OR	NAND	NOR
NAND	$\overline{A} = \overline{A \cdot A}$	$A \cdot B = \overline{\overline{A \cdot B}}$	$A+B = \overline{\overline{A} \cdot \overline{B}}$	$\overline{A+B} = \overline{A+B}$	$\overline{A+B} = \overline{\overline{A \cdot B}}$
NOR	$\overline{A} = \overline{A+A}$	$A \cdot B = \overline{\overline{A}+\overline{B}}$	$A+B = \overline{\overline{A+B}}$	$\overline{A \cdot B} = \overline{\overline{A}+\overline{B}}$	$\overline{A+B} = \overline{A+B}$

Tabelul 1.5 Codificarea termenilor canonici produs şi sumă (n=3)

i	Valoarea variabilelor			Mintermul si codul P_i	Maxtermul si codul S_i
	x_2	x_1	x_0		
0	0	0	0	$\overline{x}_2\overline{x}_1\overline{x}_0 = P_0$	$x_2 + x_1 + x_0 = S_0$
1	0	0	1	$\overline{x}_2\overline{x}_1 x_0 = P_1$	$x_2 + x_1 + \overline{x}_0 = S_1$
2	0	1	0	$\overline{x}_2 x_1\overline{x}_0 = P_2$	$x_2 + \overline{x}_1 + x_0 = S_2$
3	0	1	1	$\overline{x}_2 x_1 x_0 = P_3$	$x_2 + \overline{x}_1 + \overline{x}_0 = S_3$
4	1	0	0	$x_2\overline{x}_1\overline{x}_0 = P_4$	$\overline{x}_2 + x_1 + x_0 = S_4$
5	1	0	1	$x_2\overline{x}_1 x_0 = P_5$	$\overline{x}_2 + x_1 + \overline{x}_0 = S_5$
6	1	1	0	$x_2 x_1\overline{x}_0 = P_6$	$\overline{x}_2 + \overline{x}_1 + x_0 = S_6$
7	1	1	1	$x_2 x_1 x_0 = P_7$	$\overline{x}_2 + \overline{x}_1 + \overline{x}_0 = S_7$

0. Un termen produs se notează prin P_i. Numărul tuturor termenilor produs P_i este 2^n, deci $i = 0, 1, 2, \ldots, 2^n - 1$.

Dar cum se codifică un minterm prin simbolul P_i ? Se va exemplifica pentru cazul $n = 3$. Un termen canonic produs are valoarea logică 1 numai atunci când toţi factorii săi au valoarea logică 1. Pentru exemplul anterior de termen produs $x_2 \cdot \overline{x}_1 \cdot x_0$, cu valoarea logică 1, rezultă o unică configuraţie de valori: $x_2 = 1$, $x_1 = 0$, $x_0 = 1$. Cuvântul binar format din valorile variabilelor este 101, care este numărul cinci în zecimal, reprezentat în codul de numeraţie binar natural. Deci, iată că mintermul $x_2 \cdot \overline{x}_1 \cdot x_0$ poate fi codificat prin litera P cu indicele 5, adică P_5; dar această codificare trebuie să fie o aplicaţie bijectivă deci şi trecerea inversă de la codul P_i la exprimarea ca produs logic canonic a mintermului trebuie să fie unică. De exemplu, trecerea de la termenul canonic produs P_6 la mintermul corespunzător se face în felul următor: $6|_{10} = 110|_2 \rightarrow x_2 \cdot x_1 \cdot \overline{x}_0$.

Rezultă următoarea regulă de codificare a termenilor canonici produs: pentru reprezentarea unui minterm prin simbolul P_i variabilelor negate li se atribuie valoarea zero, iar variabilelor nenegate li se atribuie valoarea unu. Este corectă şi reciproca, în cuvântul de cod P_i pentru un bit cu valoarea unu corespunde în produsul logic canonic o variabilă nenegată iar pentru un bit cu valoarea zero corespunde o variabilă negată. Aplicând această regulă pentru funcţia de trei variabile se pot scrie relaţiile din *Tabelul 1.5*.

Definiţia 1.7 **Termenul canonic sumă** este suma logică a tuturor celor n variabile ale funcţiei, negate sau nenegate. Termenul canonic sumă este referit ca **maxterm**. $\diamond$

Pentru o funcţie de trei variabile ($n = 3$) ca un exemplu de termen canonic sumă poate fi acesta $\overline{x}_2 + x_1 + \overline{x}_0$. La fel, ca şi la termenul canonic produs, un termen canonic sumă nu poate fi compus din mai mult de n variabile, în caz contrar în termen ar exista suma $x + \overline{x} = 1$, deci valoarea termenului ar fi egală cu constanta 1. Numărul total de termeni canonici sumă este 2^n iar codificarea se face prin simbolul S_i.

Termenul canonic sumă $\overline{x}_2 + x_1 + \overline{x}_0$ are valoarea logică zero numai atunci când

fiecare termen al sumei are valoarea zero, adică $x_2 = 1$, $x_1 = 0$, $x_0 = 1$. Rezultă indicele i al simbolului S_i ca fiind egal cu numărul zecimal ce este reprezentat în binar de cuvântul format din valorile celor trei variabile adică $101|_2 = 5|_{10}$, deci S_5. Trecerea inversă, de exemplu de la S_6 la maxtermul corespunzător, se face în felul următor $6|_{10} = 110|_2 \rightarrow \overline{x}_2 + \overline{x}_1 + x_0$.

Rezultă următoarea regulă de codificare a termenilor canonici sumă pentru reprezentarea unui maxterm prin simbolul S_i: variabilelor negate li se atribuie valoarea unu, iar variabilelor nenegate li se atribuie valoarea zero. Este corectă şi reciproca, în cuvântul de cod pentru un bit cu valoarea unu corespunde în sumă logică canonică o variabilă negată iar pentru valoarea zero corespunde o variabilă nenegată. Conform acestei reguli de codificare pentru $n = 3$ se pot scrie relaţiile din *Tabelul 1.5.*

Se observă că, pentru codificare, la aceeaşi variabilă nenegată se atribuie valoarea 1 în minterm şi 0 în maxterm şi pentru aceeaşi variabilă negată se atribuie valoarea 0 în minterm şi 1 în maxterm. Iar pentru trecerea inversă, de la cod la expresia logică, unui bit 1 în cuvântul de cod îi corespunde o variabilă nenegată în minterm şi o variabilă negată în maxterm şi invers pentru bitul 0. Ca o consecinţă, din aceste reguli de codificare, se pot demonstra următoarele relaţii:

$$S_i = \overline{P}_i; \qquad P_i = \overline{S}_i \tag{1.7}$$

adică termenul canonic sumă se obţine prin negarea termenului canonic produs şi invers.

La fel, pentru $i \neq j$, $i, j = 0, 1, 2, \ldots, 2^n - 1$, se pot demonstra relaţiile:

$$P_i \cdot P_j = 0; \qquad S_i + S_j = 1 \tag{1.8}$$

(pe baza $x + \overline{x} = 1$, $x \cdot \overline{x} = 0$, Axioma 6).

Valoarea unui termen P_i, respectiv S_i se poate modifica prin intermediul unui coeficient binar $d_i \in \{0, 1\}$ în felul următor:

$$a) \quad d_i \cdot P_i = \begin{cases} P_i & \text{dacă } d_i = 1 \\ 0 & \text{dacă } d_i = 0 \end{cases} \tag{1.9-a}$$

$$b) \quad d_i + S_i = \begin{cases} S_i & \text{dacă } d_i = 0 \\ 0 & \text{dacă } d_i = 1 \end{cases} \tag{1.9-b}$$

Definiţia 1.8 Forma canonică normală disjunctivă, FCND, a unei funcţii de n variabile este suma logică a tuturor termenilor de forma 1.9-a. $\diamond$

$$f(x_{n-1}, x_{n-2}, \ldots, x_1, x_0) = \sum_{i=0}^{2^n-1} d_i \cdot P_i \tag{1.10}$$

Definiţia 1.9 Forma canonică normală conjunctivă, FCNC, a unei funcţii de n variabile este produsul logic a tuturor termenilor de forma 1.9-b. $\diamond$

$$f(x_{n-1}, x_{n-2}, \ldots, x_1, x_0) = \prod_{i=0}^{2^n-1} (d_i + S_i) \tag{1.11}$$

Definiţia 1.10 Forma normală disjunctivă, FND, a unei funcţii de n variabile este o sumă numai de mintermi ai căror coeficienţi $d_i = 1$ $\diamond$

Forma FND se obţine din FCND prin eliminarea mintermilor ai căror coeficienţi au valoarea $d_i = 0$. Pentru exprimarea FND se introduce o reprezentare simbolică, sub forma unei liste

$$f(x_{n-1}, x_{n-2}, \ldots, x_1, x_0) = \sum_{i=0}^{2^n-1} (d_0, d_1, \ldots, d_{2^n-2}, d_{2^n-1}) \tag{1.12}$$

care conţine doar indicii acelor coeficienţi ai funcţiei care au valoare $d_i = 1$. De exemplu, pentru o funcţie de patru variabile:

$$f(x_3, x_2, x_1, x_0) = \sum_{i=0}^{15} (0, 3, 4, 5, 8, 9, 12, 14, 15)$$

această listă enumeră doar indicii coeficienţilor binari $d_0 = 1$, $d_3 = 1$, $d_4 = 1$, $d_5 = 1$, $d_8 = 1$, $d_9 = 1$, $d_{12} = 1$, $d_{14} = 1$ şi $d_{15} = 1$.

Definiţia 1.11 Forma normală conjunctivă, FNC, a unei funcţii de n variabile este un produs numai de maxtermi ai căror coeficienţi $d_i = 0$. $\diamond$

Forma FNC se obţine din FCNC prin eliminarea maxtermilor ai căror coeficienţi $d_i = 1$. Pentru exprimarea FNC se introduce o reprezentare simbolică sub forma unei liste

$$f(x_{n-1}, x_{n-2}, \ldots, x_1, x_0) = \prod_{i=0}^{2^n-1} (d_0, d_1, \ldots, d_{2^n-2}, d_{2^n-1}) \tag{1.13}$$

care conţine doar indicii acelor coeficienţi ai funcţiei care au valoarea $d_i = 0$. Luând ca exemplificare funcţia dată la relaţia 1.12 de data aceasta lista va fi

$$f(x_3, x_2, x_1, x_0) = \prod_{i=0}^{15} (1, 2, 6, 7, 10, 11, 13),$$

adică sunt enumeraţi doar indicii coeficienţilor binari care au valoarea zero.

Uneori este avantajos să se lucreze cu negata formei normale conjunctive sau cu negata formei normale disjunctive ale funcţiei care pot fi exprimate respectiv sub formele

$$\text{a)} \qquad \overline{f}(x_{n-1}, x_{n-2}, \ldots, x_1, x_0) = \prod_{i=0}^{2^n-1} (\overline{d}_i + S_i) \tag{1.14-a}$$

$$\text{b)} \qquad \overline{f}(x_{n-1}, x_{n-2}, \ldots, x_1, x_0) = \sum_{i=0}^{2^n-1} (\overline{d}_i \cdot P_i) \tag{1.14-b}$$

Aceste forme de scriere se pot obţine pornind de la relaţiile 1.10 şi 1.11, prin negarea ambelor părţi, apoi aplicarea teoremei lui DeMorgan şi ţinând cont de relaţiile 1.7, în felul următor:

$$\overline{f}(x_{n-1}, x_{n-2}, \ldots, x_1, x_0) = \overline{\sum_{i=0}^{2^n-1} (d_i \cdot P_i)} = \prod_{i=0}^{2^n-1} (\overline{d}_i + \overline{P}_i) = \prod_{i=0}^{2^n-1} (\overline{d}_i + S_i)$$

$$\overline{f}(x_{n-1}, x_{n-2}, \ldots, x_1, x_0) = \overline{\prod_{i=0}^{2^n-1} (d_i + S_i)} = \sum_{i=0}^{2^n-1} (\overline{d}_i \cdot \overline{S}_i) = \sum_{i=0}^{2^n-1} (\overline{d}_i \cdot P_i)$$

Aceste exprimări se pot deduce şi prin raţionament din relaţiile 1.10 şi 1.11. Din FCND se obţine FND dacă se consideră numai termenii canonici produs care au valoarea 1, relaţia 1.12, respectiv din FCNC se obţine FNC, relaţia 1.13, dacă se consideră numai termenii canonici sumă care au valoarea 0. Sub forma FND funcţia are valoarea 1 pentru suma tuturor termenilor canonici care au coeficienţii $d_i = 1$ şi are valoarea 0 pentru suma tuturor termenilor canonici care au coeficienţii $d_i = 0$. Evident că funcţia negată $\overline{f}$ va avea valoarea 1 pentru suma tuturor acelor termeni canonici care produc valoarea 0 pentru f, adică pentru acei coeficienţi d_i care sunt 0; respectiv funcţia negată $\overline{f}$ va avea valoarea 0 pentru suma tuturor acelor termeni canonici care produc valoarea 1 pentru f, adică pentru acei coeficienţi $d_i = 1$. Deci funcţia negată $\overline{f}$ poate fi scrisă ca o formă FND, relaţia 1.14-b, de acei termeni produs pentru care $\overline{d}_i = 1$, adică pentru acei coeficienţi d_i care au valoarea 0 la scrierea funcţiei f sub formă FND. Acelaşi raţionament se poate face şi pentru forma FNC, adică se scrie funcţia f pentru coeficienţii d_i care au valoarea 0 iar funcţia $\overline{f}$, relaţia 1.14-a, pentru coeficienţii care au valoarea 1, adică $\overline{d}_i = 0$.

O modalitate uzuală de definire a unei funcţii logice este cea prin tabelul de adevăr.

Definiţia 1.12 Tabelul de adevăr este un tabel care în prima coloană din stânga, coloana de intrare, listează toate configuraţiile de valori ale variabilelor de intrare $X = \{0,1\}^n$, iar în următoarele coloane, coloane de ieşire, sunt listate valorile, din $Y \subseteq \{0,1\}$, corespunzătoare ieşirilor. $\diamond$

Astfel de tabele de adevăr au fost prezentate iniţial în *Tabelul 1.1* pentru introducerea operatorilor booleeni NOT, AND, OR iar apoi în Figurile 1.1, 1.2 şi *Tabelul 1.2*, respectiv pentru funcţile logice de una, două şi trei variabile.

Exemplul 1.1 Pentru o celulă sumator complet să se deducă funcţia logică sumă, s_i şi funcţia logică pentru transferul următor, C_i.

Soluţie. În secţiunea 2.5.2 se va analiza sumarea a două cuvinte binare cu lungimea de n biţi $A = A_{n-1}A_{n-2}\ldots A_i\ldots A_1A_0$ şi $B = B_{n-1}B_{n-2}\ldots B_i\ldots B_1B_0$. Operaţia de sumare a celor două cuvinte se realizează pentru fiecare pereche de biţi (A_i, B_i) începând cu perechea $i = 0$, de rangul cel mai puţin semnificativ, (2^0), până la perechea $i = n-1$, de rangul cel mai semnificativ, (2^{n-1}). Pentru fiecare rang această sumare este efectuată cu o celulă sumator complet. O celulă sumator complet pentru rangul i are trei intrări A_i, B_i, C_{i-1} care sunt respectiv biţii celor două cuvinte A şi B şi transportul anterior C_{i-1} ce a fost generat de celula din rangul $2^{(i-1)}$. Semnalele generate la ieşire de către celula sumator complet sunt doi biţi: s_i –sumă $(= A_i + B_i + C_{i-1})$ şi C_i –transportul următor care se aplică la celula de rang $2^{(i+1)}$ ca transport anterior. **Celula sumator complet este notată uneori cu simbolul** $\sum(3,2)$, indicând faptul că are trei intrări şi două ieşiri. Există şi celula $\sum(2,2)$ care are numai două intrări A_i şi B_i (nu se consideră transportul anterior C_{i-1}) care este referită ca **celulă semisumator**. Pentru o celulă sumator complet tabelul de adevăr este prezentat în *Tabelul 1.6*.

Tabelul 1.6 Tabelul de adevăr pentru o celulă sumator/scăzător complet

Intrari			Adunare				Scadere		Numarator de 1		
A_i	B_i	$C_{i\,1}$ / $I_{i\,1}$	s_i	C_i	g_i	p_i	d_i	I_i	C_i	s_i	
0	0	0	0	0	0	0	0	0	0	0	(0)
0	0	1	1	0	0	0	1	1	0	1	(1)
0	1	0	1	0	0	1	1	1	0	1	(1)
0	1	1	0	1	0	1	0	1	1	0	(2)
1	0	0	1	0	0	1	1	0	0	1	(1)
1	0	1	0	1	0	1	0	0	1	0	(2)
1	1	0	0	1	1	0	0	0	1	0	(2)
1	1	1	1	1	1	1	1	1	1	1	(3)

Pentru ieşirea sumă s_i forma normală canonică disjunctivă, FNCD conform relaţiei 1.10, este :

$$s_i(A_i, B_i, C_{i-1}) = \sum_{i=0}^{7} d_i \cdot P_i = d_0 \cdot P_0 + d_1 \cdot P_1 + d_2 \cdot P_2 + d_3 \cdot P_3 + d_4 \cdot P_4 + d_5 \cdot P_5 + d_6 \cdot P_6 + d_7 \cdot P_7$$

unde $d_i, i = 0, 1, 2, \ldots, 7$ sunt coeficienţii funcţiei din coloana s_i din *Tabelul 1.6*. Evident că produsele pentru care $d_i = 0$ pot fi eliminate deoarece au valoarea 0 şi se obţine forma normală disjunctivă, FND.

$$\begin{aligned} s_i(A_i, B_i, C_{i-1}) &= 1 \cdot P_1 + 1 \cdot P_2 + 1 \cdot P_4 + 1 \cdot P_7 = \\ &= \overline{A}_i \cdot \overline{B}_i \cdot C_{i-1} + \overline{A}_i \cdot B_i \cdot \overline{C}_{i-1} + A_i \cdot \overline{B}_i \cdot \overline{C}_{i-1} + A_i \cdot B_i \cdot C_{i-1} \end{aligned}$$

Se observă că FND se poate scrie direct luând numai acei mintermi pentru care funcţia are valoarea 1 în tabelul de adevăr,

$$s_i(A_i, B_i, C_{i-1}) = \sum_{0}^{7} (1, 2, 4, 7)$$

de unde şi expresia de "**sinteză pe bază de 1**". Aplicând axiomele şi teoremele din *Tabelul 1.2*, şi expresiile pentru XOR şi NXOR, forma normală disjunctivă pentru s_i se transformă în felul următor:

$$\begin{aligned} s_i(A_i, B_i, C_{i-1}) &= \overline{A}_i \cdot (\overline{B}_i \cdot C_{i-1} + B_i \cdot \overline{C}_{i-1}) + A_i \cdot (\overline{B}_i \cdot \overline{C}_{i-1} + B_i \cdot C_{i-1}) = \\ &= \overline{A}_i \cdot (B_i \oplus C_{i-1}) + A_i \overline{(B_i \oplus C_{i-1})} = A_i \oplus B_i \oplus C_{i-1} \end{aligned} \tag{1.15}$$

Dar din *Tabelul 1.6* se poate face şi sinteza funcţiei negate $\overline{s}_i$, aceasta va avea valori 1 ($\overline{d}_i = 1$) acolo unde s_i are valori 0 ($d_i = 0$), relaţia 1.14-b, deci forma normală disjunctivă a funcţiei negate se scrie direct examinând tabelul de adevăr:

$$\begin{aligned} \overline{s}_i(A_i, B_i, C_{i-1}) &= \sum_{i=0}^{7} (0, 3, 5, 6) = \\ &= \overline{A}_i \cdot \overline{B}_i \cdot \overline{C}_{i-1} + \overline{A}_i \cdot B_i \cdot C_{i-1} + A_i \cdot \overline{B}_i \cdot C_{i-1} + A_i \cdot B_i \cdot \overline{C}_{i-1} = \\ &= \overline{A}_i \cdot (\overline{B}_i \cdot \overline{C}_{i-1} + B_i \cdot C_{i-1}) + A_i \cdot (\overline{B}_i \cdot C_{i-1} + B_i \cdot \overline{C}_{i-1}) = \\ &= \overline{A_i \oplus B_i \oplus C_{i-1}} \end{aligned}$$

Se pune întrebarea care cale se alege pentru sinteza funcţiei, cea prin FND pentru funcţia negată sau cea prin FND pentru funcţia nenegată? Răspunsul este evident: prin calea care solicită mai puţin efort, adică cea care duce la o formă FND cu mai puţini mintermi.

Pentru funcţia C_i sinteza se face din tabelul de adevăr, de data aceasta pe bază de zerouri, adică prin formele conjunctive. Forma canonică normală conjunctivă, FCNC, conform exprimării din relaţia 1.11, se va scrie:

$$\begin{aligned} C_i(A_i, B_i, C_{i-1}) &= \prod_{i=0}^{7}(d_i + S_i) = \\ &= (d_0 + S_0) \cdot (d_1 + S_1) \cdot (d_2 + S_2) \cdot (d_3 + S_3) \cdot (d_4 + S_4) \cdot (d_5 + S_5) \cdot \\ &\quad \cdot(d_6 + S_6) \cdot (d_7 + S_7) \end{aligned}$$

unde d_i, $i = 0, 1, \ldots, 7$ sunt coeficienţii funcţiei din coloana C_i. Se pot elimina factorii pentru care $d_i = 1$ deci se ajunge la forma normală conjunctivă FNC,

$$C_i(A_i, B_i, C_{i-1}) = \prod_{0}^{7}(0, 1, 2, 4)$$

care se putea scrie direct prin inspectarea valorilor de zero ("**sinteză pe bază de 0**") şi scrierea produsului de maxtermi în felul următor:

$$\begin{aligned} C_i &= S_0 \cdot S_1 \cdot S_2 \cdot S_4 = \\ &= (A_i + B_i + C_{i-1}) \cdot (A_i + B_i + \overline{C}_{i-1}) \cdot (A_i + \overline{B}_i + C_{i-1}) \cdot (\overline{A}_i + B_i + C_{i-1}) \end{aligned}$$

Aplicarea proprietăţii de distributivitate la această expresie duce la $3 \times 3 \times 3 \times 3 = 81$ termeni produs care apoi sunt reduşi prin aplicarea axiomelor şi teoremelor algebrei Booleene.

A doua cale de sinteză pe bază de zerouri se poate face pentru funcţia negată $\overline{C}_i$ prin inspectarea tabelului de adevăr se aleg maxtermii pentru care funcţia are valoarea 1, relaţia 1.14-a. Rezultă forma normală conjunctivă, FNC, pentru funcţia negată

$$\begin{aligned} \overline{C}_i(A_i, B_i, C_{i-1}) &= \prod_{0}^{7}(3, 5, 6, 7) = \\ &= (A_i + \overline{B}_i \cdot \overline{C}_{i-1}) \cdot (A_i + \overline{B}_i + C_{i-1}) \cdot (\overline{A}_i + \overline{B}_i + C_{i-1}) \cdot \\ &\quad \cdot(\overline{A}_i + \overline{B}_i + \overline{C}_{i-1}) \end{aligned}$$

şi de data aceasta se pot obţine 81 de termeni produs care pot fi reduşi. Uzual, se alege între sinteza prin FNC pentru funcţia negată sau sinteza prin FNC pentru funcţia nenegată prin observarea în tabelul de adevăr care cale duce la mai puţini maxtermi. Din aceste două tentative de sinteză pe bază de zero se constată dificultatea aplicării formelor conjunctive, acesta este unul din argumentele pentru care în practică se aplică aproape în exclusivitate sinteza pe bază de 1, adică FND, fie pentru funcţia negată $\overline{f}$, fie pentru funcţia nenegată f.

În consecinţă, revenind la sinteza pe bază de 1 rezultă pentru C_i

$$\begin{aligned} &\text{a) } C_i(A_i, B_i, C_{i-1}) = \sum_{i=1}^{7}(3, 5, 6, 7) = \\ &= \overline{A}_i \cdot B_i \cdot C_{i-1} + A_i \cdot \overline{B}_i \cdot C_{i-1} + A_i \cdot B_i \cdot \overline{C}_{i-1} + A_i \cdot B_i \cdot C_{i-1} \qquad (1.16) \end{aligned}$$

o formă rezonabilă, faţă de sintezele anterioare prin FNC şi care prin procedee analitice este adusă la următoarele forme disjunctive FD:

$$\begin{aligned} &\text{b) } C_i(A_i, B_i, C_{i-1}) = A_i \cdot (B_i \oplus C_{i-1}) + B_i \cdot C_{i-1} \\ &\text{c) } C_i(A_i, B_i, C_{i-1}) = \overline{\overline{C_{i-1} \cdot (A_i \oplus B_i)} \cdot \overline{(A_i \cdot B_i)}} \end{aligned}$$

1.1.5 Forme disjunctive şi conjunctive

Forma disjunctivă FD a funcţiei este o sumă de termeni necanonici de unde şi denumirea de sumă de produse, iar **forma conjunctivă FC** a funcţiei este un produs de termeni sumă necanonici, denumită şi produs de sume. Reducerea formelor normale disjunctive (FND) sau normale conjunctive (FNC), respectiv la FD sau FC este referită ca un procedeu de minimizare a funcţiei, deşi uneori nu se obţine forma minimă sau se obţin mai multe expresii (neminime).

Există trei modalităţi de minimizare:

1 - analitică, utilizând axiomele şi teoremele algebrei Booleene. Această cale este eficientă doar pentru funcţii cu număr mic de variabile şi de termeni.

2 - prin **metode grafice** (de exemplu diagrama Veitch - Karnaugh), de asemenea utilizabile pentru funcţii care nu au mai mult de 5-6 variabile.

3 - pe baza unor **algoritmi sau abordări euristice** . Există algoritmi (de exemplu Quine - McQlusky) care pot fi aplicaţi unor funcţii cu zeci de variabile şi ieşiri multiple. Aceşti algoritmi stau la baza programelor de minimizare (de exemplu Espresso - MV) incluse în medii de Programare Automată în Electronică, referite prin termenul **tehnici EDA** (**E**lectronic **D**esign **A**utomation).

Minimizarea pe cale analitică implică puţină practică în utilizarea axiomelor şi teoremelor algebrei booleene. Această practică nu se bazează pe un algoritm anume ci mai mult pe intuiţie. Astfel se pot adăuga termeni (tautologia) care apoi se grupează cu alţii încât să se aplice (cel mai frecvent) axioma de existenţă a complementului $(x + \overline{x} = 1)$, teorema absorbţiei sau absorbţia inversă.

Exemplul 1.2 Pentru următoarea formă normală disjunctivă

$$\begin{aligned} F(A,B,C,D) \;=\; & \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,C\,\overline{D} + \overline{A}\,B\,\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C}\,D + \overline{A}\,B\,C\,D + \\ & + A\,B\,\overline{C}\,D + A\,B\,C\,D + A\,B\,C\,\overline{D} + A\,\overline{B}\,\overline{C}\,\overline{D} + A\,\overline{B}\,C\,\overline{D} \end{aligned}$$

să se deducă forma minimă.

Soluţie. Se grupează câte doi termeni canonici pentru a se aplica axioma de existenţă a complementului (dar pentru aceasta unii termeni $\overline{A}\,B\,\overline{C}\,D$, $A\,B\,C\,D$ se dublează) în felul următor:

$$\begin{aligned} F(A,B,C,D) \;=\; & \overline{A}\,\overline{B}\,\overline{D}\,(\overline{C}+C) + (\overline{A}\,B\,\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C}\,D) + (\overline{A}\,B\,\overline{C}\,D + A\,B\,\overline{C}\,D) + \\ & + (\overline{A}\,B\,C\,D + A\,B\,C\,D) + (A\,B\,C\,D + A\,B\,C\,\overline{D}) + A\,\overline{B}\,\overline{D}(\overline{C}+C) = \\ = \; & \overline{A}\,\overline{B}\,\overline{D} + \overline{A}\,B\,\overline{C}\,(\overline{D}+D) + B\,\overline{C}\,D(\overline{A}+A) + B\,C\,D(\overline{A}+A) + \\ & + A\,B\,C(\overline{D}+D) + A\,\overline{B}\,\overline{D} = \\ = \; & \overline{B}\,\overline{D}(\overline{A}+A) + \overline{A}\,B\,\overline{C} + B\,\overline{C}\,D + B\,C\,D + A\,B\,C = \\ = \; & \overline{B}\,\overline{D} + B\,D(\overline{C}+C) + B(\overline{A}\,\overline{C} + A\,C) = \\ = \; & \overline{B}\,\overline{D} + B\,D + B(\overline{A}\,\overline{C} + A\,C) \end{aligned}$$

pentru care aplicând expresia funcţiei SAU EXCLUSIV NEGAT se obţine

$$F(A,B,C,D) = \overline{B \oplus D} + B\,\overline{(A \oplus C)}$$

Uneori se pune problema de a se parcurge traseul invers adică pentru o formă minimă să se deducă forma normală disjunctivă din care a provenit. În general, pentru o astfel de extindere de la un termen produs la un termen canonic produs se introduc în termenul produs variabilele care lipsesc sub forma $x + \overline{x}$. De asemenea, pentru ca o formă disjunctivă să fie extinsă la forma normală conjunctivă (produse de sume) suma de termeni produs trebuie transformată într-un produs de sume utilizând axioma distributivităţii $A + B\,C = (A + B)\,(A + C)$, iar în termenii sumă variabilele care lipseau se introduc prin axioma de existenţă a complemntului $x \cdot \overline{x} = 0$.

Exemplul 1.3 Următoarea formă disjunctivă $F(A, B, C) = A\,B + \overline{A}\,C$ să fie extinsă la forma normală conjunctivă.

Soluţie. În primul rând termenii produs sunt convertiţi în termeni sumă utilizând axioma de distributivitate.

$$\begin{aligned} F(A,B,C) &= (A\,B + \overline{A})\,(A\,B + C) = (A + \overline{A})\,(B + \overline{A})\,(A + C)\,(B + C) = \\ &= (\overline{A} + B)\,(A + C)\,(B + C) \end{aligned}$$

Fiecărui termen sumă îi lipseşte o variabilă care se introduce în felul următor:

$$\begin{aligned} \overline{A} + B &= \overline{A} + B + C\,\overline{C} = (\overline{A} + B + C)\,(\overline{A} + B + \overline{C}) \\ A + C &= A + C + B\,\overline{B} = (A + B + C)\,(A + \overline{B} + C) \\ B + C &= B + C + A\,\overline{A} = (A + B + C)\,(\overline{A} + B + C) \end{aligned}$$

În final se obţine:

$$F(A,B,C) = (A + B + C)\,(A + \overline{B} + C)\,(\overline{A} + B + C)\,(\overline{A} + B + C) = \prod_{0}^{7}(0, 2, 4, 5)$$

Expresiile sub formă de sume de produse sau produse de sume se pot modela pe două niveluri de operatori respectiv pe AND-OR sau OR-AND. De exemplu următoarele forme FD şi FC:

$$F_1 = A\,B + C\,D \quad \text{şi} \quad F_2 = (A + B)(C + D)$$

pot fi modelate ca în Figura 1.4-a şi 1.4-b. Uneori se impune ca modelarea să se facă fie numai cu operatorul NAND sau fie numai cu operatorul NOR. Pentru forme FD modelarea se face uşor pe baza operatorului NAND (notăm simbolic prin $\overline{A \cdot B} \rightarrow \ \nearrow (A, B)$) deoarece aplicând sumei de produse teorema dublei negaţii şi apoi teorema lui DeMorgan se obţine tocmai un NAND de NAND-uri. În schimb pentru forme FC modelarea se face mai uşor pe baza operatorului NOR (notăm simbolic prin $\overline{A + B} \rightarrow \ \swarrow (A, B)$) deoarece aplicând produsului de sume teorema dublei negaţii şi apoi teorema lui DeMorgan se obţine tocmai un NOR de NOR-uri.

Astfel F_1 şi F_2 devin:

$$\begin{aligned} \overline{\overline{F}}_1 &= \overline{\overline{A\,B + C\,D}} = \overline{\overline{A\,B} \cdot \overline{C\,D}} \quad \longrightarrow \quad \nearrow (\nearrow (A, B), \nearrow (C, D)) \\ \overline{\overline{F}}_2 &= \overline{\overline{(A + B)\,(C + D)}} = \overline{\overline{(A + B)} + \overline{(C + D)}} \quad \longrightarrow \quad \swarrow (\swarrow (A, B), \swarrow (C, D)) \end{aligned}$$

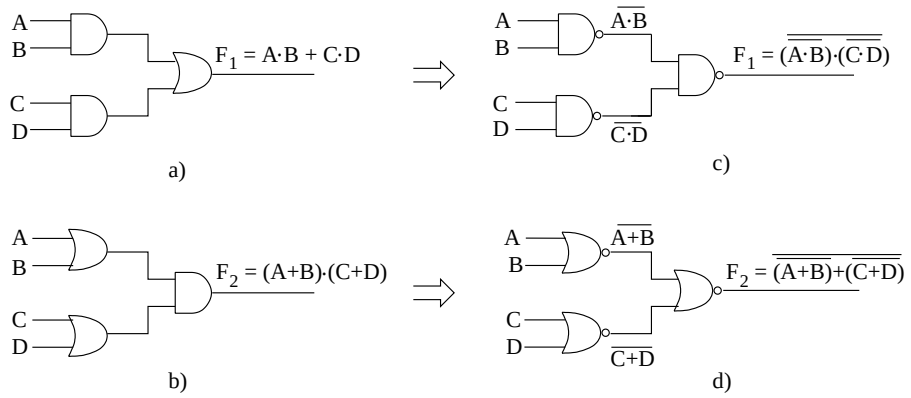


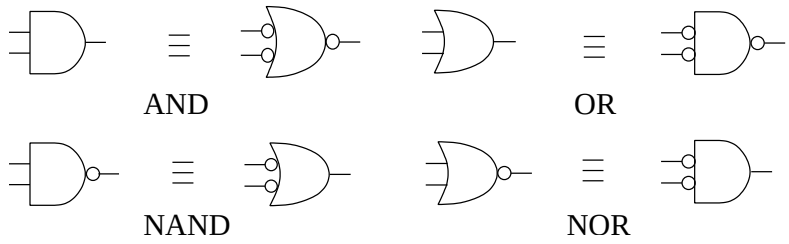
Figura 1.4 Implementarea formelor FD şi FC: a,b) pe două niveluri AND-OR, respectiv OR-AND; c,d) pe două niveluri NAND-NAND, respectiv NOR-NOR.

Numărând simbolurile $\nearrow$ şi $\swarrow$ rezultă că pentru fiecare funcţie sunt necesari trei operatori NAND respectiv NOR cu câte două intrări cum se poate vedea şi în Figura 1.4-c şi 1.4-d. Variabilele negate pot fi obţinute prin $\overline{x} \rightarrow \nearrow (x, x)$ sau $\swarrow (x, x)$. Uneori pentru conversia modelării de tip AND-OR sau OR-AND, respectiv într-o modelare de tip NAND-NAND sau NOR-NOR se pot utiliza anumite reguli grafice de transformare, care rezultă din axiomele şi teoremele algebrei Booleene. Aceste reguli sunt:

Regula 1. La ieşirea unui operator se poate introduce un cerculeţ de negaţie dar atunci trebuie introdus un cerculeţ de negaţie şi la intrarea operatorului imediat următor ("Dubla negaţie este adevăr"), deci pe conexiunea între cei doi operatori variabila nu suferă modificări;

Regula 2. Introducerea unui cerculeţ de negaţie pe ieşirea unei funcţii trebuie urmată, tot pe ieşire, de adăugarea încă a unui cerculeţ de negaţie (buffer inversor). Respectiv, introducerea la o variabilă de intrare, într-un operator, a unui cerculeţ de negaţie trebuie precedată de negarea aceleiaşi variabile de intrare (aplicarea pe intrare a variabilei negate);

Regula 3. Când se face conversia unui simbol grafic iniţial într-unul final, AND/NAND $\leftrightarrow$ OR/NOR, se pot introduce la simbolul final cerculeţe de negaţie fie la intrare, fie pe ieşire, fie atât la intrare cât şi la ieşire dar numai dacă la terminalele corespunzătoare ale simbolului iniţial nu a existat un cerculeţ de negaţie în felul următor (de fapt această regulă nu este decât o aplicare grafică a teoremei lui DeMorgan):



Exemplul 1.4 Pentru conversiile din Figura 1.4 să se aplice regulile grafice de transformare.

Soluţie. Aplicând regulile de conversie grafică se obţin succesiunile de circuite ca în Figura 1.5.

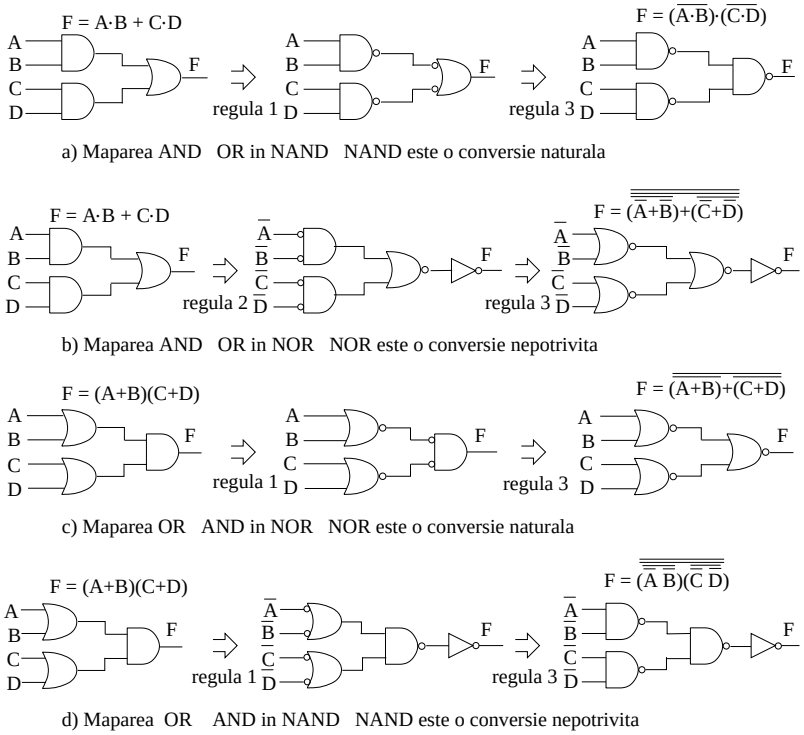


Figura 1.5 Exemple de conversii grafice de tipul AND-OR/OR-AND în NAND-NAND sau NOR-NOR.

Uneori apare şi problema inversă a modelării, pentru o modelare dată să se determine expresia FD sau FC care îl descrie, de fapt această abordare pentru un sistem deja implementat este referită ca analiza sistemului. Se poate obţine funcţia logică a modelului prin următoarele trei modalităţi:

1 - pentru fiecare configuraţie de valori logice ale variabilelor de intrare se deduc

valori logice în punctele intermediare ale modelului şi respectiv se calculează valoarea logică de ieşire şi în felul acesta se construieşte tabelul de adevăr. Apoi, din tabelul de adevăr rezultat se deduce funcţia logică.

2 - se scriu expresiile logice după fiecare simbol de operator logic, pornind de la fiecare intrare înspre ieşire, rezultând după ultimul operator expresia logică a funcţiei. Această modalitate este asemănătoare cu prima: se merge de la intrare schemei spre ieşire din punct în punct, dar la prima modalitate cu valorile funcţiei pe când la această modalitate cu expresiile funcţiei.

3 - utilizând convenţiile prin cerculeţe de negaţie.

A treia modalitate este recomandată doar atunci când operatorii au ieşirile negate (NAND, NOR) sau au intrări negate, deci parcurgerea de la intrare spre ieşire ridică anumite dificultăţi. De ce? pentru că suntem mai obişnuiţi să operăm cu operatorii AND şi OR cu intrări nenegate. De fapt, aceasta se reduce tot la a doua modalitate dar prin conversia operatorilor care conţin cerculeţe de negaţie în operatori fără aceste cerculeţe, se ajunge la o reprezentare numai cu operatori AND şi OR.

Exemplul 1.5 Pentru modelele de circuite din Figura 1.6-a, 1.6-c să se deducă expresia funcţiilor (aplicând cerculeţele de negaţie).

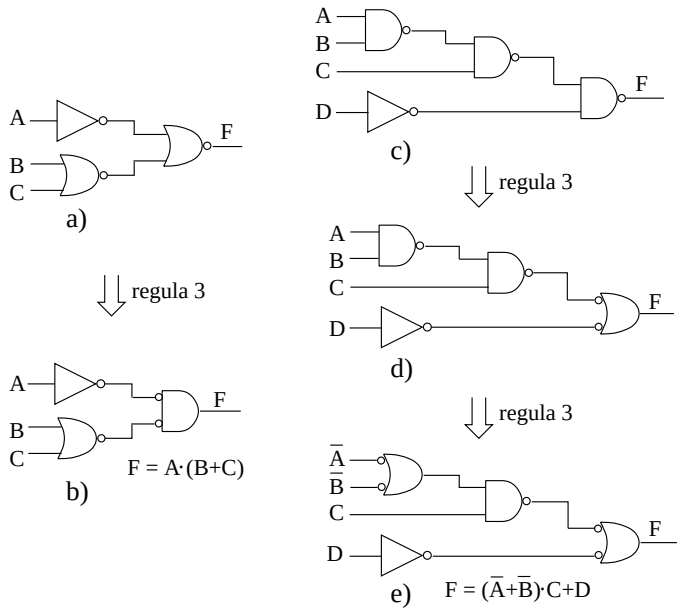


Figura 1.6 Exemple de conversie a modelelor logice spre structuri cu operatori AND şi OR pentru determinarea funcţiilor logice.

Soluţie. Pentru modelul din Figura 1.6-a se aplică regula 3 şi se obţin numai operatori AND şi OR, $F = A \cdot (B + C)$. Pentru modelul din Figura 1.6-c se aplică de două ori regula 3 şi structura poate fi considerată ca fiind compusă doar din operatori AND şi OR pentru care se scrie foarte simplu expresia logică, $F = D + C \cdot (\overline{A} + \overline{B})$.

Acest mod de transformare a modelelor logice este foarte indicat în depanarea circuitelor deoarece pornind de la intrare spre ieşire, după fiecare nivel logic de OR sau AND se poate verifica simplu corectitudinea semnalelor.

În încheierea prezentării acestor noţiuni de suport formal pentru sistemele digitale accentuăm faptul că formele normale disjunctive, FND, sau formele disjunctive, FD, (sumă de produse) pot fi modelate pe două nivele logice AND-OR. De asemenea, formele normale conjunctive, FNC, sau formele conjunctive, FC, (produse de sume) pot fi modelate pe două nivele logice OR-AND. Aceste afirmaţii sunt adevărate în mod teoretic când operatorii AND, OR se aplică pentru orice număr de intrări şi există disponibile şi variabilele negate. În practică, unde un operator este o poartă logică, trebuie luate cu precauţie aceste afirmaţii în funcţie de porţile disponibile şi de numărul de intrări ale acestora (adică de restricţiile electrice de conectivitate ale acestor porţi).

1.2 POARTA LOGICĂ

Când se trece de la formele FND, FNC la FD, FC şi de la modelele acestora pe bază de operatori NOT, AND, OR, NOR, NAND, la implementări reale pe bază de circuite electronice pentru operatorii logici se foloseşte, în exprimare, termenul de poartă logică. Prin termenul de **poartă logică** se face referire la orice circuit electronic care implementează un operator logic, deci există poarta inversor (NOT), poarta AND, poarta OR, poartă XOR, poarta NAND etc. Prin referirea tuturor circuitelor logice cu termenul de poartă logică apare un abuz de limbaj care, totuşi, are o justificare din punct de vedere al transferului semnalului (variabile logice binare) prin circuit. Considerând operatorii logici prezentaţi în *Tabelul 1.7*, cu cele două intrări A,C şi ieşirea f, se poate analiza cum pentru transferul semnalului logic A spre ieşirea f, prin condiţionarea de către semnalul C (de control), circuitul electronic care implementează un astfel de operator are un comportament similar cu utilizarea/funcţionarea unei porţi. Astfel, când circuitul poartă este "deschis" lasă semnalul A să treacă modificat sau nemodificat spre ieşirea f, iar când poarta este "închisă" semnalul A nu se transferă la ieşirea f. Intuitiv, în această analogie, poarta este închisă sau deschisă de către cineva, adică în cazul unui circuit de un semnal de control, variabila C.

De exemplu, pentru poarta AND când este deschisă, $C = 1$, semnalul de intrare A se transferă la ieşire $f = A$, iar când este închisă, $C = 0$, semnalul de intrare A nu se transferă la ieşire, f este încontinuu la valoarea logică 0. Poarta XOR, este puţin mai specială, este permanent deschisă, transferă la ieşire semnalul de intrare nemodificat pentru $C = 0$, $f = A$ ($A \oplus 0 = A$), iar pentru $C = 1$ transferă spre ieşire intrarea A negată, $f = \overline{A}$ ($A \oplus 1 = \overline{A}$); **poarta XOR are o funcţionare de circuit inversor comandat**. O astfel de interpretare, de circuit poartă, se poate găsi pentru oricare din operatorii prezentaţi în *Tabelul 1.7*.

Analizând în paralel poarta AND şi poarta OR se poate observa că poarta AND este deschisă pentru $C = 1$ iar poarta OR este deschisă pentru $C = 0$, astfel spunem că una este deschisă când semnalul C este activat în 1 logic iar cealaltă când semnalul C este activat în 0 logic. Până acum ataşam valorii de adevăr, pentru o variabilă,

Tabelul 1.7 Interpretarea operatorilor logici ca circuite poartă

INTRARI		AND		OR		NAND		NOR		XOR		NXOR	
C	A												
0	0	0		0		1		1		0		1	
0	1	0		1		1		0		1		0	
1	0	0		1		1		0		1		0	
1	1	1		1		0		0		0		1	
Iesirea este :		f = 0	f = A	f = 1	f = A	f = 1	f = $\overline{A}$	f = 0	f = $\overline{A}$	f = A	f = $\overline{A}$	f = $\overline{A}$	f = A
Pentru variabila de control:		C = 0	C = 1	C = 1	C = 0	C = 0	C = 1	C = 1	C = 0	C = 0	C = 1	C = 0	C = 1

valoarea binară 1 şi pentru valoarea de fals valoarea binară 0. Dar, în practică, se poate ataşa pentru starea de adevăr a unei variabile acea valoare binară pe care variabila o are în **starea activă** (adică starea care produce acţiunea pentru care se justifică acea variabilă), care poate fi: fie bitul 1, fie bitul 0. Iar valoarea de fals variabila o are în **starea de neactivare**, care poate fi: fie pentru valoarea binară 1, fie pentru valoarea binară 0.

În această interpretare, variabila A când este în stare activă pentru valoarea binară 1 (activ High) se notează în felul următor A_H şi când este în starea activă pentru valoarea binară 0 (activ Low) se notează A_L . La fel, şi o ieşire OUT a unei porţi poate fi considerată în stare activă (adevărată) fie când are valoarea binară 0 şi se notează OUT_L, fie când are valoarea binară 1 şi se notează OUT_H . Cu aceste notaţii tabelele de adevăr pentru porţile logice se realizează nu pentru valorile binare 1 şi 0 ci pentru stările de activare (adevărat) şi neactivare (fals) ale intrărilor şi, respectiv, pentru starea de activare (adevărat) şi neactivare (fals) a ieşirii. Pe simbolurile grafice ale porţilor pentru semnalele de intrare şi de ieşire active în L se introduc cerculeţe de negaţie. În practică, dacă un semnal este activ în starea High, de exemplu A_H sau OUT_H, nu se mai adaugă sufixul H şi se reprezintă numai A sau

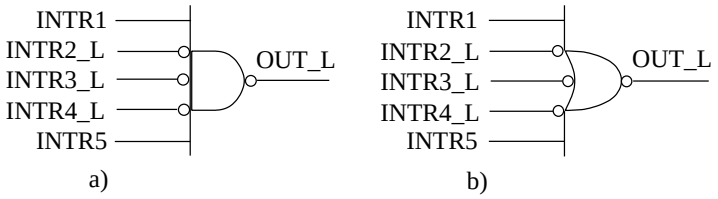


Figura 1.7 Reprezentarea mixată a semnalelor (pentru valoarea binară asignată stării de adevăr: a) pentru o poartă AND; b) pentru o poartă OR.

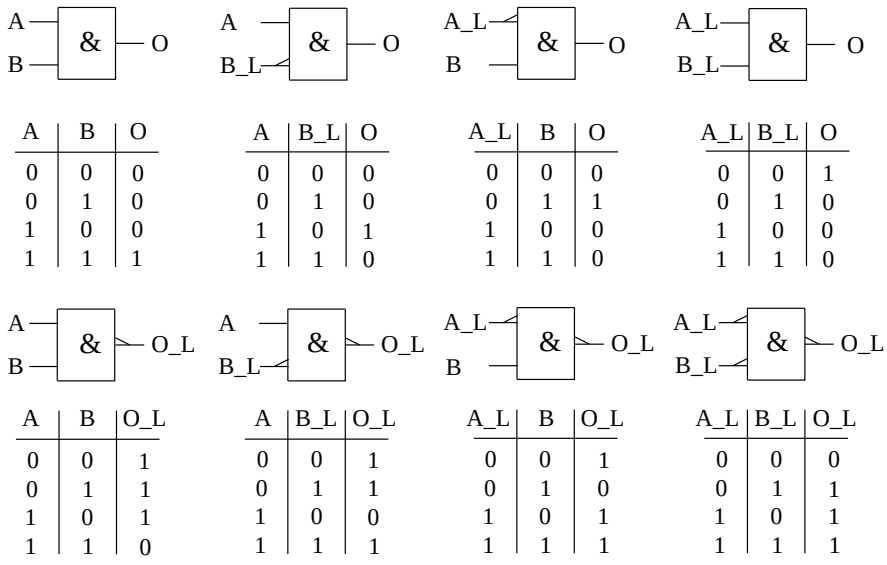
OUT înţelegându-se că au valoarea de adevăr (starea activă) în 1 logic iar valoarea de fals în 0 logic. În schimb, pentru semnalele active în starea Low se menţine sufixul *L*. Deci, poate să apară această notare mixată, adică semnalele active în starea High nu mai au sufixul *H* dar cele active în starea Low au sufixul *L*. De exemplu, pentru poarta AND din Figura 1.7-a ieşirea este activă (valoare de adevăr), $OUT\_L = 0$, se obţine când toate intrările sunt activate (au valoare de adevăr), adică: $INTR1 = 1$, $INTR2\_L = 0$, $INTR3\_L = 0$, $INTR4\_L = 0$ şi $INTR5 = 1$; iar pentru poarta OR din Figura 1.7-b ieşirea nu este activă (nu are valoare de adevăr, $OUT\_L = 1$) numai când nici una din intrări nu este activată (este falsă), adică: $INTR1 = 0$, $INTR2\_L = 1$, $INTR3\_L = 1$, $INTR4\_L = 1$ şi $INTR5 = 0$.

În practica circuitelor digitale pentru o variabilă VAR, care este activă în starea Low, se utilizează două notaţii: $\overline{VAR}$ sau VAR_L; **în această carte se va utiliza atât notaţia de variabilă negată, $\overline{VAR}$, cât şi notaţia de nivel Low, VAR_L, ambele având aceeaşi semnificaţie.**

Exemplul 1.6 Să se reprezinte tabelele de adevăr şi simbolurile grafice uniforme (vezi *Tabelul 1.1*) pentru toate variantele de activare ale variabilelor de intrare şi ieşirii la o poartă AND cu două intrări.

Soluţie. Ieşirea unei porţi AND cu două intrări este activă (are valoare de adevăr) numai când sunt active (au valori de adevăr) ambele intrări; deci o aplicaţie pe mulţimea

A	B	A·B	A+B
Fals	Fals	Fals	Fals
Fals	Adevarat	Fals	Adevarat
Adevarat	Fals	Fals	Adevarat
Adevarat	Adevarat	Adevarat	Adevarat



A	B	O
0	0	0
0	1	0
1	0	0
1	1	1

A	B_L	O
0	0	0
0	1	0
1	0	1
1	1	0

A_L	B	O
0	0	0
0	1	1
1	0	0
1	1	0

A_L	B_L	O
0	0	1
0	1	0
1	0	0
1	1	0

A	B	O_L
0	0	1
0	1	1
1	0	1
1	1	0

A	B_L	O_L
0	0	1
0	1	1
1	0	0
1	1	1

A_L	B	O_L
0	0	1
0	1	0
1	0	1
1	1	1

A_L	B_L	O_L
0	0	0
0	1	1
1	0	1
1	1	1

Figura 1.8 Variante de asignare a valorilor binare pentru semnalele active la o poartă AND

{adevăr,fals} cu valori tot în această mulţime. Ataşând fiecărei valori de adevăr (activare), pentru variabilele de intrare şi pentru ieşire, fie nivelul Low fie nivelul High rezultă 8 variante pentru tabelul de adevăr, Figura 1.8. Realizaţi variantele pentru tabelul de adevăr şi pentru poarta OR.

În tehnică, de foarte mult timp, este utilizat un element ce prezintă două stări: contactul unui releu. Un contact închis prin lamela sa stabileşte, între două puncte de circuit, un traseu de rezistenţă electrică de valoare foarte mică, teoretic zero, iar un contact deschis întrerupe un circuit între două puncte, deci realizează între cele două puncte o rezistenţă infinită. Celor două stări ale contactului (închis, deschis) li se pot asocia elementele mulţimii binare $\{0, 1\}$. Rezultă că pentru circuitele cu contacte, denumite **reţele de comutaţie**, se poate aplica formalismul algebrei Booleene, $B(2)$; apare astfel posibilitatea de a formaliza proiectarea şi analiza reţelelor de comutaţie. Acest formalism a fost prima dată folosit de Claude Shannon în 1938 care, a scos din arhiva matematicilor algebra concepută de George Boole (1852) şi a aplicat-o în acest scop.

În Figura 1.9 este schiţată o structură de circuit cu releu care, prin cele două contacte ale sale, unul normal deschis celălalt normal închis, realizează traseul circuitelor

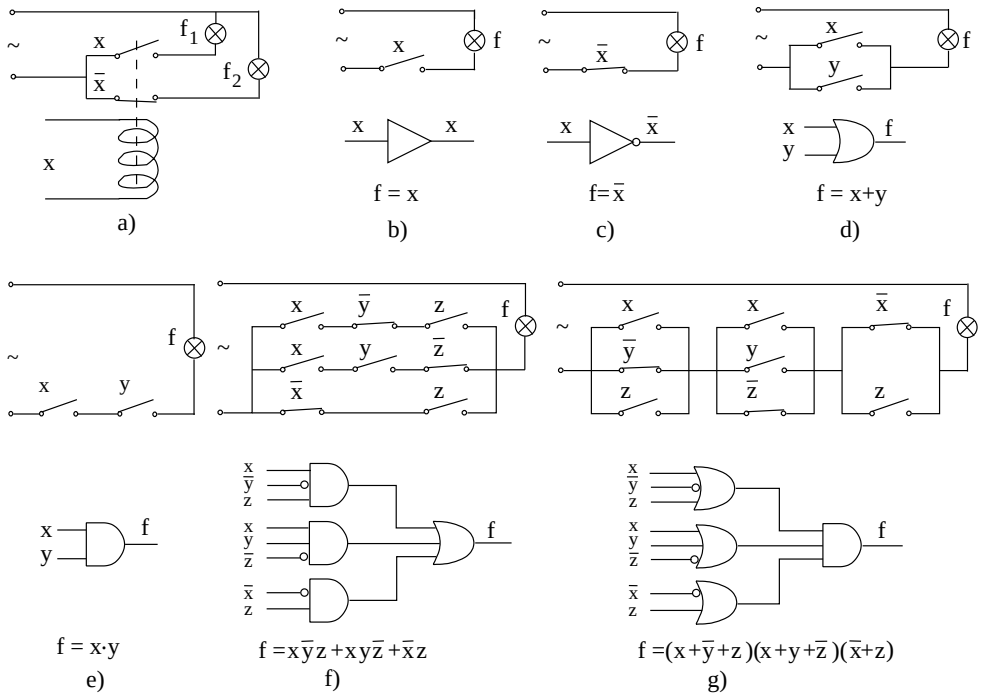


Figura 1.9 Reţele de comutaţie: a) structură de circuit releu; b,c,d,e) reţele de comutaţie pentru modelarea operatorilor de: identitate, negaţie, sumă logică şi produs logic; f,g) reţele de comutaţie pentru modelarea unei forme de sume-de-produse şi produse-de-sume.

de alimentare a două becuri notate cu f_1 şi f_2. Un **contact normal deschis** este în stare deschisă când releul nu este comandat şi trece în stare închisă când releul este comandat. Invers, **un contact normal închis** este în starea închisă când releul nu este comandat şi se deschide când releul este comandat. În această schiţă, când releul este necomandat tensiunea de alimentare a bobinei este 0, $x = 0$, armătura nu este atrasă, contactul normal închis este închis, cel normal deschis este deschis, iar becul f_1 este stins şi becul f_2 este aprins. Când se comandă releul, tensiunea de alimentare are o anumită valoare, pe care o putem nota cu 1, $x = 1$, contactul normal deschis se închde, cel normal închis se deschide deci becul f_1 se aprinde şi f_2 se stinge. Ataşând stării becurilor valoarea logică 1 (activ) când este aprins şi valoarea logică 0 (inactiv) când este stins rezultă că aplicaţia $\{0,1\} \rightarrow \{0,1\}$, conform Figurii 1.1, este funcţia f_1^1, adică identitate pentru becul f_1, $f_1 = x$ şi este funcţia f_2^1, adică negaţie pentru becul f_2, $f_2 = \overline{x}$. Evident, contactul normal deschis prin care se realizează funcţia de identitate se va nota cu variabila x, Figura 1.9-b, iar contactul normal închis prin care se realizează funcţia de negaţie se va nota cu variabila negată, $\overline{x}$, Figura 1.9-c. Rezultă, intuitiv, că două contacte înseriate realizează, pentru reţeaua de comutaţie respectivă, modelarea produsului logic de două variabile, Figura 1.9-e, iar două contacte în paralel realizează modelarea sumei logice a două variabile, Figura 1.9-d. O

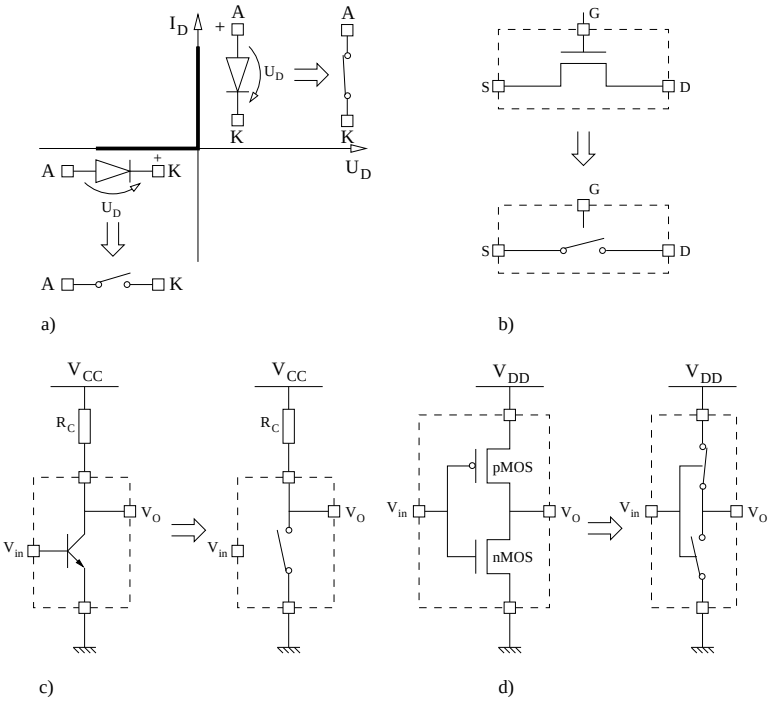


Figura 1.10 Elemente fizice componente de bază pentru structurarea porţilor logice: a,b) elemente componente pentru modelarea funcţiei de identitate. c,d) elementele componente pentru modelarea funcţiei de inversor în tehnologie bipolară şi în tehnologie CMOS.

reţea de comutaţie serie-paralel, Figura 1.9-f, modelează o funcţie disjunctivă (sumă de produse, FD) iar o reţea de comutaţie paralel-serie, Figura 1.9-g, modelează o funcţie conjunctivă (produse de sume, FC).

Elementele logice de bază în structura unei porţi sunt cele două funcţii f_1^1 (**identitate**) şi f_2^1 (**negaţie**) cu care se pot realiza diferite organizări prin intermediul celor patru tipuri de conectare: **serie**, **paralel**, **serie-paralel** şi **paralel-serie**. În circuitele poartă componentele fizice care pot modela funcţia identitate pot fi dioda sau tranzistorul de trecere. O diodă polarizată în sens direct este echivalentă unui contact închis iar polarizată în sens invers poate modela un contact deschis, Figura 1.10-a (s-a considerat caracteristica de diodă ideală). La fel, un tranzistor de trecere (vezi Figura 1.51) poate fi echivalentul unui contact închis/deschis după cum tranzistorul este comandat în conducţie/blocare, Figura 1.10-b.

Pentru funcţia de inversor există circuite simple în fiecare tehnologie. Structura unui circuit inversor în tehnologie bipolară este reprezentată în Figura 1.10-c (vezi secţiunea 1.5.2) iar pentru inversorul CMOS în Figura 1.10-d (vezi sectiunea 1.4.1).

Circuitul cu diode care produce la ieşire tensiunea de valoare minimă aplicată la intrare, $MIN(V_1, V_2, V_3)$, Figura 1.11-a şi circuitul cu diode care produce la ieşire tensiunea maximă aplicată la intrare, $MAX(V_1, V_2, V_3)$, Figura 1.11-b pot fi utilizate ca structuri de poartă AND sau OR în funcţie de convenţia de logică folosită.

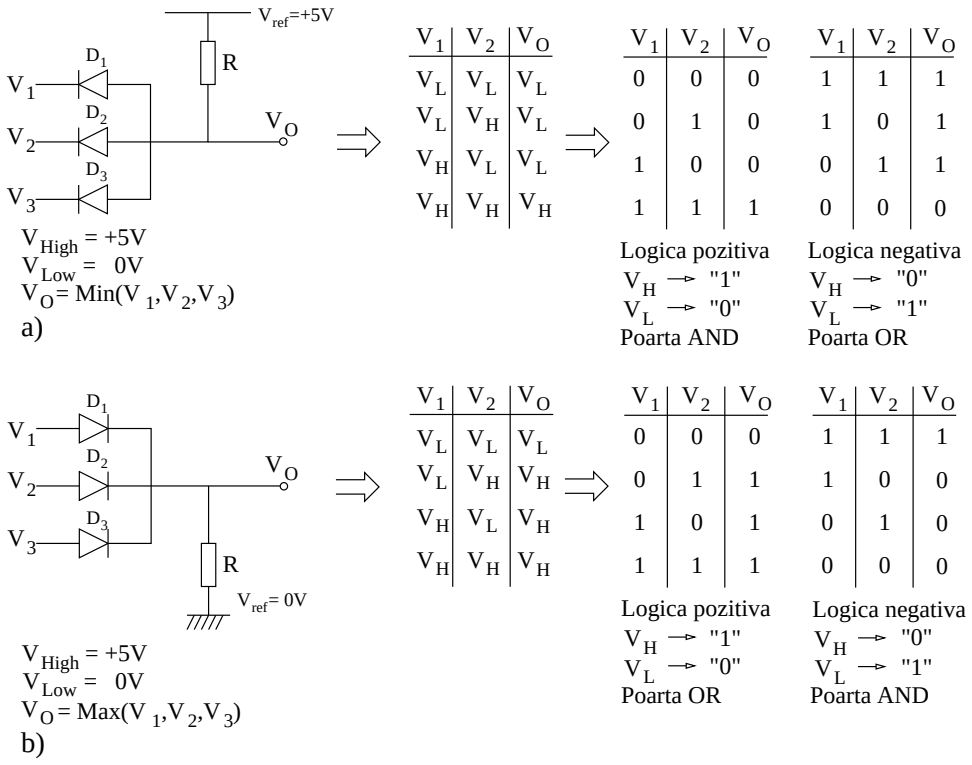


Figura 1.11 Echivalarea circuitelor MIN (a) şi MAX(b) ca porţi logice AND şi OR în funcţie de convenţia de logică pozitivă sau negativă

Prin **convenţia de logică pozitivă** într-un circuit nivelului de tensiune ridicat V_H (High), în general tensiunea de alimentare $V_H = V_{CC}$, $V_H = V_{DD}$, i se atribuie valoarea logică "1" iar nivelului de tensiune coborât V_L (Low), în general tensiunea de masă $V_L = V_{SS} = 0V$, i se atribuie valoarea logică "0". Invers, prin **convenţia de logică negativă** se fac următoarele atribuiri $V_H \rightarrow$ "0", $V_L \rightarrow$ "1". Există circuite logice care sunt alimentate cu tensiuni negative ($-V_{EE}$) faţă de masă, circuitele de tip **ECL**–**E**mitter **C**oupled **L**ogic, şi la acestea se păstrează atribuirile din convenţiile de logică pozitivă sau negativă; diferenţa faţă de circuitele care se alimentează la tensiune pozitivă este faptul că $V_H = 0V$ (tensiunea masei) şi $V_L = -V_{EE}$, Figura 1.13.

Circuitul MIN, Figura 1.11-a, va genera pentru ieşire $V_O = MIN(V_1, V_2, V_3)$ deoarece va conduce numai dioda care are aplicat pe catod tensiunea cu valoarea cea mai coborâtă, celelalte diode, cu catozii mai pozitivi, vor fi blocate. Considerând circuitul MIN realizat numai cu două diode D_1 şi D_2, la catozii cărora se aplică numai tensiunile de valori V_H sau V_L, se poate construi tabelul pentru tensiunea de ieşire V_O. Aplicând acestui tabel convenţia de logică pozitivă rezultă că circuitul MIN are o funcţionare de poartă AND, iar în convenţia de logică negativă are o funcţionare de poartă OR.

Circuitul MAX, Figura 1.11-b, va genera pentru ieşire tensiunea maximă aplicată pe intrare $V_O = MAX(V_1, V_2, V_3)$ deoarece va conduce doar dioda care are aplicat pe anod tensiunea cea mai ridicată, celelalte diode, cu potenţial mai coborât pe anod, vor fi blocate. Aplicând tabelului, care arată corespondenţa dintre tensiunea de ieşire V_O şi tensiunile de intrare V_1, V_2 (sunt considerate doar două intrări), convenţia de logică pozitivă rezultă că circuitul MAX are funcţionare de poartă OR iar în convenţia de logică negativă are o funcţionare de poartă AND.

Teorema 1.1 Duala unei funcţii de variabile negate este egală cu negata funcţiei de variabile nenegate.

$$\overline{f(x_0, x_1, \ldots, x_{n-1})} = f^D(\overline{x_0}, \overline{x_1}, \ldots, \overline{x_{n-1}}) \tag{1.17}$$

Relaţia 1.17 se poate verifica pe tabelele de adevăr din Figura 1.11-a şi 1.11-b deoarece o convenţie de logică se obţine din cealaltă convenţie prin negarea variabilelor iar operatorii AND şi OR sunt unul dualul celuilalt: $\overline{V_1 \cdot V_2} = \overline{V_1} + \overline{V_2}$ şi $\overline{V_1 + V_2} = \overline{V_1} \cdot \overline{V_2}$. De fapt, relaţia 1.17 este o generalizare a teoremei lui DeMorgan.

Exemplul 1.7 Să se conceapă o organizare de circuit, folosind circuitele MIN şi MAX, care să producă toţi maxtermii şi mintermii de două variabile. Se poate extinde aceasta pentru generarea de sume de produse de două variabile?

Soluţie. Mintermii de două variabile $\overline{x}_1\,\overline{x}_0$, $\overline{x}_1\,x_0$, $x_1\,\overline{x}_0$, $x_1\,x_0$ pot fi produşi de patru circuite MIN, ca în Figura 1.11-a, dar cu patru diode pe catozii cărora se aplică variabilele x_1, $\overline{x}_1$, x_0, $\overline{x}_0$. Cele patru circuite MIN se pun în paralel ca în Figura 1.12-a realizând o matrice AND. Maxtermii de două variabile $x_1 + x_0$, $x_1 + \overline{x}_0$, $\overline{x}_1 + x_0$, $\overline{x}_1 + \overline{x}_0$ pot fi generaţi de patru circuite MAX ca în Figura 1.11-b, cu variabilele de intrare x_1, $\overline{x}_1$, x_0, $\overline{x}_0$, iar cele patru circuite sunt conectate în paralel realizând matricea OR din Figura 1.12-b. (Atenţie care catozi sunt conectaţi la linii pentru circuitele MIN şi la coloane pentru circuitul MAX).

Evident că se pot obţine sume de produse de două variabile dacă cele patru ieşiri de la matricea AND sunt conectate, fiecare, la câte o intrare la matricea OR realizând o configuraţie

de circuit pe două niveluri logice, AND-OR. Când se realizează această înseriere de niveluri logice apar două probleme: prima, este căderea de tensiune pe joncţiunile în conducţie, ceea ce duce ca semnalul de ieşire în stare logică 1 să fie sub valoarea V_H, iar a doua, este lipsa unei decuplări (izolare) între intrare şi ieşire. Pentru valorile următoare $V_H = 5V$, $V_L = 0V$, $V_{Don} = 0,7V$, $R_1 = 1K\Omega$, $R_2 = 10K\Omega$ rezultă tensiunea de ieşire în starea H egală cu

$$V_H - \left[(V_H - V_{Don}) \cdot \frac{R_1}{(R_1 + R_2)}\right] = 3,9V \leq V_H = 5V$$

Structura obţinută prin înserierea unei matrice AND cu o matrice OR poate fi suport pentru implementarea oricărei funcţii sumă de produse dacă matricile respective sunt programabile, adică pe nivelul AND se poate genera oricare termen produs iar pe nivelul OR se poate selecta oricare produs în obţinerea unei sume (vezi sectiunea 2.4.7). Fizic, generarea acestor termeni cu structura similară celei din Figura 1.12 se face, într-un nod al celor două matrice, prin neconectarea sau conectarea diodei la linie respectiv la coloană, ceea ce se reduce la arderea sau menţinerea unui fuzibil înseriat cu dioda dintr-un nod. Prin fabricaţie, realizând în fiecare nod o diodă înseriată cu un fuzibil, i se oferă utilizatorului ca ulterior să aibă posibilitatea de a programa fiecare nod din matricea AND şi matricea OR în funcţie de expresia particulară a funcţiei exprimată ca sumă de produse. În procesul de programare, cu ajutorul unui aparat programator, utilizatorul poate selecta oricare nod din cele două matrice şi prin aplicarea unei tensiuni de valoare $10 \div 30V$ să ardă fuzibilul. Dispozitivele programabile de către utilizator sunt foarte eficiente şi flexibile în etapa de dezvoltare a unui produs când se încearcă diferite variante până la obţinerea variantei finale. Producătorii de dispozitive programabile produc aceste dispozitive cu toate nodurile matricei înzestrate fie cu fuzibil fie cu un antifuzibil.

La dispozitivele cu fuzibil, prin programare, rezistenţa conexiunii dintr-un nod este modificată de la valoarea zero la valoara infinită (arderea fuzibilului). Un fuzibil din tungsten-titan sau nichel-crom cu lăţimea de $0.15\mu m$ necesită pentru ardere un curent de $10 \div 60mA$ timp de $1 \div 10ms$ (Texas Instruments), iar un fuzibil din polisiliciu cu

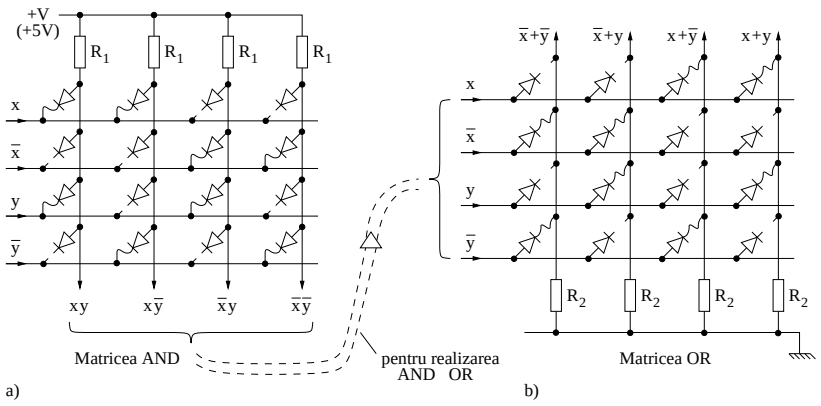


Figura 1.12 Organizarea unei matrice programabile în logica pozitivă: a)pentru termeni produs; b)pentru termeni sumă.

lăţimea de $25\mu m$ se poate "arde" cu un curent de $20 \div 80mA$ timp de $15\mu s$ (AMD). Dar, există pericolul ca în timp, datorită efectelor termice din circuitul integrat, unele fuzibile arse să ducă la refacerea conexiunii.

La dispozitivele cu antifuzibil, prin programare, rezistenţa conexiunii dintr-un nod este modificată de la o valoare iniţială foarte mare de ordinul $100M\Omega$ la o valoare sub $1K\Omega$. Trasei de antifuzibil realizată fie dintr-un dielectric (de exemplu ONO, Oxigen-Nitrura-Oxigen), fie din siliciu amorf, în procesul de programare, i se aplică un curent de ordinul zeci de mA cu durata sub $1\mu s$, prin aceasta producând modificarea rezistenţei. Conexiunile pe bază de antifuzibil, spre deosebire de cele pe bază de fuzibil, nu se pot reface întâmplător (adică să revină la rezistenţe de $100M\Omega$).

Dispozitivele programabile atât cele cu fuzibil cât şi cele cu antifuzibil sunt referite ca dispozitive o singură dată programabile,**OTP** (**O**ne **T**ime **P**rogrammable).

1.3 PARAMETRII PORŢILOR LOGICE

Până în prezent s-a parcurs traseul de la expresie logică la un model al acesteia pe o reţea de operatori care, fizic, sunt porţi logice. Dar, într-o implementare de sistem conectarea porţilor logice nu este fără limite ci trebuie să se respecte anumite restricţii care sunt exprimate prin anumite valori şi care sunt referite prin termenul de **parametrii de catalog** ai porţilor logice. Prin parametrii (de catalog) ai unei porţi logice se înţeleg acele valori care îi caracterizează funcţionarea sa în interconectarea cu alte porţi din aceeaşi familie sau în condiţii de test. Uneori, poarta logică este interconectată şi cu alte porţi din alte familii ceea ce impune o specificare unificată a parametrilor pentru toate familiile de porţi logice.

Parametrii unei porţi trebuie să caracterizeze regimul de curent continuu, regimul tranzitoriu şi regimul de zgomot. Frecvent, aceşti parametri sunt daţi în catalog ca valori tipice (normale), precum şi cu valorile pentru cazul cel mai defavorabil. **Cazul cel mai defavorabil** presupune că: circuitul cel mai defavorabil, din lotul admis, este în condiţiile cele mai defavorabile de funcţionare (temperatură, umiditate, tensiune de alimentare). Gama uzuală de temperatură este între 0°C şi 70°C, pentru aplicaţiile civile, şi se extinde la intervalul $-55°C \div +125°C$, pentru aplicaţiile militare. Condiţiile defavorabile pentru tensiunea de alimentare se specifică prin abaterile faţă de valoarea nominală V, adică prin valorile $V \pm \Delta V$. În proiectare, chiar şi pentru cazul cel mai defavorabil, poarta trebuie să realizeze valori pentru parametrii săi care să nu iasă din plaja valorilor prevăzute în catalog. Când funcţionarea porţii se face în afara condiţiilor specificate mai sus se impune măsurarea parametrilor săi şi aceşti parametri măsuraţi să fie în limitele valorilor de catalog pentru a putea fi interconectate cu alte porţi.

Nivelurile de tensiune. Într-un circuit logic se poate face referire la două niveluri constante de tensiune: **tensiunea de alimentare** V_{DD}, V_{CC}, $-V_{EE}$ şi **tensiunea de masă** $0V$ sau V_{SS}; aceste două niveluri de tensiune sunt notate prin V_H şi V_L şi li se pot asigna cele două valori ale mulţimii binare $B = \{0, 1\}$, fie conform convenţiei de logică pozitivă, fie conform convenţiei negative. De fapt, cele două niveluri de tensiune, practic, se extind la două intervale de tensiune notate cu ΔV_H şi ΔV_L ca în Figura 1.13, deci oricare valoare a tensiunii din aceste intervale (reprezintă V_H respectiv V_L) şi corespunde cifrei binare 1 sau 0 conform convenţiei de logică

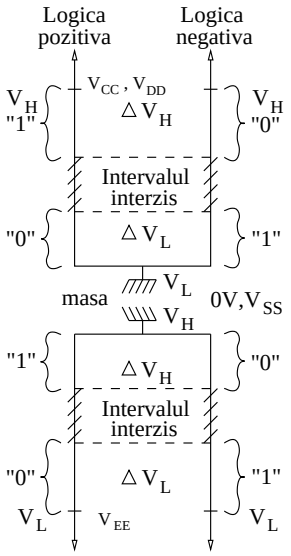


Figura 1.13 Nivelurile de tensiune pentru valorile logice $''1''$ şi $''0''$

adoptată. Prin extinderea de la o valoare fixă de tensiune la un interval se insensibilizează variaţiile produse de: modificarea tensiunii de alimentare, îmbătrânirea pieselor, temperatură, zgomot. Între cele două intervale de tensiune permise ΔV_H, ΔV_L există un interval de tensiune interzis; pentru o poartă cu funcţionare normală valorile tensiunilor de intrare şi de ieşire nu pot să se situeze în intervalul interzis. Detectarea doar a cifrelor logice de 1 sau 0 impune pentru o poartă logică o comportare "procustiană", deci de fiecare dată când semnalul trece printr-o poartă logică este readus la nivelul logic de 0 sau de 1, adică în interiorul intervalelor permise. Deoarece fiecare poartă readuce semnalul în intervalele permise ΔV_H, ΔV_L rezultă că la propagarea semnalului printr-un lanţ de porţi zgomotul suprapus este eliminat în fiecare nivel logic şi nu se amplifică, precum la circuitele cu funcţionare analogică.

Marginea de zgomot în curent continuu şi imunitatea la perturbaţii. O poartă logică are specificate în foaia de catalog **nivelurile de tensiune garantate la ieşire** şi **nivelurile de tensiune permise la intrare**, Figura 1.14-a. Nivelurile de tensiune garantate la ieşire sunt acoperitoare în raport cu nivelurile de tensiune permise la intrare, această acoperire fiind gândită în scopul preîntâmpinării influenţei zgomotelor. (Prin zgomot se înţelege orice semnal electric ce se suprapune peste semnalul logic). Această acoperire se reflectă în parametrii: margine de zgomot M_H în curent continuu pentru starea H şi marginea de zgomot M_L în curent continuu în starea L.

Definiţia 1.13 Marginea de zgomot pentru nivelul H în curent continuu, $\boldsymbol{M_H}$, este diferenţa dintre tensiunea de ieşire minimă garantată în starea H, V_{OHmin}, şi tensiunea de intrare minimă permisă în starea H, V_{IHmin}. **Marginea de zgomot pentru nivelul L în curent continuu**, $\boldsymbol{M_L}$, este diferenţa dintre tensiunea de intrare maximă permisă în starea L, V_{ILmax}, şi tensiunea de ieşire maximă garantată în starea L, V_{OLmax}. $\diamond$

$$\begin{aligned} M_H &= V_{OHmin} - V_{IHmin}; \\ M_L &= V_{ILmax} - V_{OLmax}. \end{aligned} \tag{1.18}$$

Valorile garantate la ieşire şi cele permise la intrarea unei porţi pot fi corelate cu **caracteristica de transfer, VTC** (**V**oltage **T**ransfer **C**haracteristic). Caracteristica de transfer $V_O = f(V_I)$ exprimă grafic dependenţa statică între tensiunea la ieşirea porţii, V_O, şi tensiunea aplicată la intrarea porţii, V_I. Pentru o poartă inversor VTC-ul este reprezentat în Figura 1.14-b. Evident, că această caracteristică de transfer a unei porţi este situată în afara zonelor interzise (reprezentate haşurat)

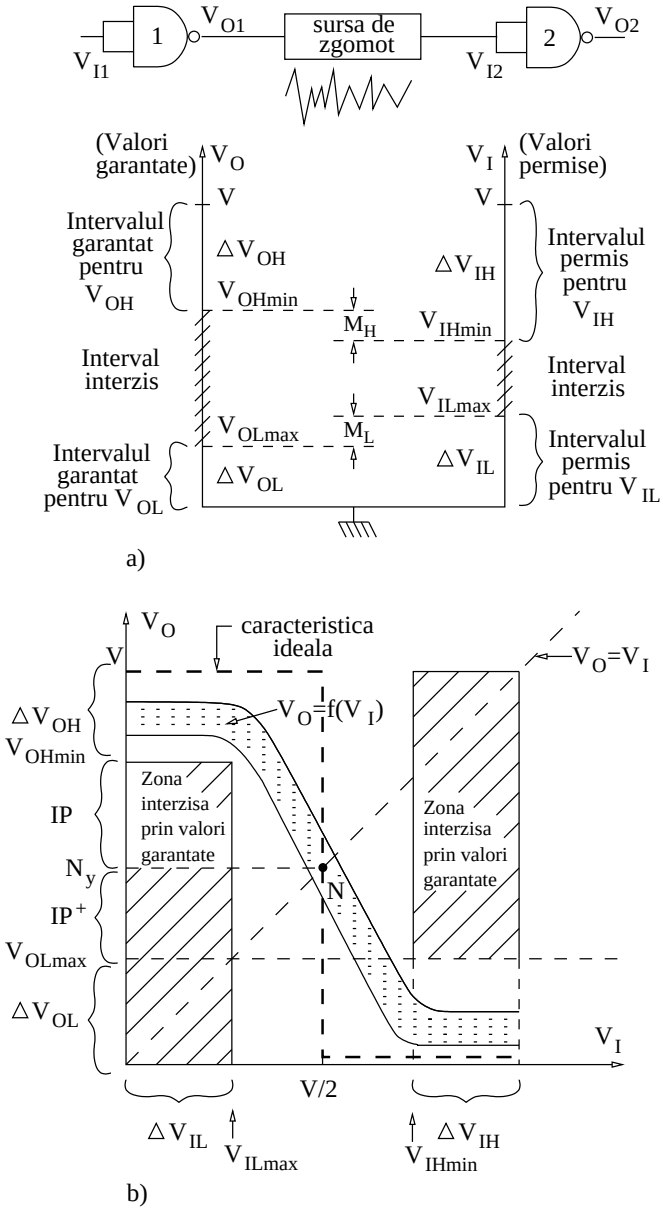


Figura 1.14 Tensiunile de intrare şi ieşire la o poartă logică: a) definirea nivelurilor/(intervalelor) de tensiune H şi L garantate la ieşire şi permise la intrare; b) caracteristica statică de transfer $V_O = f(V_I)$ pentru o poartă inversor.

în planul V_I, V_O şi este desenată ca o bandă (punctată), pentru a indica faptul că porţile de acelaşi tip din cadrul unei familii au caracteristici ce nu se suprapun ci sunt dispersate în această bandă.

Imunitatea la perturbaţii, $\boldsymbol{IP^+}$ **respectiv** $\boldsymbol{IP^-}$, se defineşte prin tensiunile proporţionale cu următoarele segmente din Figura 1.14-b:

$$\begin{aligned} IP^+ &= V_{OLmax} N_y[V] \\ IP^- &= V_{OHmin} N_y[V] \end{aligned} \tag{1.19-a}$$

Se presupune că tensiunea de ieşire a unei porţi este V_{OHmin} şi peste aceasta se suprapune (se scade) o tensiune de zgomot cu amplitudinea cel mult egală cu IP^-; iar tensiunea rezultată aplicată, ca tensiune de intrare la intrarea porţii următoare (comandată), nu produce pentru această poartă deplasarea punctului de funcţionare pe caracteristica de transfer dincolo de punctul N (tensiunea de intrare nu devine mai mică decât $\frac{V}{2}$ care ar corespunde trecerii tensiunii de intrare de la V_{IH} la V_{IL}). La fel se consideră că şi în starea L peste valoarea V_{OLmax} se poate suprapune cel mult tensiunea de zgomot IP^+ fără ca să se depăşească pe caracteristică punctul N. Punctul N este intersecţia caracteristicii $V_O = f(V_I)$ cu prima bisectoare, care are coordonatele $(\frac{V}{2}, \frac{V}{2})$ când caracteristica este simetrică. Dacă amplitudinile de zgomot aplicate tensiunii de ieşire în stările H sau L depăşesc respectiv valorile IP^- sau IP^+ atunci tensiunea de intrare la poarta comandată trece dincolo de coordonatele punctului N şi produce o comutaţie eronată; coordonatele punctului N definesc pragul logic de comutaţie al porţii (vezi Definiţia 1.14).

Pentru aplicaţiile din mediile cu zgomot, de amplitudine mare, se recomandă porţi care au valori mari pentru IP^+ şi IP^-; astfel de porţi denumite cu **imunitate ridicată la zgomot** au tensiuni de alimentare care pot ajunge la 30V. Pentru realizarea unei posibilităţi de comparaţie a imunităţii la perturbaţii a diferitelor familii de porţi logice, care au valori diferite pentru nivelurile logice de tensiune V_H, V_L şi pentru tensiunile de alimentare, se introduc coeficienţii adimensionali – **factorii de imunitate la perturbaţii**:

$$\begin{aligned} FIP^+[\%] &= \frac{IP^+}{\Delta V} \cdot 100 \\ FIP^-[\%] &= \frac{IP^-}{\Delta V} \cdot 100 \end{aligned} \tag{1.19-b}$$

unde ΔV este saltul de tensiune între nivelurile logice "0" şi "1". Dacă se consideră (cazul ideal) că ΔV este egală cu tensiunea de alimentare V_{CC}, V_{DD} şi $IP^+ = \frac{1}{2}V$, $IP^- = \frac{1}{2}V$ rezultă pentru $FIP^+ = FIP^- = 50\%$, în general $FIP < 50\%$. Caracteristica ideală de inversor este cea de tip releu fără histerezis trasată cu linie întreruptă îngroşată în Figura 1.14-b. Porţile în tehnologie CMOS au caracteristica de transfer care se apropie cel mai mult de cea ideală.

Timpul de propagare τ_p**.** Timpul de propagare este un parametru care reflectă viteza de răspuns/comutaţie a unei porţi, altfel spus, este întârzierea în timp între momentul aplicării semnalului logic la intrarea porţii şi momentul apariţiei semnalului la ieşirea porţii. Deoarece, practic, măsurările se fac pe formele de variaţie în timp ale semnalelor de intrare şi ieşire se vor defini unele puncte specifice fixate pe aceste semnale.

În Figura 1.15-a sunt definite următoarele mărimi (pe semnalele de intrare şi de ieşire de la o poartă inversor):

τ_r- timpul de creştere (rise time); intervalul de timp între valorile 10% şi 90% pe frontul de creştere de la L la H al semnalului de intrare în poartă;

τ_f- timpul de descreştere (fall time); intervalul de timp între valorile 10% şi 90% pe frontul de descreştere de la H la L al semnalului de intrare în poartă;

τ_{LH}- durata frontului de creştere; intervalul de timp între valorile 10% şi 90% pe variaţia de la L la H a semnalului de ieşire din poartă;

τ_{HL}- durata frontului de cădere; intervalul de timp între valorile 90% şi 10% pe variaţia de la H la L a semnalului de ieşire din poartă;

τ_{pLH}- timpul de propagare prin poartă la comutarea ieşirii de la L la H; măsurarea se face între punctele cu amplitudine 50% ale variaţiei semnalului de intrare şi de ieşire;

τ_{pHL}- timpul de propagare prin poartă la comutarea ieşirii de la H la L; măsurarea se face între punctele cu amplitudine 50%;

T_{ciclu}- perioada de ciclu este intervalul de timp între două puncte identice pe două cicluri succesive de variaţie ale unui semnal. În practica circuitelor digitale se recomandă să se lucreze cu semnale care îndeplinesc relaţia:

$$T_{ciclu} \geq (15 \div 40)\tau_p$$

(pentru circuitele care funcţionează la frecvenţe de peste 1GHz T_{ciclu} scade sub $10\tau_p$).

Timpul de propagare este definit prin relaţia:

$$\tau_p = \frac{\tau_{pHL} + \tau_{pLH}}{2} \tag{1.20}$$

De multe ori în practică, pentru uşurarea măsurării intervalelor de timp şi fără a de introduce erori semnificative, se consideră pentru semnalul de intrare V_I o variaţie dreptunghiulară, Figura 1.15-b. Pentru acest semnal de intrare ideal, cu panta fronturilor infinită, valorile timpilor de propagare τ_{pHL} şi τ_{pLH} se măsoară de la aceste fronturi până în punctul de amplitudine 50% de pe variaţia semnalului de ieşire.

Timpul de propagare este dependent de structura porţii şi sarcina comandată la ieşire. Dependenţa de sarcina comandată este foarte puternică la porţile CMOS.

Un model simplificat al circuitului de ieşire al unei porţi logice şi a sarcinii comandate este reprezentat în Figura 1.16-a. În acest model rezistenţa echivalentă R include rezistenţa internă a etajului de ieşire al porţii (a generatorului G) plus rezistenţa sarcinii (a porţii sau porţilor comandate), iar capacitatea echivalentă C include capacităţile interne ale etajului de ieşire, capacitatea sarcinii comandate şi capacităţile parazite ale conexiunilor. Conform acestui model rezultă variaţia tensiunii pe capacitatea de sarcină respectiv la incărcare şi descărcare:

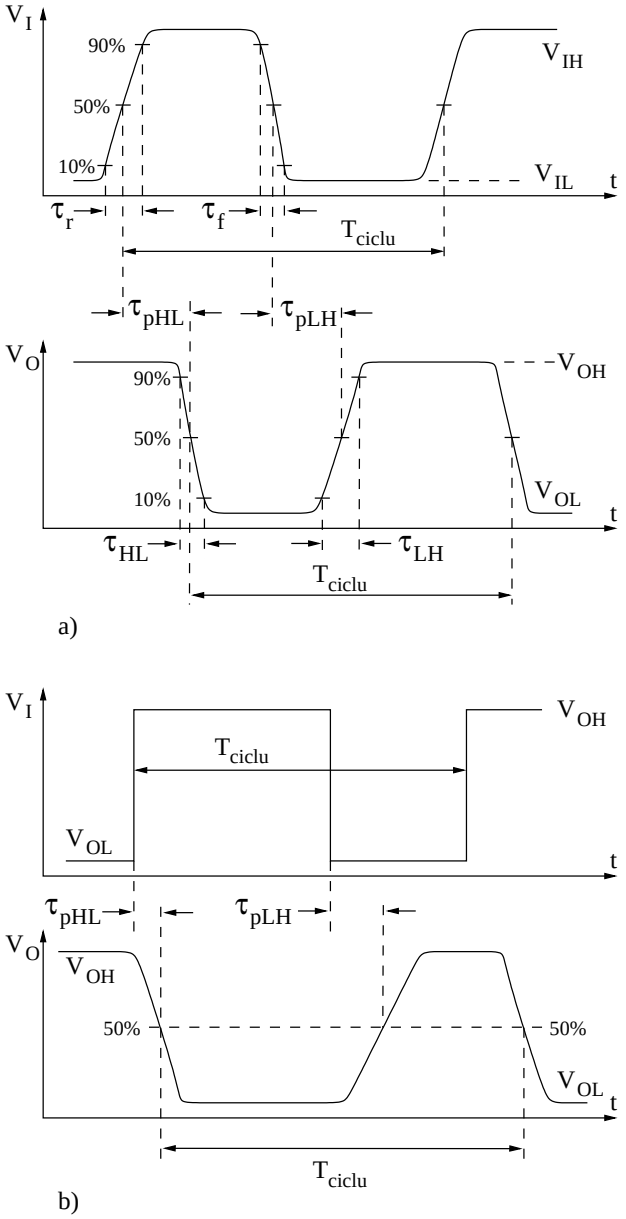


Figura 1.15 Definirea parametrilor de timp pe variaţia semnalelor de la intrarea şi ieşirea unei porţi inversor: a) modul de definire a intervalelor: τ_{r}, τ_{f}, τ_{HL}, τ_{LH}, τ_{pHL}, τ_{pLH} şi T_{ciclu}; b) model simplificat pentru măsurarea de τ_{pHL} şi τ_{pLH}.

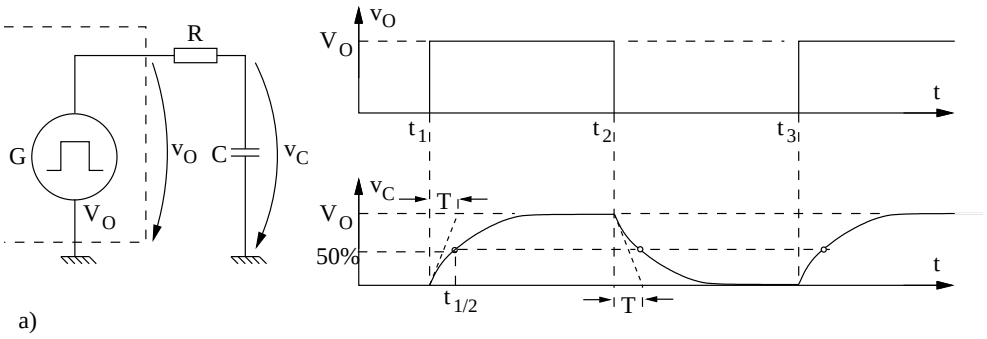


Figura 1.16 Modelul simplificat pentru circuitul de ieşire al unei porţi (a) şi variaţia în timp a tensiunii pe o sarcină capacitivă conectată la ieşire (b).

$$\begin{aligned} \text{a)} \quad v_C &= V_O\left(1 - e^{-\frac{t}{T}}\right) \\ \text{b)} \quad v_C &= V_O \cdot e^{-\frac{t}{T}} \end{aligned} \tag{1.21}$$

unde $T = R \cdot C$ (variaţiile în timp ale tensiunii de ieşire de la 0V la V_O şi de la V_O la 0V sunt simetrice deoarece constantele de timp sunt egale). Timpul $t_{\frac{1}{2}}$, la valoarea tensiunii de ieşire 50% V_O, calculat cu relaţiile 1.21, conform modelului simplificat din Figura 1.15-b, este egal cu τ_{pHL}, τ_{pLH}.

$$t_{\frac{1}{2}} = \tau_{\text{pHL}} = \tau_{\text{pLH}} = RCln2 = 0.69\,RC$$

Variaţia simetrică a tensiunii v_c determină valori egale şi pentru τ_{r} şi τ_{f}.

$$\tau_{\text{r}} = \tau_{\text{f}} = T = RC\,(ln10 - ln\frac{10}{9}) = 2.2\,RC$$

Exemplul 1.8 Rezistenţa de ieşire a unei porţi în starea H este $R_1 = 2K\Omega$ iar în starea L este $R_2 = 25\Omega$. Sarcina capacitivă pe ieşire are valoarea de $100pF$. Întârzierea internă τ_{pi} este de $25ns$. Să se determine timpul de propagare τ_{p} al porţii.

Soluţie:

$$\begin{aligned} \tau_{\text{pLH}} &= \tau_{\text{pi}} + R_1 C \ln 2 = 25ns + (2 \cdot 10^3\Omega)(100 \cdot 10^{-12}) \cdot ln2 \approx 160ns \\ \tau_{\text{pHL}} &= \tau_{\text{pi}} + R_2 C \ln 2 = 25ns + (25\Omega)(100 \cdot 10^{-12}) \cdot ln2 \approx 27ns \\ \tau_{\text{p}} &= \frac{\tau_{\text{pHL}} + \tau_{\text{pHL}}}{2} = \frac{27 + 160}{2} = 93.5ns \end{aligned}$$

Factorii de încărcare la intrare şi la ieşire. Factorul de încărcare la intrare (**fan in**, input loading factor) reprezintă sarcina pe care o intrare o introduce când este conectată la ieşirea unei porţi. Deoarece într-o familie de circuite logice există diferite porţi cu număr diferit de intrări trebuie să se fixeze care tip de intrare reprezintă sarcina standard. În general, se admite că sarcină este standard sarcina care corespunde unei intrări de la o poartă NAND cu două intrări (NAND2).

În funcţie de tehnologia de realizare a porţilor sarcina de intrare se măsoară în mărimi fizice diferite. Pentru tehnologia bipolară, unde comanda se face printr-un curent pe baza unui tranzistor, sarcina de intrare se măsoară în unităţi de curent. Iar pentru tehnologia CMOS, unde comanda se face în tensiune pe o capacitate (echivalentă) de intrare, sarcina de intrare se măsoară în unităţi de capacitate.

În Figura 1.17 sunt prezentate notaţiile şi sensurile curenţior de la intrarea şi ieşirea porţilor. Referitor la o bornă, care este un terminal al unei porţi, unui curent care intră prin acea bornă i se asociază semnul plus (+I), iar unui curent care iese din acea bornă i se ataşează semnul minus (-I); evident, simbolurile celor doi curenţi care intră şi ies, la o aceeaşi bornă, au semne opuse. În foaia de catalog a unei porţi este specificată sarcina maximă (curent sau capacitate) pe care intrarea unei porţi o prezintă pentru nivelurile permise de tensiune H şi L. O poartă trebuie să poată absorbi/genera la ieşire un curent mai mare sau egal cu suma tuturor curenţilor necesari generaţi/absorbiţi de către toate intrările porţilor care sunt conectate la acea ieşire, dar în acelaşi timp poarta trebuie să asigure la ieşire şi nivelurile garantate de tensiune pentru starea H şi L. Cu aceste valori necesare pe o intrare şi disponibile pe o ieşire se poate determina un factor de încărcare la ieşire (**fan-out**, output loading

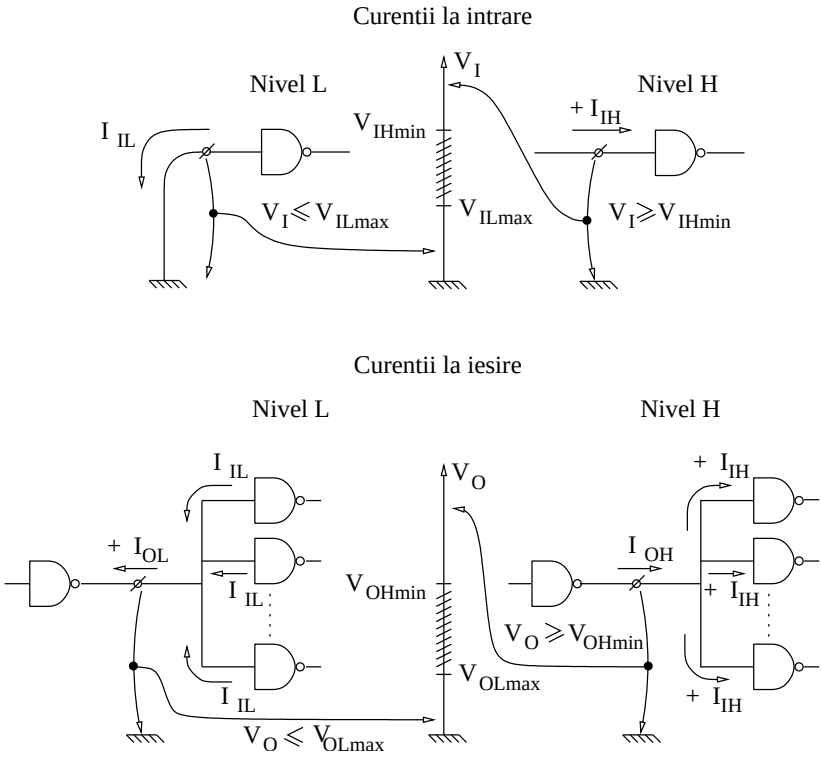


Figura 1.17 Explicativă pentru simbolurile şi sensurile curenţilor la intrarea (a) şi ieşirea (b) unei porţi

factor) atât în starea L, FI_L cât şi în starea H, FI_H conform relaţiilor:

$$\begin{aligned} FI_L &= \frac{I_{OLmax}}{I_{ILmax}}; \\ FI_H &= \frac{I_{OHmax}}{I_{IHmax}}; \\ FI &= min\{FI_L, FI_H\} \end{aligned} \tag{1.22}$$

Rezultă că factorul de încărcare maxim FI este egal cu valoarea minimă dintre FI_L, şi FI_H. În general, dacă se consideră că toate intrările porţilor au aceeaşi sarcină atunci factorul de încărcare maxim exprimă numărul maxim de intrări comandate de ieşirea unei porţi fără deteriorarea nivelurilor normale de tensiune. Acelaşi raţionament, ca şi pentru curenţi, se poate face când sarcina este capacitivă (se determină numărul maxim de sarcini standard/unitare pe care le poate comanda o poartă pe ieşire).

Pentru o poartă, ideal, FI ar trebui să fie infinit dar în realitate are valori de ordinul unităţilor pentru tehnologia bipolară şi de ordinul zecilor pentru tehnologia CMOS. Prin **"tăria"** unui semnal se înţelege abilitatea de a absorbi sau genera un curent. Cu cât un semnal este mai puternic cu atât curentul generat sau absorbit este mai mare (prin convenţie bara de alimentare V_{DD}/V_{CC} generează un curent iar bara de masă 0V/V_{SS} absoarbe un curent). Pentru porţile logice ieşirile sunt surse de nivelurile 1 sau 0 mai puternice decât intrările. Liniile de alimentare V_{DD}, V_{CC} sau V_{SS} sunt sursele cele mai puternice de 1 logic şi 0 logic. Un semnal poate fi "întărit", adică să poată comanda mai multe intrări, prin intermediul unui buffer/driver. Un buffer este un circuit care spre deosebire de poartă nu procesează logic semnalul, eventual îl inversează - buffer inversor, dar prezintă la ieşire un FI mult mărit. Ca funcţie logică bufferul neinversor este funcţia identitate f_1^1, iar bufferul inversor este funcţia f_2^1, vezi secţiunea 1.1.3 . Circuitele buffer, în general, sunt realizate în grup de câte opt pe cip pentru a putea întări un cuvânt de 8 biţi (1 byte). Unele porţi au pe ieşire un etaj circuit buffer, caz în care sunt referite ca **porţi "bufferate"** .

Exemplul 1.9 Pentru seriile de porţi logice din familia TTL să se determine FI.

Soluţie. Curenţii de ieşire şi intrare în starea H şi L, extraşi din catalog sunt prezentaţi în *Tabelul 1.8*.

Tabelul 1.8 Valorile curenţilor pentru seriile de porţi logice din familia TTL

Seria	INTRARE		IEŞIRE	
	nivel H	nivel L	nivel H	nivel L
	$I_{IHmax}, \mu A$	I_{ILmax}, mA	I_{OHmax}, mA	I_{OLmax}, mA
74	40	-1.6	-0.4 / -0.8^*	16
74S	50	-2.0	-1.0	20
74LS	20	-0.36	-0.4	8
74AS	20	-2,0	-0.4	$4/8^*$
74ALS	20	-0.1	-0.4	8

*depinde de circuit

Pentru seria 74 (o poartă din seria 74 comandă porţi tot din seria 74):

$$FI_L = \frac{I_{OLmax}}{I_{ILmax}} = \frac{16mA}{1.6mA} = 10;$$
$$FI_H = \frac{I_{OHmax}}{I_{IHmax}} = \frac{800\mu A}{40\mu A} = 20;$$
$$FI = min\{20, 10\} = 10$$

Pentru seria 74LS (o poartă din seria 74LS comandă porţi tot din seria 74LS):

$$FI_L = \frac{I_{OLmax}}{I_{ILmax}} = \frac{8mA}{0.36mA} = 22;$$
$$FI_H = \frac{I_{OHmax}}{I_{IHmax}} = \frac{400\mu A}{20\mu A} = 20;$$
$$FI = min\{20, 20\} = 20$$

În acelaşi mod se pot calcula factorii de încărcare pentru seriile: 74S, 74AL şi 74ALS. De asemenea se pot calcula FI pentru cazurile când o poartă dintr-o serie comandă porţi din altă serie.

Exemplul 1.10 O poartă logică CMOS din familia 74HC cu o rezistenţă de ieşire $R = 300\Omega$, generează o tensiune de ieşire $V_O = 4.5V$ şi are o întârziere internă $\tau_{\text{pi}} = 5ns$. Să se determine numărul de porţi de acelaşi tip care pot fi comandate de această poartă astfel încât timpul de propagare τ_{p} să nu fie mai mare de 40ns.

Soluţie. Pentru porţile CMOS sarcina pe intrare este capacitivă (o valoare aproape standard a capacităţii de intrare pentru o poartă CMOS, realizată discret, este $C_{in} = 5pF$), curentul de intrare este neglijabil. De asemenea, se consideră că procesul de încărcare şi descărcare al capacităţii este simetric ca în Figura 1.16-b deci $\tau_{\text{pHL}} = \tau_{\text{pLH}} = t_{\frac{1}{2}}$. Rezultă:

$$\tau_{\text{p}} = \tau_{\text{pi}} + \frac{\tau_{\text{pHL}} + \tau_{\text{pLH}}}{2} = 40ns \qquad \Rightarrow \qquad t_{\frac{1}{2}} = 40ns - 5ns = 35ns$$

Pentru $\tau_{\text{p}} = 35ns$ sarcina comandată de poartă poate fi egală cu

$$t_{\frac{1}{2}} = 35ns = 0.69\,RC \Rightarrow C = 170pF$$

Numărul de porţi comandate este $\frac{170pF}{5pF} = 34$.

Se observă că mărimea sarcinii comandate de o poartă CMOS determină valoarea timpului de propagare τ_{p}. Creşterea timpului de propagare cu creşterea sarcinii comandate este mult mai pregnantă la porţile CMOS decât la cele bipolare.

Consumul de putere. Consumul de putere P_d (puterea disipată) pentru porţile unei familii rezultă ca o sumă între două componente una statică P_{dcc} şi una dinamică P_{dca}, $P_d = P_{dcc} + P_{dca}$.

1. Consumul de putere în regim static (în c.c.) se defineşte cu relaţia:

$$P_{dcc} = \frac{P_H + P_L}{2} = \left(\frac{I_{CCH} + I_{CCL}}{2}\right) V_{CC} \tag{1.23}$$

în care: I_{CCH}, I_{CCL}- sunt curenţii absorbiţi de poartă de la sursă în starea H, respectiv L; V_{CC} - tensiunea de alimentare a porţii.

De remarcat, că P_{dcc} este componenta principală de putere disipată la familia TTL (tipic, 1mW/poartă pentru seria 74ALS şi în jur de 8.5mW/poartă pentru seria AS) pe când la tehnologia CMOS această componentă este neglijabilă (tipic 2.5nW/poartă pentru seria 74HC).

2. Consumul de putere în regim dinamic P_{dca} (în c.a.). Această componentă apare numai pe durata intervalelor de comutaţie între cele două niveluri logice şi se evidenţiază sub două forme:

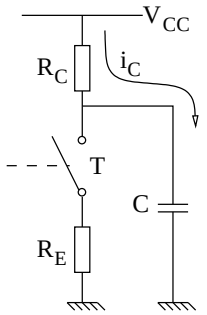


Figura 1.18 Circuit echivalent (simplficat) al unei porţi pentru calculul puterii P_{dca}

(a) În prima formă, puterea disipată apare prin încărcarea şi descărcarea capacităţilor din circuit care, uneori pentru simplificare, se substituie cu o singură capacitate echivalentă C = capacitate internă din circuit + capacitatea conexiunilor + capacităţile de sarcină. Un circuit echivalent pentru calculul acestei puteri este prezentat în Figura 1.18. Când contactul T (tranzistor) este comandat spre deschidere, ieşirea comută din L în H, condensatorul C se încarcă prin rezistenţa R_C stocând o cantitate de energie $\frac{1}{2}CV_{CC}^2$. Pe durata procesului de încărcare, când variaţia curentului este exprimată prin relaţia $i_c = \frac{V_{CC}}{R_C} \cdot e^{-\frac{t}{T}}$, energia disipată pe rezistenţa R_C se calculează în felul următor:

$$W = \int_0^\infty i_c^2 \cdot R_C \mathrm{d}t = \frac{V_{CC}^2}{R_C} \int_0^\infty e^{-\frac{2t}{C \cdot R_C}} \mathrm{d}t =$$
$$= \frac{V_{CC}^2}{R_C}\Big|_0^\infty - \frac{R_C \cdot C}{2} \cdot e^{-\frac{t}{C \cdot R_C}}\Big|_0^\infty = \frac{1}{2}CV_{CC}^2$$

rezultă că energia nu depinde de valoarea rezistenţei R_C. La comanda spre închidere a contactului T, când ieşirea comută din H în L, energia $\frac{1}{2}CV_{CC}^2$ stocată pe condensatorul C este disipată pe rezistenţa echivalentă $R_C \parallel R_E$. Pe durata unei perioade $T = \frac{1}{f}$, a semnalului de comandă, energia consumată pe rezistenţele R_C şi R_E este $2\left(\frac{1}{2}CV_{CC}^2\right) = CV_{CC}^2$. Rezultă că puterea disipată (energia în unitatea de timp) pentru încărcarea şi descărcarea capacităţii echivalente este exprimată de relaţia:

$$P_{dca} = CV_{CC}^2 \cdot f \tag{1.24}$$

(b) A doua formă pentru puterea disipată în regim dinamic apare datorită scurtcircuitării sursei de alimentare la masă pe durata fronturilor de comutaţie. Pe durata acestor fronturi, când unele tranzistoare din structura porţii comută din blocat în conducţie iar altele din conducţie în blocare, există un interval scurt când toate tranzistoarele conduc (vezi Figura 1.22-e) ceea ce duce la scurtcircuitarea sursei la masă (pe unele trasee putând exista anumite rezistenţe), producând vârfuri de curent (spike) în linia (V_{CC}) de alimentare a porţii. Apariţia acestor vârfuri de curent pe linia de alimentare pot să provoace comutaţii false la alte porţi care se alimentează de la aceeaşi linie. Puterea disipată de scurtcircuit pe poartă depinde de valoarea de vârf a curentului de scurtcircuit, care este funcţie de tensiunea de alimentare V_{CC}, şi depinde de frecvenţa f de apariţie a comutaţiilor. Se poate exprima puterea disipată de scurtcircuit printr-o relaţie tot de aceeaşi formă ca în 1.24, în care se introduce o capacitate echivalentă de calcul a cărei valoare nu depăşeşte 20% din valoarea capacităţii echivalente utilizată pentru puterea disipată la încărcarea şi descărcarea capacităţilor. Practic, luarea în calcul şi a puterii disipate de scurtcircuit se face prin mărirea valorii capacităţii din relaţia 1.24. Puterea disipată de scurtcircuit este cu atât mai mică cu cât semnalele de comandă au fronturi mai abrupte, adică τ_{f} şi τ_{r} au valori cât mai mici. O regulă practică spune că: durata fronturilor trebuie să fie cel mult a zecea parte din timpul de propagare ($\tau_{\mathrm{r}}, \tau_{\mathrm{f}} < 10\tau_{\mathrm{p}}$), pentru ca puterea disipată pe poartă să nu o distrugă prin creşterea de temperatură.

Pentru circuitele CMOS principala componentă de putere disipată este cea dinamică, relaţia 1.24. Reducerea acesteia se poate realiza prin micşorarea factorilor din această relaţie în special a tensiunii de alimentare care introduce o dependenţă pătratică. Şi la porţile în tehnologie bipolară există componenta dinamică $CV_{CC}^2 \cdot f$, dar aceasta la frecvenţe joase şi medii este neglijabilă faţă de cea disipată în regim static P_{dcc}, relaţia 1.23.

Exemplul 1.11 Valoarea tipică a puterii disipate în c.c. pe o poartă CMOS din seria HC este de $2,5nW$ (la $V_{DD} = 5V$ şi $25°C$) iar în cazul cel mai defavorabil poate ajunge la $30\mu W$. Dacă această poartă este comandată la frecvenţa de $100KHz$ şi la ieşire sunt conectate zece porţi de acelaşi tip, $FI = 10$, să se determine cât va fi puterea disipată pe poartă.

Soluţie. Pentru porţile din seria HC în catalog se specifică sarcina pentru o intrare ca fiind $C_{intr} = 5pF$ iar capacităţile interne totale se dau ca fiind $22pF$. Se obţine capacitatea echivalentă totală care trebuie să fie încărcată şi descărcată la ieşirea porţii

$$C = 22pF + 10 \times 5pF = 72pF$$

Rezultă puterea disipată în regim dinamic :

$$P_{dac} = CV_{DD}^2 \cdot f = 72 \cdot 10^{-12} \times 5^2 \times 10^5 = 180\mu W$$

Această valoare a puterii disipate este de 10^5 ori mai mare decât valoarea disipată tipică în regim de c.c. ($2,5nW$) şi de 6 ori mai mare decât în cazul cel mai defavorabil ($30\mu W$).

Factorul de merit. Factorul de merit notat **PDP** (**P**ower **D**elay **P**roduct) este un parametru sintetic, în sensul că poate caracteriza poarta atât din punct de vedere al puterii disipate cât şi din din punct de vedere al timpului de propagare şi este definit prin produsul dintre puterea disipată şi timpul de propagare:

$$PDP[Joule] = P_d[Watt] \times \tau_{\text{p}}[s] \tag{1.25}$$

Acest parametru poate fi interpretat ca fiind energia consumată pe decizie logică (pe comutaţie). Există porţi logice care au un factor de merit de ordinul pJ sau chiar mai mic, situându-se sub valoarea factorului de merit corespunzător unui neuron!

Dependenţa între parametrii unei familii de circuite logice. Poarta logică ideală ar trebui să prezinte simultan valori optime pentru toţi parametrii: FI foarte mare, $FIP = 50\%$, $\tau_{\text{p}} = 0$, $P_d = 0$, $V_H = V_{CC}$ sau V_{DD}, $V_L = 0V$, $\Delta V = V_{CC}$ sau V_{DD}. Pentru o astfel de poartă inversor VTC-ul din Figura 1.14-b ar fi o caracteristică de tip releu (fără histerezis) cu panta infinită la tensiunea de intrare $\frac{V}{2}$. Cel mai mult se apropie de o caracteristică ideală porţile din tehnologia CMOS. În practică optimizarea simultană a tuturor parametrilor este contradictorie.

Mărirea excesivă a lui FI atrage un curent absorbit de valoare mai mare de la sursă determinând deci mărirea lui P_d. Iar dacă sarcinile comandate sunt capacitive, o mărime a acestora poate duce la creşterea lui τ_{HL} şi τ_{LH} în semnalul de ieşire.

Micşorarea lui τ_{p} implică forţarea regimului de comutaţie prin mărirea tensiunii V_{CC} sau V_{DD} sau prin micşorarea rezistenţei circuitului, ceea ce duce la curenţi mari absorbiţi de la sursă şi în consecinţă o creştere a lui P_d. Prin opoziţie, micşorarea lui P_d prin scăderea curenţilor, ar duce la mărirea lui τ_{p}.

Creşterea factorului de imunitate la perturbaţii ar necesita un salt logic de tensiune cu valoare mărită ceea ce este posibil prin mărirea tensiunii de alimentare, deci, implicit puterea disipată P_d creşte foarte mult. Eventual, se poate menţine puterea disipată în limite nepericuloase, pentru a nu se distruge circuitul prin încălzire, dacă simultan cu mărirea tensiunii de alimentare s-ar micşora curentul prin mărirea rezistenţelor dar aceasta ar duce la timpi de tranziţie foarte lungi deci creşterea lui τ_{p}.

Tendinţa de optimizare simultană a tuturor parametrilor între anumite limite rezonabile pentru aplicaţii eficiente a dus în cadrul unei familii de circuite logice la crearea unei serii standard; parametrii standard au valori cvasioptimale. În paralel cu seria standard, într-o familie de circuite logice, există serii speciale care optimizează doar un singur parametru, ceilalţi parametrii fiind menţinuţi în limite acceptabile. Astfel, există: seria de viteză ridicată, seria de putere redusă, seria de viteză ridicată şi putere redusă, seria cu imunitate sporită la perturbaţii. Toate aceste serii ale unei familii de circuite logice putând fi în variante pentru aplicaţii civile cu gama de tempratură admisibilă $0°C \div 70°C$ cât şi în variantă militară cu gama de temperatură admisibilă $-55°C \div 125°C$.

Porţile logice pot exista fie ca entităţi separate sub forma unor circuite integrate discrete, care sunt utilizate în realizarea unor sisteme mai complexe, fie pot exista în interiorul unui sistem complex realizat în totalitate integrat. Pentru porţile logice realizate discret, care se interconectează în exterior cu alte porţi, respectarea parametrilor este foarte strictă. În schimb, pentru porţile logice dintr-un sistem integrat unde interconexiunile între porţi se fac în siliciu, valorile parametrilor pot fi mai puţin restrictive.

1.4 PORŢI LOGICE ÎN TEHNOLOGIA BIPOLARĂ

Elementele componente de circuit, pe care se bazează tehnologia bipolară, sunt joncţiunea semiconductoare (dioda) şi tranzistoarele bipolare npn şi pnp. Tehnologia de integrare bipolară este caracterizată prin viteză ridicată, densitate de integrare relativ redusă şi consum de putere ridicat. Cu aceste caracterisitici tehnologia bipolară, de la introducerea ei, la începutul anilor 1960, a fost tehnologia cea mai uzuală până la începutul anilor 1990. Începând cu mijlocul anilor 1980 au fost realizate progrese în tehnologia MOS şi mai ales în tehnologia CMOS ceea ce a dus ca în anii 1990 tehnologia bipolară să fie substituită cu cea CMOS. Deşi, acum implementările în tehnologia bipolară sunt reduse, totuşi, minime cunoştinţe despre această tehnologie sunt necesare pentru aplicaţii ocazionale cum ar fi operaţii de depanare a unor sisteme existente şi pentru aspecte de interfaţare TTL/CMOS. Pentru însuşirea acestor minime cunoştinţe se vor prezenta în continuare următoarele trei subiecte: inversorul bipolar, porţi TTL şi porţi pentru magistrale.

1.4.1 Inversorul bipolar

Circuitul inversor este elementul de bază în structura oricărui circuit logic dintr-o familie de porţi logice. Structura de inversor se poate identifica în structura oricărei porţi şi oricare poartă logică se obţine dintr-un circuit inversor prin completare. Se vor studia două aspecte ale circuitului inversor: caracteristica statică de transfer şi timpul de comutaţie.

Structura şi caracteristica de transfer. Circuitul inversor, Figura 1.19-b, este de fapt un etaj amplificator cu saturaţie, cu sarcina în colector, în conexiunea emitor comun. Layout-ul pentru implementarea în siliciu a inversorului este prezentat în Figura 1.19-c. Tranzistorul npn este realizat, prin difuzie, într-o insulă de tip n în stratul epitaxial. La fel şi rezistenţele de bază R_B şi de sarcină R_C sunt realizate prin difuzie în insule de tip n din stratul epitaxial.

Caracteristica statică de transfer $V_O = f(V_I)$ (fără încărcare la ieşire, în gol) este trasată prin linii drepte având ca puncte fixe tensiunea de alimentare V_{CC} şi cele două puncte de frângere $PF1$ şi $PF2$, Figura 1.20. Punctele de frângere $PF1$ şi $PF2$ se situează tocmai la limita de trecere a punctului de funcţionare al tranzistorului de la regimul blocat la regimul activ (PF1), respectiv de la activ la regimul de saturaţie (PF2).

În intervalul de variaţie a tensiunii de intrare $V_I = [0, V_{IL})$ tranzistorul este blocat, tensiunea de ieşire este constantă $V_O = V_{OH} = V_{CC}$, iar caracteristica $V_O = f(V_I)$ este o dreaptă orizontală.

În punctul de frângere $PF1$ tensiunea de intrare devine egală cu $V_{BE(on)}$, tranzistorul intră în conducţie, iar caracteristica de transfer se aproximează în continuare cu o dreaptă ce uneşte punctele $PF1$ (V_{IL}, V_{CC}), $PF2$ $\left(V_{IH}, V_{CE(sat)}\right)$.

În punctul de frângere $PF2$ este începutul regimului de saturaţie pentru care tensiunea de intrare are valoarea:

$$V_I = V_{IH} = V_{BE(sat)} + \frac{1}{\beta} \times \frac{V_{CC} - V_{CE(sat)}}{R_C} \times R_B$$

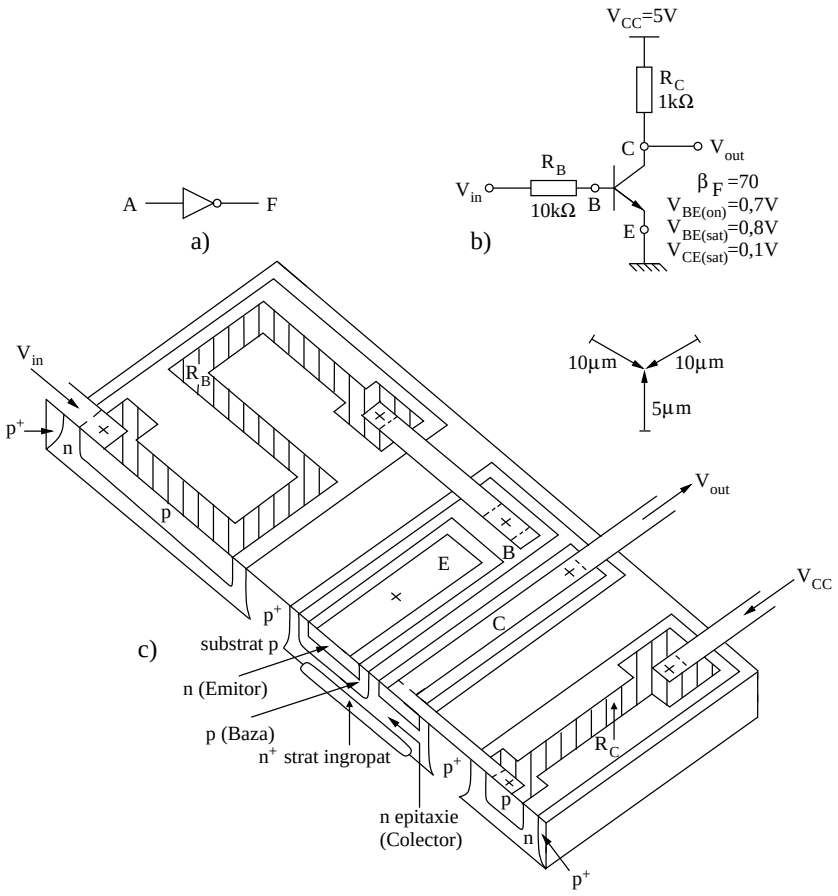


Figura 1.19 Circuitul inversor: a) simbol; b) structura circuitului; c) layout-ul în siliciu (substrat de tip p).

O creştere în continuare a tensiunii de intrare V_I peste V_{IH} produce o micşorare nesemnificativă a tensiunii de ieşire, caracteristica de transfer este o dreaptă orizontală $V_{OL} = V_{CE(sat)}$.

Panta caracteristicii $V_O = f(V_I)$ prezintă numai două valori. Valoarea zero când inversorul este în starea H sau L şi o valoare ridicată $A_V = \frac{\Delta V_O}{\Delta V_I}$ în zona de trecere între cele două stări. În reprezentarea idealizată din figură punctele de frângere apar la intersecţia acestor drepte, dar într-o reprezentare exactă aceste puncte s-ar afla pe o curbă, care ar racorda aceste drepte, unde panta ar fi unitară $|A_V| = 1$. Această caracteristică exactă se plasează în interiorul benzii punctate din Figura 1.14-b.

Punctul de funcţionare al inversorului trebuie să fie doar în cele două zone: blocat sau saturat. Trecerea între cele două zone când tranzistorul este în regimul activ, specific funcţionării în circuitele analogice, trebuie să se realizeze într-un timp cât mai scurt; de fapt, această trecere ar corespunde, pe Figura 1.13, intervalului interzis de tensiuni.

Timpii de comutaţie ai inversorului. Timpii de comutaţie ai tranzistorului

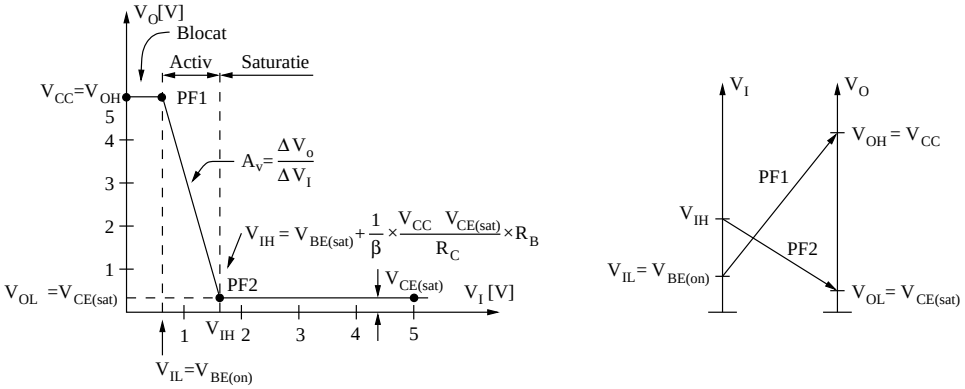


Figura 1.20 Caracteristica de transfer, $V_O = f(V_I)$ pentru un inversor bipolar la funcţionarea în gol

din blocat în saturaţie şi din saturaţie în blocat nu sunt egali, ceea ce determină valori diferite pentru τ_{pHL}, τ_{pLH} ($\tau_{\text{pLH}} > \tau_{\text{pHL}}$). În raport cu timpul de comutaţie directă, din regimul blocat înspre saturaţie, timpul de comutaţie inversă, din regimul de saturaţie înspre blocare, este mai mare. Această diferenţă apare datorită procesului de eliminare a sarcinii stocate în surplus în bază, când tranzistorul este în saturaţie, faţă de sarcina din bază când tranzistorul este în regim activ direct. În regim de saturaţie se injectează în bază purtători majoritari atât din emitor cât şi, în surplus, din colector deoarece ambele joncţiuni sunt polarizate în sens direct. Eliminarea sarcinii în surplus din bază necesită un timp τ_{s} (timp de saturaţie). Deci timpul de comutaţie inversă, la un tranzistor în saturaţie, faţă de timpul de comutaţie inversă al unui tranzistor în regim activ direct este mai mare cu valoarea τ_{s}, vezi Figura 1.21-a. Figura reprezintă variaţia tensiunii de comandă v_I a inversorului (o variaţie dreptunghiulară) şi variaţia tensiunii v_O la ieşirea inversorului. În această figură timpii respectivi au următoarele semnificaţii:

- $\tau_{\text{d}} = t_1 - t_0$, timpul de întârziere. Este timpul necesar pentru creşterea tensiunii aplicate pe joncţiunea emitoare şi colectoare. Deoarece aceste joncţiuni sunt zone sărăcite de purtători acestea se comportă ca nişte condensatoare, deci este timpul de încărcare al acestor condensatoare;

- $\tau_{\text{f}} = t_2 - t_1$, timpul de coborâre (fall time). Este determinat, mai ales, de timpul de tranzit al purtătorilor prin bază;

- $\tau_{\text{s}} = t_4 - t_3$, timpul de saturaţie. Timpul necesar pentru eliminarea sarcinii în surplus stocată în bază;

- $\tau_{\text{r}} = t_5 - t_4$, timpul de creştere. Similar cu τ_{f} dar acum trebuie anulaţi purtătorii din bază pentru că tranzistorul se comandă spre blocare.

Micşorarea lui τ_{pLH} se poate realiza în două modalităţi. Prima, constă în forţarea comenzii pe baza tranzistorului prin suprapunerea unei componente derivative de curent peste curentul de bază de comandă normal. Condensatorul C conectat în paralel cu rezistenţa de bază R_B, Figura 1.21-b, formează cu rezistenţa echivalentă

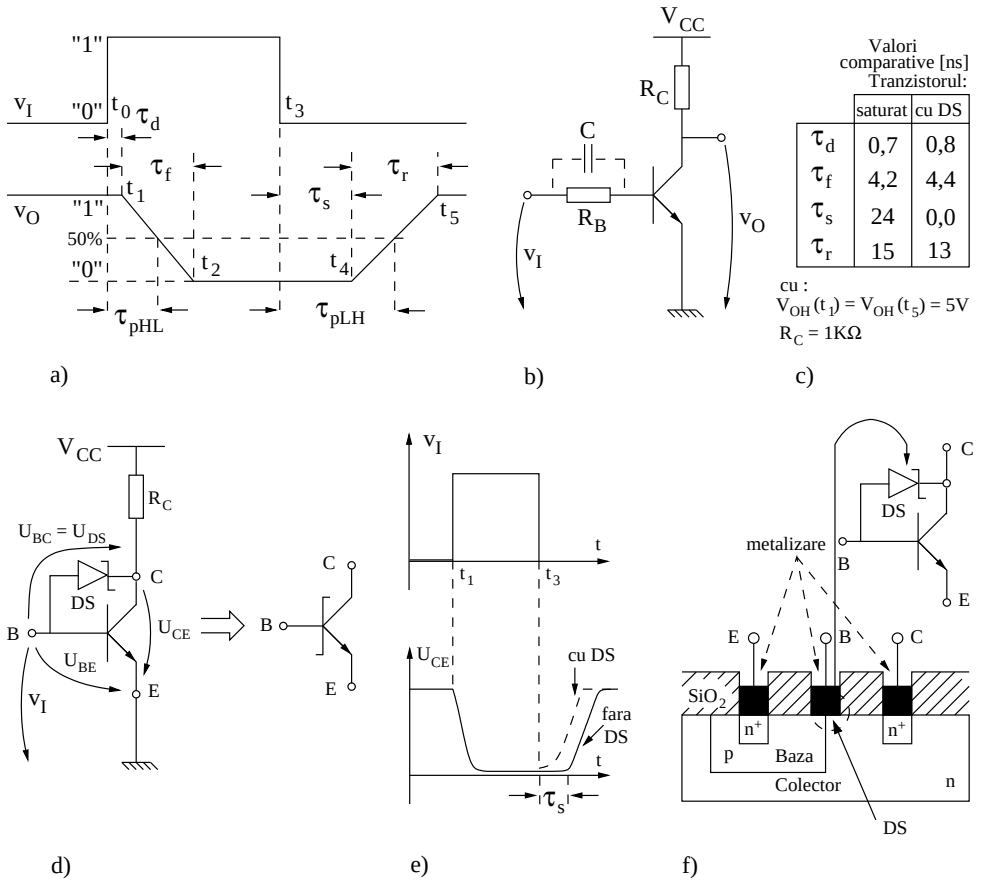


Figura 1.21 Inversorul bipolar: a) definirea timpilor care caracterizează procesul de comutaţie; b) inversor cu forţarea comenzii prin producerea unei componente derivative în curentul de bază; c) valori pentru parametrii de timp la un tranzistor normal şi la un tranzistor Schottky; d,e,f) realizarea unei structuri de tranzistor Schottky.

bază emitor R_{BE} a tranzistorului un circuit derivativ, deci generează componenta de forţare (derivativă) în curentul de comandă din bază.

A doua modalitate se realizează prin utilizarea unui tranzistor Schottky, care se obţine prin conectarea între bază şi colectorul unui tranzistor normal a unei joncţiuni metal-semiconductor, adică o diodă Schottky, DS, Figura 1.21-d. Implementarea unei astfel de diode în acel loc se realizează, în procesul de fabricare al tranzistorului, extrem de simplu şi practic fără cost, prin extinderea metalizării (Al – aluminiu sau Pt – platină) de contact a terminalului bazei B şi peste zona semiconductoare de tip n a colectorului. Extinderea metalizării, zona încercuită din Figura 1.21-f, formează la interfaţa metal-colector o diodă Schottky Aln sau Ptn. Tensiunea la polarizare directă a unei diode Schottky este $V_{DS} = 0,4V$. Conectând o astfel de diodă între bază şi colector (deci în paralel cu joncţiunea colectoare) tensiunea pe joncţiunea colectoare nu va putea coborî cu mai mult de $0,4V$ sub tensiunea bazei tranzistorului, rezultă că tranzistorul nu mai poate intra în saturaţie, nu se mai injectează din colector sarcină în surplus (faţă de regimul nesaturat) în bază, deci τ_s devine zero. Cu valorile tensiunilor din Figura 1.19-b se pot face următoarele calcule:

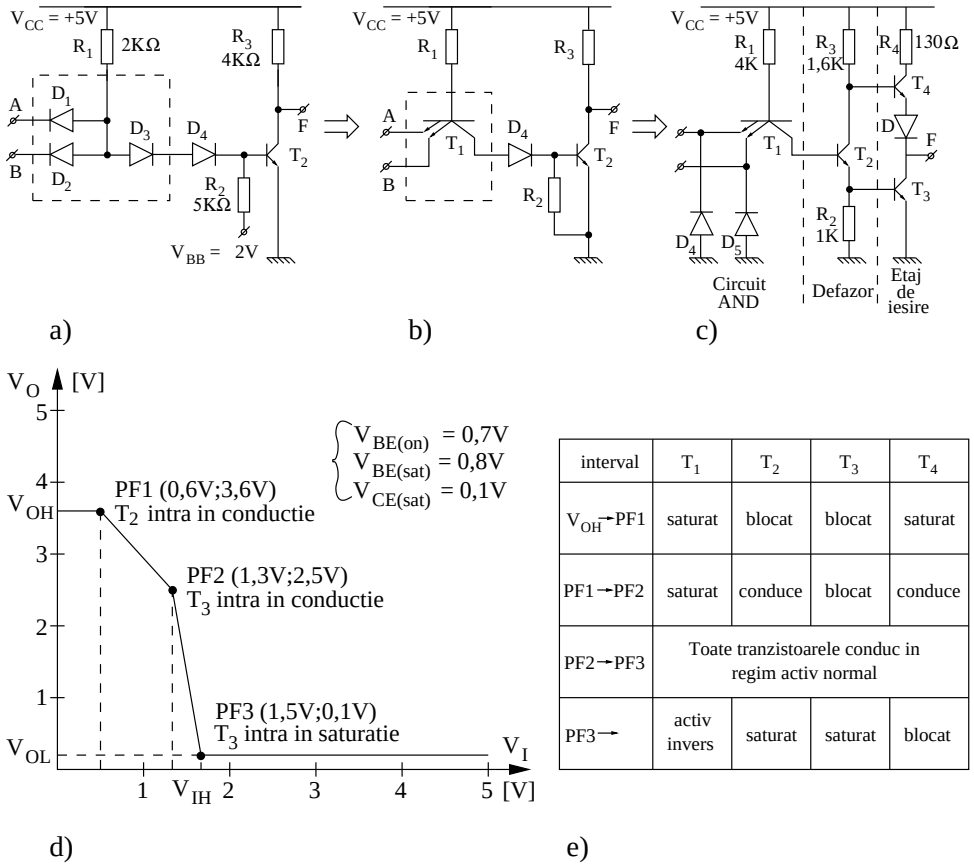
- fără diodă Schottky: $V_{BC} = V_{BE(sat)} - V_{CE(sat)} = 0,8V - 0,1V = 0,7V$, tensiune ce comandă joncţiunea bază colector în conducţie, tranzistorul este în saturaţie;

- cu diodă Schottky $V_{CE} = V_{BE(sat)} - V_{DS} = 0,8V - 0,4V = 0,4V$ tranzistorul nu este în saturaţie (tensiunea de deschidere a unei joncţiuni este de $\sim 0,7V$).

Valori pentru timpii de comutaţie, definiţi mai sus, sunt prezentate în tabelul din Figura 1.21-c, atât pentru un tranzistor normal cât şi pentru un tranzistor Schottky. În seriile de porţi logice din tehnologia bipolară, cu excepţia seriei standard iniţială, toate tranzistoarele de comutaţie sunt tranzistoare Schottky – ceea ce este indicat prin litera S în codul seriei respective.

1.4.2 Porţi logice TTL

Poarta NAND cu două intrări. Seria standard 74XX de porţi logice din familia de circuite logice TTL (Transitor-Transitor-Logic) a fost introdusă de firma Sylvania în 1963, dezvoltată şi comercializată de Texas Instruments. De fapt, la poarta NAND cu două intrări integrată s-a ajuns pornind de la structura de poartă NAND cu două intrări DTL (Diode-Transistor-Logic) care era deja realizată cu componente discrete, Figura 1.22-a. Poarta DTL este compusă dintr-un circuit MIN cu diode, ca în Figura 1.11-a, care realizează operatorul AND în logică pozitivă, şi la ieşirea căruia s-a ataşat un inversor realizat cu trazistorul T2. Diodele D3 şi D4 sunt introduse în divizorul format de R_1 şi R_2, conectat între $+5V$ şi $-2V$, pentru a produce o coborâre a nivelului de tensiune pe baza tranzistorului T2, asigurând o blocare sigură. Structura DTL a evoluat în structura TTL prin integrarea diodelor D1, D2, D3 sub forma unui tranzistor multiemitor T1, Figura 1.22-b. La această structură s-a adăugat un etaj de ieşire (totem-pole) compus din tranzistoarele T3 şi T4, Figura 1.22-c, obţinându-se structura standard de poartă NAND cu două intrări (7400). Această structură se compune din circuitul AND de intrare, circuitul defazor, realizat pe tranzistorul T2, pentru comanda tranzistoarelor T3 şi T4 din etajul de ieşire.



interval	T_1	T_2	T_3	T_4
V_{OH}→PF1	saturat	blocat	blocat	saturat
PF1→PF2	saturat	conduce	blocat	conduce
PF2→PF3	Toate tranzistoarele conduc in regim activ normal			
PF3→	activ invers	saturat	saturat	blocat

Figura 1.22 Poarta TTL standard: a) structura poţii DTL; b) structura iniţială a porţii TTL; c) structura (standard) pentru poarta 7400-NAND2; d) caracteristica statică de transfer şi regimurile de funcţionare ale transiztoarelor pentru poarta 7400.

Diodele D4 şi D5 pe intrare sunt pentru protecţia împotriva supratensiunilor negative care pot apare la intrare. Existenţa acestor diode va produce scurtcircuitarea la masă a oricărei tensiuni negative faţă de masa aplicată pe intrare şi care în valoare absolută este mai mare decât tensiunea de deschidere a diodei ($\approx 0,7V$).

Caracteristica statică de transfer a porţii se poate trasa prin segmente de dreaptă, la fel ca cea a inversorului din Figura 1.20, calculând punctele de frângere. Calculul acestor puncte de frângere este realizat în [Toacşe '96]. Cu intrarea B legată la $V_{CC} = 5V$ iar tensiunea V_A pe intrarea A parcurge intervalul $[0, V_{CC}]$ se obţin următoarele puncte de frângere:

- când $V_A = 0$. T1 conduce, T2 este blocat, T4 conduce iar ieşirea este în starea H, $V_{OH} = 3,6V$;
- $PF_1\,(0,6V; 3,6V)$ când $V_A = 0,6V$. T1 conduce iar T2 intră în conducţie, T4 conduce;

- $PF_2\,(1,3V; 2,5V)$ când $V_A = 1,3V$, T1 şi T2 conduc iar T3 intră în conducţie, T4 conduce;

- $PF_3\,(1,5V; 0,1V)$ când $V_A = 1,5V$. T1 conduce în regim activ inversat, T2 şi T3 conduc în saturaţie iar T4 este blocat, ieşirea este în nivel L.

În intervalul dintre PF_2 şi PF_3 conduc toate tranzistoarele, sursa este conectată la masă prin rezistenţele circuitului, puterea disipată corespunde componentei dinamice de scurtcircuit. Se recomandă ca parcurgerea acestui segment de caracteristică să se realizeze într-un interval de timp cât mai scurt ceea ce impune ca fronturile tensiunii de intrare V_I să fie cât mai abrupte.

Când T2 şi T3 conduc în saturaţie, tensiunea de comandă pe T4 este $V_{BE_{T4}} = V_{BE(sat)_{T3}} + V_{CE(sat)_{T2}} - V_{CE(sat)_{T3}} = V_{BE(sat)_{T3}} = 0,8V$, ceea ce ar determina intrarea în conducţie şi a acestui tranzistor. Pentru a împiedica conducţia lui T4, în emitorul său, se introduce dioda D şi astfel tensiunea de $0,8V$ nu este suficientă să deschidă cele două joncţiuni înseriate (joncţiunea BE(T4) + dioda D).

În starea H, $V_{OH} = 3,6V$, poarta generează un curent la ieşire, de la sursa V_{CC} prin R_3 şi T4, pentru intrările porţilor comandate, iar în starea L, $V_{OL} \approx 0,1V$, poarta absoarbe la masă, prin tranzistorul T3, curenţii de pe intrările porţilor comandate.

Seriei standard 74XX i s-au adus îmbunătăţiri, pe baza progreselor din domeniul tehnologic de integrare, obţinându-se serii cu performanţe mai ridicate. Prima nouă serie obţinută, după cea standard, a fost **seria Schottky 74SXX** care utilizează pentru comutaţie tranzistoare Schottky. Apoi, combinând tranzistoarele Schottky simultan cu creşterea rezistenţelor, pentru micşorarea curenţilor, s-a obţinut **seria TTL Schottky de putere redusă** (**Low-Power Schottky**) notată prin 74LSXX. Următoarea serie a fost **Schottky îmbunătăţită** (**Advanced Schottky TTL**) notată prin 74ASXX, care oferă viteză dublă faţă de 74SXX, dar la aceeaşi putere disipată. Seria îmbunătăţită a fost perfecţionată în contiuare pentru a se reduce puterea disipată şi s-a obţinut **seria Schottky îmbunătăţită de putere redusă** (**Advanced-Low-Power-Schottky TTL**) notată 74ALSXX. Cu performanţe de mediere între seriile AS şi ALS s-a realizat **seria rapidă** (**Fast TTL**) notată 74FXX şi care este probabil cea mai populară din familia TTL, pentru cerinţele de viteză în noile sisteme, la ora actuală. Sufixul XX din notaţia acestor serii este un număr zecimal şi constituie codul porţii respective, o poartă cu acelaşi XX, indiferent de serie, realizează aceeaşi funcţie logică. De exemplu, poarta logică NAND cu două intrări (NAND2) are următoarele referiri în funcţie de seria în care este implementată: 7400 (seria standard), 74S00, 74LS00, 74ALS00,74F00. Parametrii porţilor din seriile familiei TTL sunt prezentaţi în *Tabelul 1.9*.

O altă famillie de circuite logice în tehnologia bipolară, şi care se prezintă la fel sub formă de circuite integrate discrete, conţinând diferite tipuri de porţi este **ECL** (**E**mitter-**C**oupled-**L**ogic). Circuitele ECL sunt caracterizate prin viteze foarte ridicate ajungând până la $0,5ns$ pentru timpul de propagare. Viteza sporită la comutaţie se obţine tot prin evitarea regimului de saturaţie al tranzistorului dar nu prin tranzistor Schottky ci prin comutarea de curent între două canale. Structura de bază este cea de amplificator diferenţial cu un generator de curent în emitor (de unde şi denumirea de cuplaj prin emitor) iar curentul este comutat între cele două ramuri ale amplificatorului diferenţial [Toacşe 1996].

În familia ECL există două serii: 10K şi 100K, seria 100K având performanţe îm-

Tabelul 1.9 Parametrii seriilor din familia TTL

	Seriile familiei TTL					
Simbol parametru	74	74S	74LS	74AS	74ALS	74F
τ_{p} [ns] (NAND2)	10	3	9	1.7	4	3
P_d [mW] (NAND2)	10	19	2	8	1.2	4
PDP [p] (NAND2)	100	57	18	13.6	4.8	12
V_{ILmax} [V]	0.8	0.8	0.8	0.8	0.8	0.8
V_{OLmax} [V]	0.4	0.5	0.5	0.5	0.5	0.5
V_{IHmin} [V]	2.0	2.0	2.0	2.0	2.0	2.0
V_{OHmin} [V]	2.8	2.7	2.7	2.7	2.7	2.7
I_{ILmax} [mA]	-1.6	-2.0	-0.4	-0.5	-0.2	-0.6
I_{OLmax} [mA]	16	20	8	20	8	20
I_{IHmax} [μ A]	40	50	20	20	20	20
I_{OHmax} [μ A]	-800	-1000	-400	-2000	-400	-1000
τ_{pLHmax} [ns]	22	4.5	15	4.5	11	
τ_{pHLmax} [ns]	15	5	15	4	8	

bunătăţite faţă de seria 10K. Iată valorile câtorva parametrii din seria 100K: tensiunea de alimentare $V_{EE} = -4,5V$; salt de tensiune între niveluri $\Delta V = 0,8V$, $\tau_{\text{p}} = 0,75ns$, $P_d = 40mW$, $PDP = 30$. În aplicaţii porţile acestei familii nu sunt compatibile cu porţile din familia TTL sau CMOS. Porţile ECL îşi găsesc locul în sisteme logice şi de interfaţă de viteză foarte ridicată, evident cu un consum mare de putere, cum ar fi supercalculatoarele sau comunicaţiile de mare viteză pe cablu sau fibră optică (reţele ATM-Asynchronous Transfer Mode, Gigabit Ethernet).

Tot o tehnologie bipolară este cea referită ca logică integrată de injecţie, I^2L (**I**ntegrated **I**njection **L**ogic) care în raport cu tehnologia TTL a permis o anumită îmbunătăţire a compromisului viteză-putere disipată şi a densităţii de integrare pentru circuitele VLSI, dar care în timp, s-ar părea că, iese din utilizările curente.

Tehnologii alternative non-silicon sunt cele în arseniură de galiu,**GaAs**, sau cele cu **joncţiuni Josephson**. Dispozitivele în GaAs au viteză de comutaţie cam de cinci ori mai ridicată decât cele care se pot obţine în Si, sunt mai rezistente la radiaţii dar au un proces de fabricaţie mai complicat, în consecinţă sunt mai scumpe (mobilitatea electronului în siliciu poate ajunge până la $\mu_n = 1500cm^2/V \cdot s$ pe când în GaAs este $\mu_n = 8500cm^2/V \cdot s$). Caracteristici tipice pentru GaAs sunt $\tau_{\text{p}} < 200ps$, $P_d = 0,2mW$/poartă; există implementări de circuite integrate în GaAs dar nu există serii de porţi sub formă de circuite integrate discrete. Dispozitivele Josephson oferă $\tau_{\text{p}} = 1 \div 10ps$,$P_d = 1 \div 10mW$/poartă dar funcţionează la temperaturi foarte scăzute. Oricum, recentele dezvoltări de materiale semiconductoare la temperaturi obişnuite promit utilizarea acestor dispozitive în viitor.

1.4.3 Porţi pentru magistrale

Realizarea conectării, pentru comunicare, fiecare cu fiecare (punct-la-punct) a n puncte dintr-un sistem, necesită $\frac{n(n-1)}{2}$ legături distincte între aceste puncte. Evitarea acestor multor legături distincte se poate prin realizarea unei magistrale. O magistrală este o cale de transfer a informaţiei/semnalelor sub formă de cuvânt. Rezultă că prima

caracteristică a magistralei este lăţimea sa egală cu lungimea cuvântului, deci pentru transferul paralel a cuvântului de n biţi magistrala este compusă din n conductoare. În general lungimile uzuale de cuvânt sunt de : 8 biţi-1byte, 16 biţi-semicuvânt, 32 biţi-cuvânt, 64 biţi-dublu cuvânt.

La o magistrală de n biţi se conectează toate elementele unui sistem atât cele care sunt emiţătoare (talker) cât şi cele care sunt receptoare (listener); uneori un element poate avea funcţionare atât de emiţător cât şi de receptor, dar nu simultan. Regula de funcţionare în magistrală este: la un moment dat pe magistrală poate exista cel mult un singur emiţător activ, dar nu este limitat, teoretic, numărul receptorilor. Această regulă impune: dacă sunt mai multe emiţătoare legate la magistrală când unul dintre acestea este activ (înscrie informaţie pe magistrală) celelalte să fie inactive, adică din punct de vedere electric să fie deconectate. Pentru o astfel de funcţionare porţile de pe ieşirea emiţătorului trebuie să poată fi conectate sau deconectate la liniile magistralei după cum emiţătorul este activ sau inactiv. Pe o linie de magistrală, la care este conectată ieşirea porţii emiţătorului activ, pentru ieşirile porţilor emiţătoarelor inactive şi intrările porţilor receptoarelor, conectate la aceiaşi linie, trebuie realizate specificaţiile electrice care formează suportul nivelurilor logice de 0 şi 1. Pentru realizarea acestor specificaţii poarta normală este modificată în structura sa obţinându-se poarta cu colectorul în gol sau poarta TSL.

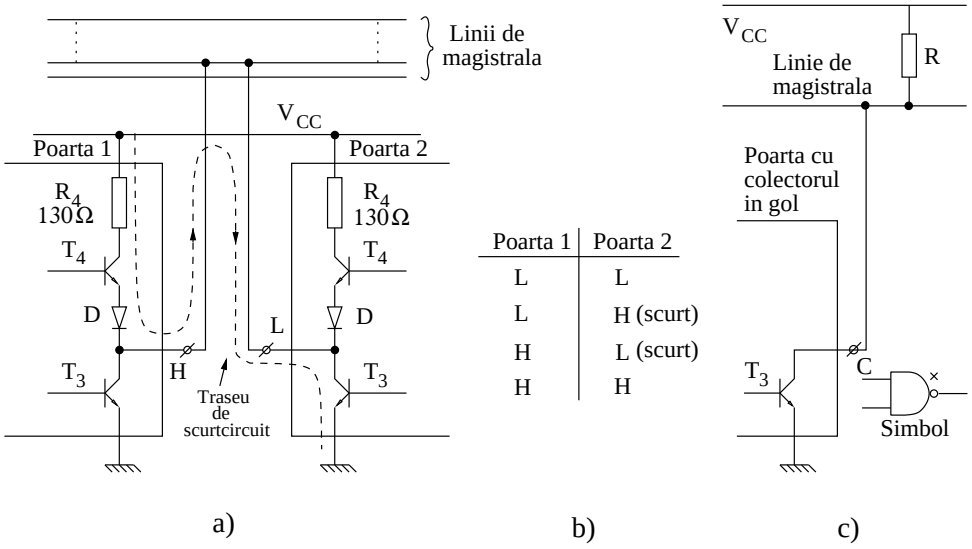


Figura 1.23 Poarta cu colectorul în gol: a) traseul de scurtcircuitare la conectarea în comun a ieşirilor porţilor TTL pentru condiţiile logice specificate în tabelul (b); c) structura etajului de ieşire la poarta cu colectorul în gol.

Poarta cu colectorul în gol. Prin conectarea ieşirilor a două sau mai multor porţi TTL, care au etajul de ieşire în contratimp (totem pole), la aceeaşi linie de magistrală, Figura 1.23-a, şi dacă două sau mai multe dintre aceste porţi au la ieşire stările logice opuse atunci sursa V_{CC} va fi scurtcircuitată la masă pe un traseu format din T4 şi T3 de la porţi diferite. Pentru eliminarea posibilităţii de apariţie a traseului

de scurtcircuit, la conectarea împreună a ieşirilor de la mai multe porţi, este necesar a se modifica structura etajului de ieşire. În acest sens, din etajul de ieşire se elimină repetorul pe emitor T4 rămânând numai tranzistorul T3 al cărui colector, la ieşire, este în gol obţinându-se poarta denumită cu colectorul în gol, Figura 1.23-c.

Colectorul în gol al tuturor porţilor se leagă împreună (cablează),Figura 1.24-a şi apoi se conectează printr-o rezistenţă comună la sursa de alimentare. Potenţialul pe colectorul comun, linia de magistrală, va fi în starea H numai când toate porţile au ieşirea în stare H şi va fi în stare L când cel puţin una din porţi are ieşirea în starea L. Această conexiune (cablare), în convenţia de logică pozitivă, realizează funcţia SI, de unde şi denumirea de SI-cablat (wired AND), iar în logică negativă funcţia SAU-cablat. Se poate obţine şi în logica pozitivă funcţia SAU-cablat iar în logică negativă SI-cablat dacă în etajul de ieşire nu se elimină tranzistorul care "trage" în sus ieşirea la nivelul H (pull-up transistor) ci se elimină tranzistorul care "trage" în jos ieşirea la nivelul L (pull-down transistor), ca în cazul porţilor din familia ECL.

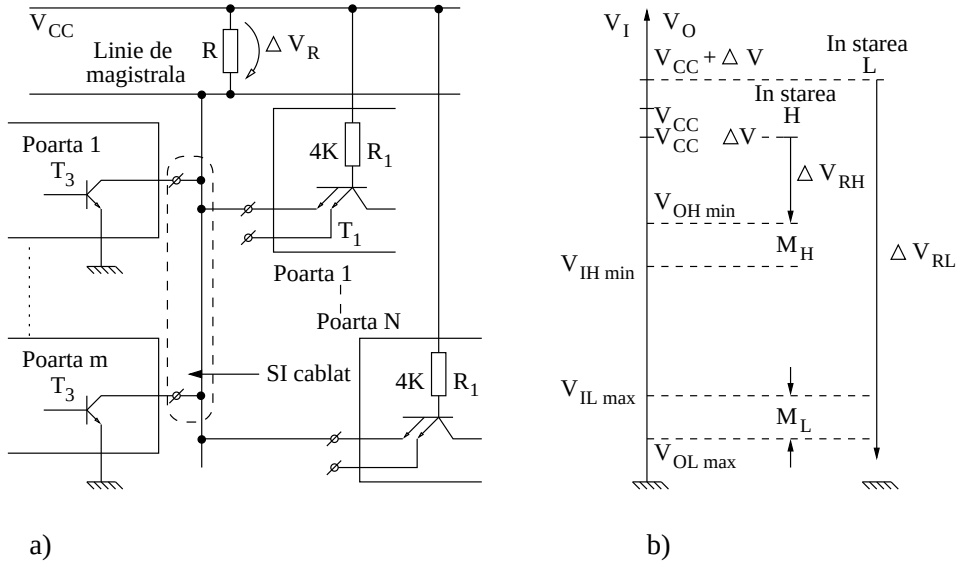


Figura 1.24 Explicativă pentru calculul rezistenţei comune R: a) cablarea porţilor cu colectorul în gol într-o conexiune SI-cablat; b) reprezentarea căderilor de tensiune pe rezistenţa comună pentru starea H şi L.

Calculul rezistenţei comune R, din colector, Figura 1.24, se face pentru asigurarea nivelurilor logice de tensiune garantate la ieşire şi a nivelurilor de tensiune permise la intrare, cu respectarea marginilor de zgomot M_H, M_L în condiţiile de variaţie ale tensiunii de alimentare $V_{CC} \pm \Delta V$. Pentru nivelul H la ieşire, care se obţine când toate cele m porţi emiţătoare au ieşirea în starea H, căderea de tensiune ΔV_{RH} pe rezistenţa comună nu trebuie să producă o scădere a tensiunii de ieşire sub valoarea minimă garantată $V_{OHminim}$.

$$R_{max}\,(m \cdot I_{OHmax} + N \cdot I_{IHmax}) \leq [V_{CC} - \Delta V - (V_{IHmin} + M_H)]$$

rezultă

$$R_{max} \leq \frac{V_{CC} - \Delta V - (V_{IHmin} + M_H)}{m \cdot I_{OHmax} + N \cdot I_{IHmax}} \qquad (1.26\text{-a})$$

Pentru nivelul L la ieşire, care se obţine când cel puţin o poartă emiţătoare este în stare L, căderea de tensiune ΔV_{RL} trebuie să determine o scădere a tensiunii de ieşire sub valoarea maximă garantată V_{OLmax}:

$$R_{min}\,(I_{OLmax} - N \cdot I_{ILmax}) \geq [V_{CC} + \Delta V - (V_{ILmax} - M_L)]$$

rezultă

$$R_{min} \geq \frac{V_{CC} + \Delta V - (V_{ILmax} - M_L)}{I_{OHmax} - N \cdot I_{ILmax}} \qquad (1.26\text{-b})$$

Deci rezistenţa comună de colector se alege în următoarele intervale de valori:

$$R_{min} \leq R \leq R_{max} \qquad (1.26\text{-c})$$

O utilizare curentă a porţilor cu colectorul în gol este comanda unor sarcini externe ca de exemplu becuri, LED-uri, bobine, relee sau rezistenţe.

Exemplul 1.12 O utilizare foarte frecventă a conexiunii SI-cablat apare la implementarea sistemului de întreruperi multiple de la microprocesoare. Microprocesorul (μP) posedă un singur terminal IRQ (Intrerupt ReQuest, cerere de întrerupere) pe care este înştiinţat că unul sau mai multe din periferice solicită în acel moment o cerere de întrerupere prin activarea (în starea L) a semnalului IRQ_{i_L}. Pentru că toate semnalele IRQ_{i_L} se aplică pe acelaşi IRQ, acestea formează un SI-cablat, deci toate ieşirile corespunzătoare de la periferice trebuie să fie cu colectorul în gol, Figura 1.25.

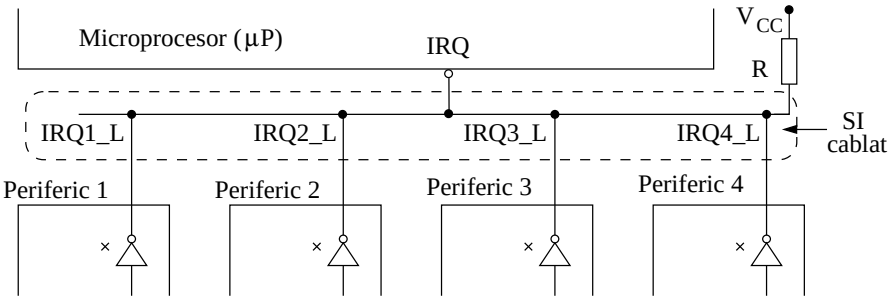


Figura 1.25 Sistemul de întreruperi multiple, pe intrarea de cerere întrerupere IRQ a unui microprocesor, realizat printr-o conexiune SI-cablat a porţilor cu colector în gol de la ieşirea perifericelor.

Exemplul 1.13 Folosind circuitul 74ALS136, patru porţi SAU EXCLUSIV cu două intrări, cu colectorul în gol, să se realizeze un detector pentru cuvântul de patru biţi: $b_3\,b_2\,b_1\,b_0 = 1001$. Rezultatul operaţiei de detectare comandă 5 circuite 74ALS136.

Soluţie. O poartă SAU EXCLUSIV va avea ieşirea în starea H numai când cele două intrări sunt diferite. În cazul când ieşirile formează un SI-cablat nivelul de ieşire este H (se detectează cuvântul $b_3\,b_2\,b_1\,b_0 = 1001$) numai dacă pe cea de a doua intrare a porţilor este aplicat cuvântul 0110, ceea ce rezultă din relaţia $(b_3 \oplus 0)\,(b_2 \oplus 1)\,(b_1 \oplus 1)\,(b_0 \oplus 0) = 1$, Figura 1.26.

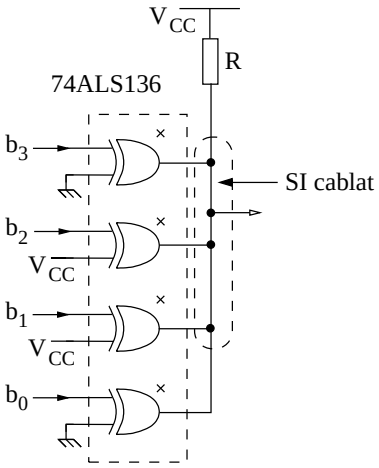


Figura 1.26 Comparator pentru cuvântul $b_3\,b_2\,b_1\,b_0 = 1001$

Se dau următoarele date $V_{CCmin} = 4,5V$, $V_{CCmax} = 5,5V$, $M_H = 0V$, $M_L = 0V$, $m = 4$, $N = 5$ iar din *Tabelul 1.9* se citesc parametrii: $I_{OLmax} = 8mA$, $I_{OHmax} = 400\mu A$, $I_{IHmax} = 20\mu A$, $I_{ILmax} = 0,2mA$, $V_{OHmin} = 2,7V$, $V_{OLmin} = 0,5V$.

Aplicând relaţiile 1.26 rezultă pentru rezistenţa comună o valoarea în intervalul $714\Omega \leq R \leq 1050\Omega$.

Exemplul 1.14 Circuitul 7406 este un hex buffer neinversor cu colectorul în gol ce poate comanda tensiuni de maxim 30V şi absorbi un curent $I_{OLmax} = 40mA$. Să se comande un bec (cu filament) de semnalizare cu $U_n = 14V$, $I_n = 80mA$ când este alimentat de la o tensiune $V = 20V$.

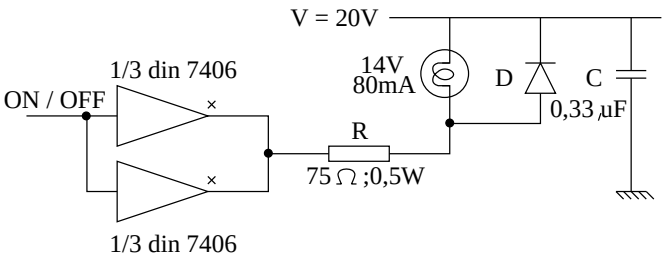


Figura 1.27 Comandă realizată cu bufferul cu colectorul în gol 7406

Soluţie. Deoarece curentul $I_n > I_{OLmax}$ trebuie să se conecteze în paralel două buffere (1/3 din 7406), Figura 1.27. Pentru preluarea căderii de tensiune $\Delta V = 20V - 14V = 6V$ se

înseriază cu becul o rezistenţă $R = \frac{6V}{80mA} = 75\Omega$, puterea disipată pe această rezistenţă este $\frac{\Delta V^2}{R} = 0,48W$, deci se alege o rezistenţă de 75Ω care admite o putere disipată de $0,5W$.

Pentru ca salturile de curent generate la comutarea bufferelor să nu producă zgomot, pe linia de alimentare de la sursă (vezi 1.6.2.4), se recomandă conectarea, ca filtru, a unui condensator cu tantal de $0,33\mu F$ pentru fiecare grup de 4 circuite 7406. În plus, pentru ca inductivitatea filamentului L să nu producă supratensiuni periculoase pentru tranzistorul de ieşire al bufferului, se recomandă ca în paralel cu becul să se conecteze o diodă D de protecţie (supresoare, amplitudinea tensiunii electromotoare generată pe inductivitate, $-L\frac{di}{dt}$, va fi limitată la valoarea tensiunii pe dioda de protecţie, $\sim 0,8V$).

Dezavantajele porţilor cu colectorul în gol sunt:

- Deoarece impedanţa de ieşire în starea H este mare ($R > R_{T4}$) rezultă pentru durata fronturilor τ_{LH} şi a timpilor de propagare, τ_{pLH}, la tranziţia din L în H, valori mult mai mari decât la porţile normale (care au tranzistorul T4).

- Imunitate mai scăzută la zgomot şi necesită o rezistenţă comună R calculată de fiecare dată în funcţie de circuit, valori uzuale între 470Ω şi $4,7K\Omega$.

Poarta cu trei stări logice, TSL (Three-State-Logic). Structura porţii TSL se obţine prin completarea porţii normale TTL, Figura 1.22-c, cu dioda D1 şi a inversorului I ca în Figura 1.28-a. Când **semnalul de validare a ieşirii,** OE_L (**O**utput **E**nable) este activ, pe emitorul E2 şi pe catodul lui D1 tensiunea aplicată este H, deci poarta funcţionează normal, ieşirea este validată şi poate fi în starea L sau H după cum este valoarea logică aplicată pe intrarea A. Dacă OE_L nu este activ, este în nivelul H, atunci tensiunea de valoare L de la ieşirea inversorului aplicată pe: emitorul E2 va duce la blocarea tranzitorului T3 din etajul de ieşire şi aplicată pe catodul diodei D1 va comanda conducţia acesteia, deci semnal L pe baza lui T4, ceea

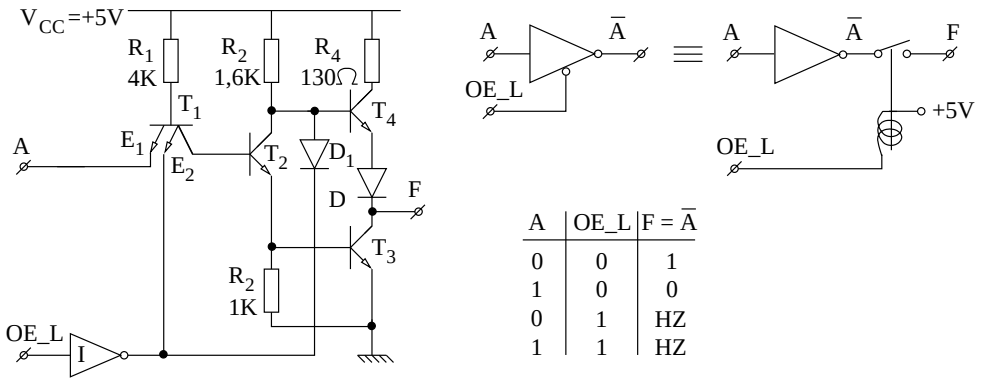


Figura 1.28 Poarta TSL: a) schemă electrică; b) echivalarea porţii TSL cu o poartă normală având un contact pe ieşire (comandat de semnalul de validare OE_L).

ce duce şi la blocarea tranzistorului T4 din etajul de ieşire; ambele tranzistoare din etajul de ieşire sunt blocate.

Poarta generează la ieşire (numai!) cele două stări logice H sau L când comanda de validare este activă $OE\_L = 0$, iar când comanda de validare este inactivă $OE\_L = 1$ poarta are cele două tranzistoare $T3$, $T4$ blocate. Ultima stare, cu T3, T4 blocate, nu este o stare logică comandată, prin semnalele logice aplicate pe intrările porţii, aceasta este denumită **stare de înaltă impedanţă HZ** (High Z). În starea HZ potenţialul pe ieşirea porţii este fixat de potenţialul care există pe linia de magistrală la care această ieşire este conectată (şi nu de către poarta TSL). Potenţialul pe linia de magistrală este forţat în 0 sau în 1 de către o altă poartă legată la aceea linie de magistrală, poartă care este comandată în starea normală de funcţionare. Dacă pe linia de magistrală nivelul forţat este H atunci trazistorul T4 de la o poartă TSL în HZ va genera un curent de maxim ordinul μA, iar dacă nivelul pe linia de magistrală este forţat în L atunci tranzistorul T3 de la o poartă TSL în HZ va absorbi un curent maxim de ordinul μA. Cu aceste valori foarte mici ale curenţilor absorbiţi sau generaţi de poarta TSL în HZ se poate considera că poarta, electric, este deconectată de la linia de magistrală. Poarta TSL poate fi echivalată cu o poartă normală în a cărei ieşire este înseriat un contact care este închis când Output Enable este activ, $OE\_L = 0$, şi este deschis (în starea HZ), când Output Enable nu este activ, $OE_L = 1$, Figura 1.28-b.

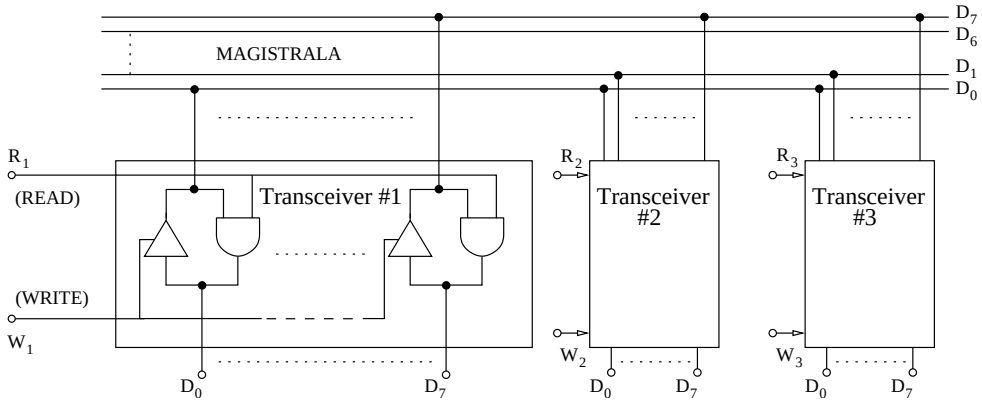


Figura 1.29 Conectarea la magistrală a trei blocuri bidirecţionale de 8 biţi prin intermediul circuitelor transceiver

Pentru blocurile care sunt conectate la o magistrală şi care pot avea o funcţionare atât de emiţător cât şi receptor transferul spre linia de magistrală trebuie să fie bidirecţional. Aceste blocuri bidirecţionale trebuie să posede pe ieşire un circuit TRANSCEIVER (TRANSmitter + reCEIVER) care este compus pentru fiecare linie/bit a magistralei dintr-un buffer TSL şi o poartă SI conectate în paralel, Figura 1.29. La magistrala din figură participă trei elemente fiecare având pe ieşire un transceiver de opt biţi. Înscrierea pe magistrală (WRITE) se face prin activarea, în starea H, doar a unui singur semnal $W_i (i = 1, 2, 3)$, la bufferele TSL, pentru a asigura accesul doar a unui singur emiţător la un moment dat. Citirea de pe magistrală (READ) se activeează prin semnalul $R_i (i = 1, 2, 3)$ la porţile AND. La transceiverul

i semnalele W_i şi R_i sunt exclusive adică nu pot fi active simultan $W_i \cdot R_i = 0$.

Avantajele utilizării porţilor TSL în raport cu utilizarea celor cu colectorul în gol sunt:

- Oferă o impedanţă mică la ieşire în H şi L (ca la poarta normală);
- Nu necesită rezistenţă adiţională ;
- În starea de HZ încarcă insignifiant circuitele cu care sunt cuplate la ieşire (la un moment dat încarcă numai circuitul care forţează potenţialul pe linia de magistrală).

1.5 PORŢI ÎN TEHNOLOGIA CMOS

Componentele integrate în electronică, introduse la începutul anilor 1960, au fost realizate aproape în exclusivitate în tehnologia bipolară, timp de două decenii, până la începutul anilor 1980. În deceniul al nouălea a fost trecerea de la tehnologia bipolară la tehnologia unipolară în special la cea CMOS. Începând cu anii 1990, aproape în exclusivitate, circuitele integrate produse au fost CMOS. Schimbarea de tehnologie s-a produs când au apărut circuitele integrate pe scară largă **VLSI** (**V**ery **L**arge **S**cale **I**ntegration) care conţineau de ordinul sutelor de mii de componente pe cip. La acest ordin de mărime al densităţii de integrare, datorită puterii disipate ridicate, deci regim termic solicitant pentru circuit (datorită puterii disipate ridicate), tehnologia bipolară nu mai era corespunzătoare, iar alternativa a fost tehnologia CMOS. Prin valorile parametrilor: margine de zgomot ridicată, puterea disipată în curent continuu aproape neglijabilă, fan-out ridicat, viteză mărită şi înalt grad de automatizare în proiectare şi implementare, tehnologia CMOS s-a impus. Astăzi circuitele CMOS domină piaţa circuitelor integrate de la cele simple la cele mai complexe.

1.5.1 Tranzistorul MOSFET

1.5.1.1 Tehnologia de fabricaţie a tranzistorului MOS

Tranzistorul MOS sau extinzând la circuitul integrat, fizic, constau din zone difuzate de conductivitate p^+ sau n^+ incluse într-un substrat de siliciu, Si, între care există anumite conexiuni şi la care, din exterior, se aplică tensiuni de polarizare. Pentru obţinerea acestor zone de conductivitate p^+ sau n^+ este necesar întâi a se realiza ferestre pentru difuzie, în locurile respective, pe placheta de siliciu. Succesiunea etapelor pentru realizarea acestor ferestre este descrisă succint în continuare.

Dintr-un lingou de Si, sub forma unui cilindru, cu rezistivitatea aproximativ $10\Omega \cdot cm$ se taie discuri (wafer) cu grosimea în jur de $0.6mm$, Figura 1.30-a. Aceste discuri în tehnologia actuală pot fi cu un diametru nu cu mult mai mare de 30 cm; pe una din suprafeţele waferului se realizează simultan un număr de zeci sau sute de circuite integrate identice, apoi prin tăiere se obţin plachetele cu circuitul integrat.

Pe suprafaţa discului printr-un proces de oxidare se realizează un strat de bioxid de siliciu, SiO_2, referit prin termenul de oxid. Există două procedee de oxidare: umedă şi uscată. Oxidarea umedă este un rapid proces de oxidare într-o atmosferă

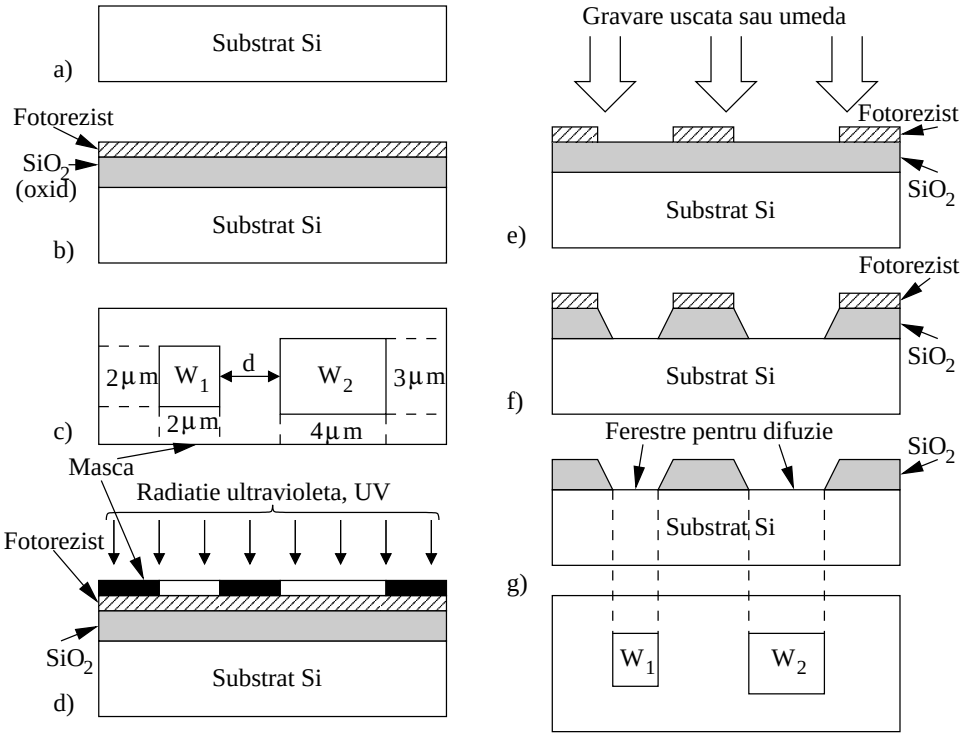


Figura 1.30 Etapele realizării ferestrelor prin litografie optică: a) substrat (wafer); b) ansamblu substrat + strat de SiO_2 + fotorezist; c,d) ansamblu cu mască supus radiaţiei UV; e,f) înlăturarea stratului de SiO_2 prin gravare; g) substrat cu ferestrele (pentru difuzie) realizate în stratul de oxid.

care conţine vapori de apă la temperaturi între $900° \div 1000°C$. Oxidarea uscată se realizează într-o atmosferă de oxigen pur la o temperatură în jur de $1200°C$, viteza de creştere a grosimii stratului de oxid este mai mică decât cea obţinută prin procedeul de oxidare umedă. Două caracteristici fizice ale stratului de oxid sunt utilizate în tehnologia circuitelor integrate: rezistivitatea electrică mare, deci poate constitui un strat izolator foarte bun, şi impenetrabilitatea impurităţilor, deci poate constitui un strat opac pentru difuzia de impurităţi. Apoi, prin depunerea unei picături de fotorezist peste stratul de oxid şi centrifugarea întregului wafer, se formează un strat de fotorezist cu grosimea aproximativ $1\mu m$, Figura 1.30-b. Încălzit la o temperatură de aproximativ $100°C$ stratul subţire de fotorezist de pe suprafaţă se întăreşte (devine sticlos).

Urmează apoi confecţionarea măştilor pentru realizarea ferestrelor de difuzie, care este o etapă foarte complexă şi costisitoare. Să presupunem că se doreşte realizarea a două ferestre dreptunghiulare cu dimensiunile $W_1 = 2\mu m \times 2\mu m$, $W_2 = 3\mu m \times 4\mu m$ pentru care s-a confecţionat masca din Figura 1.30-c. Această mască se aplică peste stratul de fotorezist întărit (sticlos) şi întreg ansamblul este expus într-un fascicul de radiaţie ultravioletă UV, Figura 1.30-d. Fotorezistul de sub ferestrele măştii (zonele

transparente) este penetrat de radiaţia UV, polimerizează şi devine solubil spre deosebire de cel din zonele opace ale măştii unde fotorezistul rămâne întărit (sticlos). Apoi, prin spălare cu solvent, este înlăturat fotorezistul solubil din zonele de sub ferestre dar nu şi cel întărit de pe restul suprafeţelor. Acest tip de fotorezist este numit pozitiv, pentru că există şi fotorezist negativ care este iniţial solubil şi devine insolubil după ce este penetrat de UV; evident pentru fotorezistul negativ masca trebuie să aibă transparenţă în zonele unde nu trebuie realizate ferestre pe substrat. Măştile pentru cele două tipuri de fotorezist sunt una complementară celeilalte.

În continuare, ansamblul obţinut este supus unui proces de gravare. Prin gravare stratul de oxid de siliciu din zonele care au rămas neacoperite de fotorezist este înlăturat (până la suprafaţa de substrat) obţinându-se fereastra, Figura 1.30-e şi 1.30-f. Gravarea se poate face cu un solvent chimic (săruri ale acidului florhidric, HF), referită ca gravare umedă, sau printr-o tehnologie cu plasmă, referită ca gravare uscată. Apoi, se utilizează un alt tip de solvent care poate înlătura stratul întărit de fotorezist rămas deasupra stratului de oxid, Figura 1.30-g, terminându-se astfel această succesiune de operaţii de realizare a ferestrelor denumită fotolitografie sau litografie optică ce poate fi utilizată cu rezultate bune până la rezoluţii de $0,8\mu m$. $R = k \cdot \lambda / AN$; R–rezoluţia, k–un parametru al stratului de fotorezist, λ–lungimea de undă a radiaţiei utilizate, AN–apertura numerică (puterea de rezoluţie a aparatului utilizat).

Tehnologia fotolitografică permite o reducere cu 30%, la fiecare doi ani, a valorii caracteristicii de proces. Dar, actual, s-a ajuns la valori ale caracteristicii de proces care sunt sub lungimile de undă utilizate în litografia optică. Pentru depăşirea limitărilor litografiei optice au apărut alte variante, ca de exemplu litografia **EUV** (**E**xtreme **U**ltra-**V**iolet), care utilizează lungimea de undă de $13,4nm$—lungime cu peste un ordin de mărime mai scurtă decât lungimile de undă din litografia optică—dar cu care se pot obţine linii de trasee de dimensiuni sub $50nm$.

Fotolitografia se repetă ori de câte ori este necesară realizarea unei noi ferestre; evident ferestrele care sunt pentru utilizări similare la diferite circuite şi sunt pe acelaşi wafer se realizează în paralel (în Figura 1.30 au fost realizate simultan două ferestre W_1 şi W_2).

Pentru rezoluţii sub $0,8\mu m$ litografia optică, recent, este substituită de litografia cu fascicol de electroni. Avantajele litografiei cu fascicol de electroni în realizarea ferestrelor în raport cu cea optică sunt:

- fereastra poate fi generată direct digital fără confecţionarea măştii;
- nu există succesiunea tuturor etapelor (măşti, expunere UV etc.) procesul de gravare este direct;
- modificarea formelor şi dimensiunilor ferestrelor precum şi plasarea lor pe suprafaţa waferului se pot realiza uşor şi repede.

Pentru realizarea unui tranzistor nMOS procesul este descris în continuare. Se realizează pe un substrat de siliciu de conductivitate de tip p (conductivitate prin goluri) o fereastră pentru **zona activă**, adică zona pe suprafaţa de Si unde se va implementa tranzistorul, Figura 1.31-a. Stratul gros de oxid (~ 5000Å,1Å$= 10^{-10}m$) denumit oxid de câmp care înconjoară zona activă are rolul de a izola tranzistorul de alte tranzistoare. Uneori, izolarea tranzistorului de cel vecin se face şi în substrat

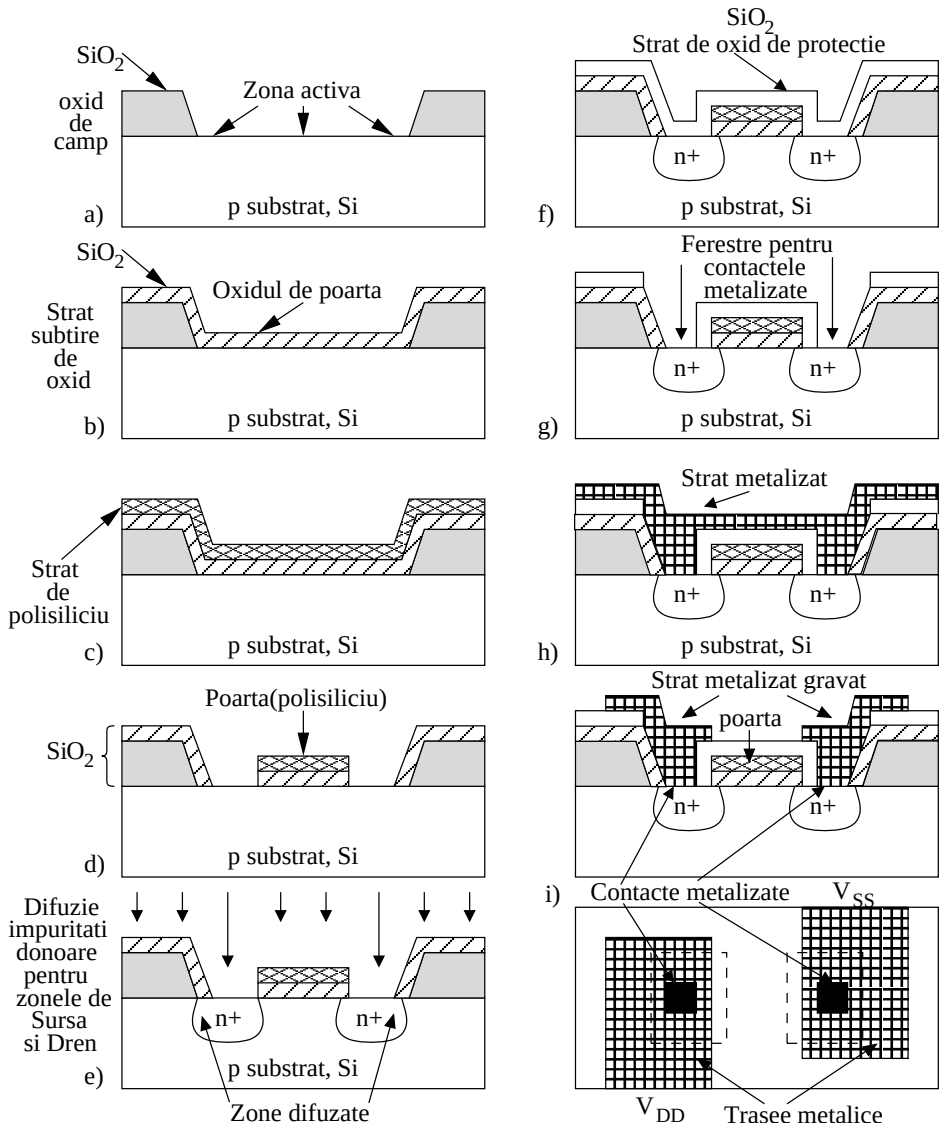


Figura 1.31 Etapele în tehnologia de realizare a unui tranzistor nMOS

prin difuzarea între zonele active ale acestor tranzistoare vecine a unei zone de tip p^+ denumită zonă de stopare (se pot vedea astfel de zone de stopare în structura din Figura 1.19-c). Notaţiile n^+ sau p^+ semnifică faptul că există o concentraţie de purtători de tip n respectiv de tip p mult mai mare decât concentraţiile normale.

Apoi, întreaga suprafaţă este acoperită cu un strat subţire (aproximativ 200 Å) de oxid de calitate superioară, care va fi şi stratul izolator de sub poarta viitorului tranzistor implementat în zona activă, Figura 1.31-b. Calitatea şi grosimea acestui strat de oxid determină foarte pronunţat tensiunea de deschidere/prag a viitorului tranzistor. În continuare se depune un strat de polisiliciu, Figura 1.31-c, din care, după un proces de gravare, se formează trasa ce va constitui poarta tranzistorului, Figura 1.31-d. Acest strat depus cu grosimea aproximativă de 3000Å din polisiliciu se realizează printr-un proces de depunere de vapori de Si. Materialul de substrat (waferul) este un singur cristal, adică în întreg volumul substratului structurarea cristalografică este aceeaşi. În schimb, materialul de polisiliciu al porţii este format, după cum şi denumirea exprimă, din mai multe zone fiecare cu dispunere cristalografică diferită. Zonele de material policristalin, referite şi prin termenul de **polisiliciu**, sunt utilizate pentru realizarea electrodului de poartă sau a unor trasee de conexiune. Totodată materialul policristalin, ca şi oxidul sau nitrura de siliciu, este utilizat şi ca material de mascare în procesul de difuzie al impurităţilor; sunt impenetrabile la impurităţi. Ansamblul obţinut până în această etapă are deja realizate două ferestre pentru implementarea zonelor de sursă şi dren ale tranzistorului.

Implementarea zonelor de dren şi sursă rezultă în urma dopării cu impurităţi donoare (Fosfor sau Arseniu) care schimbă conductivitatea din tip p a substratului, din zonele respective, în conductivitate de tip n^+. Doparea poate fi realizată prin difuzie sau prin implantare ionică. Pentru difuzie este necesară o atmosferă ce conţine impurităţi donoare la o anumită presiune şi o temperatură de peste $800°C$, impurităţi ce pătrund în substrat pe o anumită adâncime. În procesul de difuzie stratul de polisiliciu al porţii împreună cu stratul oxid înconjurător îndeplinesc rolul de mască (impenetrabile pentru impurităţile donoare difuzate) în jurul celor două zone pentru dopare, Figura 1.31-e. Utilizarea stratului de polisiliciu gravat – adică poarta – ca mască duce la o foarte precisă poziţionare a difuzării zonelor de sursă şi dren faţă de poarta tranzistorului, această procedură mai este referită ca **proces de autoaliniere**. Prin **implantare ionică**, folosind un spectrometru de masă şi un câmp electric accelerator, se imprimă ionilor de impuritate o energie (între $10 \div 100KeV$, $1eV = 1,6 \times 10^{-19}J$) suficientă ca prin impact cu suprafaţa de substrat să penetreze în adâncime câţiva microni şi să schimbe în zona respectivă tipul de conductivitate.

După realizarea zonelor de conductivitate n^+ pentru dren şi sursă se acoperă întreaga suprafaţă cu un strat de oxid (de protecţie), Figura 1.31-f, care apoi este gravat pentru realizarea unor ferestre, realizându-se acces până la suprafaţa de substrat de conductivitate n^+, Figura 1.31-g; suprafeţele acestor ferestre pe zonele dopate de tip n^+, după metalizare, vor forma contactele metalice la dren şi sursă. Printr-un proces de evaporare în vid se depune pe întreaga suprafaţă un strat de aluminiu, Al, cu grosimea de 5000÷6000Å, Figura 1.31-h. Stratul metalic depus pe întreaga suprafaţă este apoi gravat astfel încât să formeze pe suprafaţa superioară a ansamblului traseele metalice pentru interconexiuni, aceste trasee pătrund prin contactele metalizate până la suprafaţa zonelor difuzate de dren şi sursă.

Figura 1.31-i reprezintă secţiunea şi o vedere în plan a părţii superioare a tranzis-

torului nMOS (realizat într-un substrat de tip p). Pe vederea în plan, a părţii superioare a ansamblului (desenul de jos), sunt două trasee metalice, pentru aducerea potenţialelor de sursă V_{SS} şi de dren V_{DD}, iar în interiorul acestor trasee sunt figurate două pătrate înegrite corespunzătoare contactelor metalice care ajung la suprafeţele n^+ de dren şi de sursă. În această figură nu este scos la suprafaţă terminal de polisiliciu al porţii (stratul de polisiliciu se vede doar în secţiune, cu haşură dublă, în desenul de sus). Zona de sub poartă este zona de canal (indus).

Pentru realizarea unui tranzistor pMOS se porneşte de la un substrat de tip n; etapele sunt identice cu deosebirea că impurităţile utilizate în procesul de difuzie, pentru obţinerea zonelor dopate de tip p^+ pentru dren şi sursă, sunt de tip acceptor (Bor).

În exemplul acesta pentru realizarea conexiunilor necesare pe suprafaţa superioară a cipului/plachetei a fost suficient un singur strat metalic (zona caroiată pregnant din Figura 1.31-h). Pentru circuitele VLSI complexe realizarea tuturor conexiunilor necesită mai multe straturi metalice suprapuse, în tehnologia actuală pentru μP s-a ajuns până la opt straturi metalice, fiecare strat metalic fiind izolat de cel de sub sau de cel de deasupra printr-un strat izolator de oxid. Trecerea conexiunilor între straturile metalice de niveluri diferite, sau între straturi şi suprafaţa de substrat a zonelor (difuzate) de diferite conductivităţi, se face prin intermediul unor găuri (vias) realizate pe verticală prin straturile izolatoare de oxid. Devine mult mai importantă şi mai complexă operaţia de realizare a straturilor/conexiunilor decât realizarea componentelor (tranzistoarelor) în substrat (circuitele integrate complexe sunt caracterizate prin numărul de straturi metalizate, iar pentru traseele metalizate, actual se utilizează cuprul).

Activitatea de proiectare a unui circuit integrat poate fi complet separată în timp şi spaţiu de procesul de fabricaţie a circuitului integrat de la turnătoria de siliciu. Legătura între proiectant şi inginerul de proces se face prin intermediul regulilor de proiectare a layout-ului (geometria pe siliciu). Aceste reguli de proiectare exprimă anumite constrângeri impuse de procesul de fabricaţie de la o anumită turnătorie de siliciu. Constrângerile pot fi referitoare la lăţimea minimă a unei linii/trase realizabilă cu acel proces, distanţa minimă între două linii, dimensiunile minime pentru o zonă de difuzie etc.; de exemplu în Figura 1.30-c se poate prescrie distanţa minimă dintre cele două zone de difuzie pentru un anumit proces. Respectând aceste reguli geometrice de proiectare proiectantul va proiecta măştile necesare procesului care, probabilistic, duc la un procent de recoltă ridicat (exprimare procentuală a numărului de circuite funcţionale obţinute pe un wafer din numărul total de circuite pornite iniţial pe un wafer) şi la circuite cu o siguranţă mare în funcţionare.

Regulile de proiectare a layout-ului pot fi exprimate în două modalităţi: micronic şi pe bază de λ. Regulile pe bază de lambda (λ) cuprind toate constrângerile referitoare la optenabilitatea dimensiunilor geometrice ca multiplu întreg (uneori fracţionar) de λ. Practic, acestea sunt un set de relaţii, toate în funcţie de λ, cu care proiectantul calculează dimensiunile layout-ului pentru un anumit circuit. Lambda este **caracteristica de proces**, adică rezoluţia/dimensiunea minimă garantată pe care o poate realiza acel proces. De exemplu, în Figura 1.30-c distanţa dintre cele două zone difuzate ar putea fi exprimată prin relaţia $d = \lambda$ sau lăţimea minimă a trasei de polisiliciu pentru poartă este 2λ, etc. Avantajul exprimării pe bază de λ este păstrarea aceloraşi relaţii de proiectare când se trece de la un proces la unul îmbunătăţit, de exemplu

când se trece de la un proces cu $\lambda = 1\mu m$ la unul cu $\lambda = 0,6\mu m$ (relaţiile geometrice de proiectare se înmulţesc cu $\frac{0,6}{1} = 0,6$). Teoretic, această scalare a lui λ când se trece la un proces îmbunătăţit ar fi fezabilă dar, practic, se constată că atunci când **procesul este submicronic** (corespunde la lungimi de canal sub $1\mu m$) şi în special la **procesele adânc submicronice** (lungimi de canal sub $0,35\mu m$) scalarea lui λ nu mai este liniară şi constantă.

Exprimarea micronică a regulilor se reduce la exprimări în valori absolute, în μm, pentru dimensiunile layout-ului, ceea ce elimină inconsistenţa care apare la regulile pe bază de lambda când se aplică scalarea. Uneori cele două modalităţi, de exprimare a informaţiei necesare proiectantului pentru definirea layout-ului, sunt mixate.

1.5.1.2 Ecuaţiile tranzistorului MOS

Structura tranzistorului MOS (Metal-Oxid-Semiconductor) este cea din Figura 1.31-i, iar pentru o explicare a funcţionării este reprezentat, într-o nuanţă didactică, în Figura 1.32-a. În principiu, un tranzistor MOS se reduce la un condensator metal-semiconductor, cele două armături fiind una o placă G (Gate) din polisiliciu sau metal, de dimensiuni $W \times L$, iar cealaltă armătură este constituită din substratul de semiconductor B, între acestea dielectricul este format din stratul de SiO_2 cu grosimea D_{ox} (< 200Å). Când se aplică o tensiune pozitivă între poartă şi substrat $V_{GC} > 0$ (grilă - canal), deoarece densitatea de purtători de sarcină[1] -electroni liberi- în placa metalică este mult mai mare decât densitatea de purtători liberi în semiconductor, câmpul electric dintre plăcile acestui condensator metal-semiconductor pătrunde în substrat dar nu pătrunde în placa metalică. Sarcina electrică (pozitivă în cazul acesta) de pe poarta (placa metalică) are o repartizare de suprafaţă iar sarcina negativă, indusă în substratul de tip p, are o repartizare volumetrică; la interfaţa cu oxidul sunt atraşi electronii liberi din substrat care formează un strat de sarcină negativă cu grosimea ≈ 50Å. Acest strat de sarcină liberă, indusă la suprafaţa substratului, este referit prin (zona de) canal (indus). Cantitatea de sarcină negativă indusă în canal este determinată de intensitatea câmpului dintre poartă şi substrat, deci de valoarea tensiunii V_{GC} (sau V_{GB}). Deoarece la cele două capete ale canalului există două "rezervoare" de sarcină negativă liberă, când V_{GC} este este suficient de mare ca să inducă un canal cele două rezervoare, obţinute prin difuzia de dren D şi de sursă S, acestea vor fi unite printr-o cale conductivă - canalul indus în substrat, cu lungimea L. Aplicând o diferenţă de potenţial între cele două zone de dren şi de sursă apare o deplasare de electroni în canal, deci un curent de canal. În concluzie, se poate afirma că tranzistorul MOS este un dispozitiv la care valoarea curentului de conducţie în canalul, dintre dren şi sursă, este controlat de tensiunea aplicată pe poartă. Deci intensitatea curentului din canal este o funcţie de tensiune aplicată dintre dren şi sursă, V_{DS}, şi tensiunea poartă-canal V_{GC}, $I_{canal} = f(V_{DS}, V_{GC})$. Pornind de la această explicaţie calitativă a funcţionării dispozitivului MOS, în continuare, vor fi deduse relaţii simple care pot exprima şi cantitativ funcţionarea dispozitivului nMOS.

Tensiunea pozitivă V_{GC}, pentru valori crescătoare pornind de la 0V, va respinge sarcina mobilă pozitivă din zona de interfaţă oxid-substrat înspre masa substratului, adică golurile care sunt purtători mobili majoritari în substratul de tip p. În urma

[1] Densitatea de electroni liberi în: metale - $10^{28}/cm^3$, izolatoare - $10^7/cm^3$, semiconductoare - $2 \cdot 10^{13}/cm^3$ pt Ge şi $1,45 \cdot 10^{10}/cm^3$ pt Si

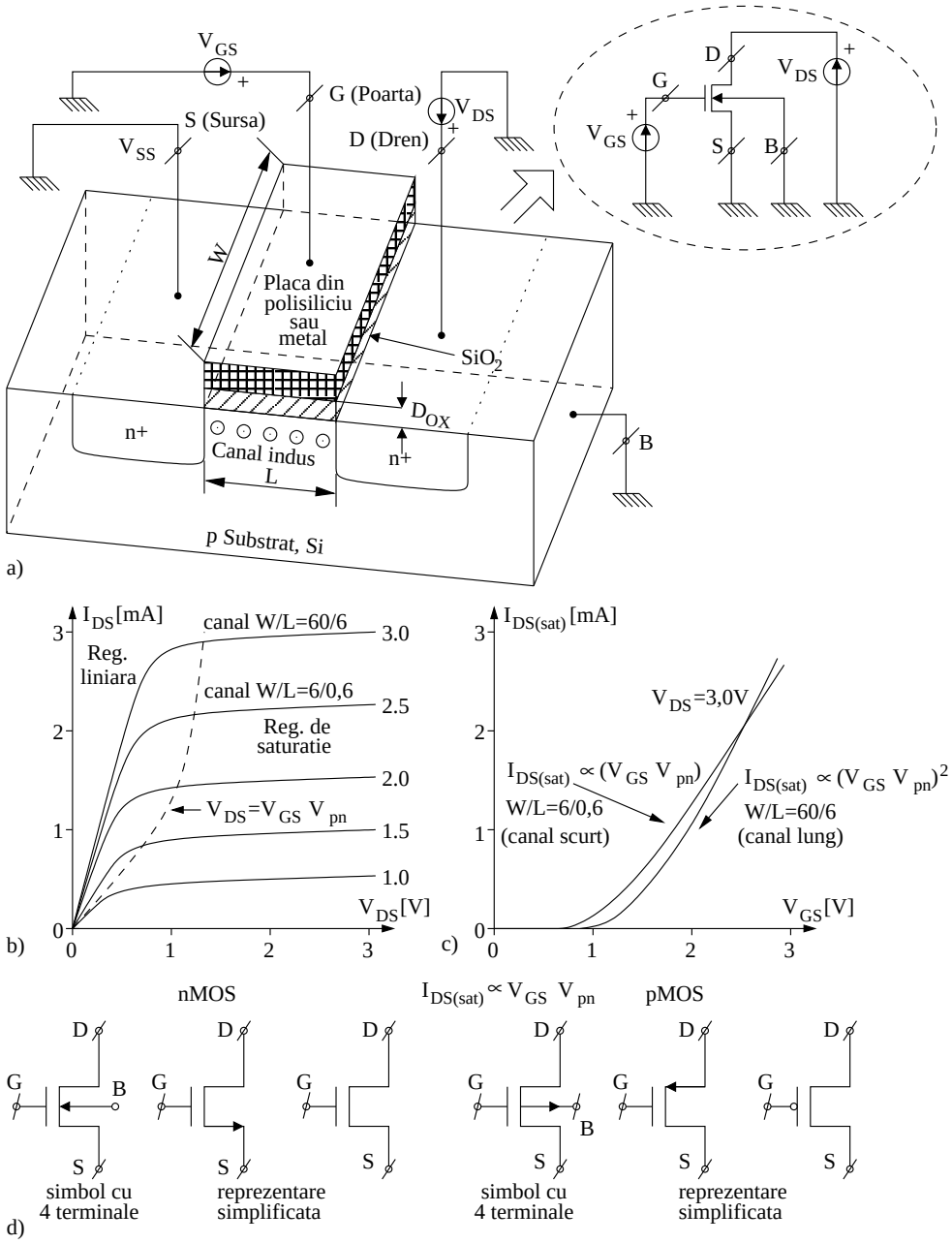


Figura 1.32 Tranzistorul nMOS: a) structură fizică de principiu pentru un tranzistor nMOS; b) caracteristicile statice de ieşire $I_D = f(V_{DS})_{V_{GS}=const}$; c) dependenţa I_{Dsat} de tensiunea V_{GS}; d) simbolurile de reprezentare pentru tranzistoarele nMOS şi pMOS.

acestei respingeri în zona de interfaţă a stratului rămâne doar sarcina negativă, fixă în reţeaua cristalină, formată din ionii negativi ai atomilor acceptori. Rezultă că în stratul de interfaţă apare un regim de sărăcire de purtători majoritari şi chiar la aplicarea unei tensiuni între dren şi sursă curentul din canal este nul.

Dar când tensiunea V_{GC} devine mai mare decât o valoare V_{pn} (valoare de prag/ deschidere pentru nMOS) câmpul electric al condensatorului poartă-substrat atrage, din masa substratului, sarcini electrice minoritare mobile (electronii) determinând astfel în zona de interfaţă o inversiune a sarcinilor mobile, adică conductivitatea iniţială de tip p (goluri) devine de tip n. Electronii atraşi în zona de interfaţă vor constitui o sarcină mobilă în zona de canal, deci apare posibilitatea unui curent de canal. Cantitatea de sarcină poate fi exprimată prin relaţia:

$$Q = C(V_{GC} - V_{pn}) \tag{1.27}$$

În această expresie C este capacitatea echivalentă a condensatorului plan format între placă-substrat (cu suprafaţa $W \times L$, stratul dielectric având grosimea D_{ox} iar ε_{ox} permitivitatea oxidului) a cărei valoare este dată de relaţia clasică:

$$C = \varepsilon_{ox} \cdot \frac{W \cdot L}{D_{ox}} = C_{ox} \cdot W \cdot L \tag{1.28}$$

unde $C_{ox} = \frac{\varepsilon_{ox}}{D_{ox}}$ este capacitatea pe unitatea de suprafaţă a stratului de oxid de sub poartă, care se obţine din relaţia 1.28 când $W = 1$, $L = 1$. Cu valorile $\varepsilon_{ox} \approx 3,45 \times 10^{-11} F \cdot m^{-1}$, $D_{ox} = 100\text{Å} = 10^{-8} m$ se obţine $C_{ox} \approx 3fF/\mu m^2$.Capacitatea specifică a stratului de oxid este o caracteristică a fiecărui proces tehnologic şi trebuie luată de către proiectantul de circuite ca o constantă dată. Diferenţa de tensiune $V_{GC} - V_{pn}$ poate fi privită ca tensiunea efectivă care comandă sarcina din canal, deoarece până când tensiunea aplicată pe poartă nu este mai mare decât tensiunea de prag V_{pn} nu există sarcină mobilă în canal, deci nici curent în canal, tranzistorul este blocat.

Prin existenţa sarcinii induse, $V_{GC} - V_{pn} > 0$, şi prin polarizarea zonelor de dren şi sursă, în canal se produce deplasare de sarcină, deci curent. Polarizarea se face în felul următor: pe dren se aplică o tensiune pozitivă faţă de substrat $V_{DB} > 0$ iar sursa se conectează la substrat $V_{SB} = V_{SS} = 0V$; în felul acesta potenţialul sursei este identic cu potenţialul de substrat şi este potenţialul de referinţă (masă), iar V_{DB} devine V_{DS}. Într-o abordare simplă, pentru deducerea unui model matematic pentru curentul din canal, se echivalează canalul cu o rezistenţă pe care sursa de tensiune V_{GC} produce o cădere de tensiune graduală în funcţie de lungimea x din canal, $V_{GC}(x)$. Se admite pentru această cădere de tensiune o variaţie liniară în funcţie de lungimea din canal astfel: pentru $x = 0$, în zona de sursă, $V_{GC} = V_{GS}$ şi pentru $x = L$ (lungimea canalului) în zona de dren $V_{GC} = V_{GS} - V_{DS}$. Calculul sarcinii din canal se face cu relaţia 1.27 în care se introduce căderea de tensiune ca o medie aritmetică a valorilor tensiunilor de la capetele canalului:

$$Q = C\left[\frac{(V_{GC} - V_{pn})_{x=0} + (V_{GC} - V_{pn})_{x=L)}}{2}\right] = C\left[(V_{GS} - V_{pn}) - \frac{1}{2}V_{DS}\right] \tag{1.29}$$

şi utilizând expresia capacităţii exprimată prin relaţia 1.28 se obţine:

$$Q = C_{ox}WL\left[(V_{GS} - V_{pn}) - \frac{1}{2}V_{DS}\right] \tag{1.30}$$

Sub acţiunea câmpului electric din canal $E_x = -V_{DS}/L$ (se consideră numai componenta V_x în lungul canalului) produs de tensiunea V_{DS} electronii sunt deplasaţi de la sursă la dren cu viteza:

$$v_x = -\mu_n \cdot E_x = \mu_n \frac{V_{DS}}{L} \tag{1.31}$$

unde μ_n este mobilitatea electronilor, cu valori între $500 \div 1500 cm^2/V \cdot s$; pentru goluri valorile mobilităţilor sunt $\mu_p = 100 \div 400 cm^2/V \cdot s$. Mobilitatea purtătorilor scade cu concentraţia de dopare şi cu creşterea tempteraturii.

Timpul de deplasare/tranzit al electronilor prin canal rezultă simplu:

$$\tau_{\mathrm{f}} = \frac{L}{v_x} = \frac{L^2}{\mu_n \cdot V_{DS}} \tag{1.32}$$

Această relaţie exprimă o dependenţă pătratică a timpului de tranzit în canal în funcţie de lungimea canalului; apare evidenta concluzie că scalarea (micşorarea dimensiunilor) duce la dispozitive mai rapide. În final, din relaţiile 1.30 şi 1.32, se obţine expresia curentului dren - sursă, I_{DS}:

$$\begin{aligned} I_{DS} = \frac{Q}{\tau_{\mathrm{f}}} = \mu_n C_{ox} \frac{W}{L} \left[(V_{GS} - V_p) - \frac{1}{2} V_{DS} \right] V_{DS} = \\ = K_n \frac{W}{L} \left[(V_{GS} - V_p) - \frac{1}{2} V_{DS} \right] V_{DS} \end{aligned} \tag{1.33}$$

Constanta:

$$K_n = \mu_n C_{ox} \tag{1.34}$$

denumită factorul de câştig al procesului, la fel ca şi tensiunea de prag V_{pn}, este o constantă de proces şi are valori în intervalul $100 \div 100 \mu A/V^2$. Nu este neobişnuit ca în cadrul aceluiaşi proces, în funcţie de materialul iniţial utilizat ca substrat, de variaţia în grosime a stratului de oxid de sub poarta D_{ox}, factorul de câştig al procesului să aibă abateri de $10 \div 20\%$.

Se defineşte factorul de câştig al tranzistorului, β_n:

$$\beta_n = K_n \cdot \frac{W}{L} \tag{1.35}$$

Raportul dintre lăţimea canalului W şi lungimea canalului L este referit ca **factorul de formă al tranzistorului**, de fapt conductanţa canalului este proporţională cu raportul dintre lăţimea şi lungimea sa. **Raportul** W/L este unul din cei mai importanţi parametri asupra căruia proiectantul poate acţiona în etapa de dimensionare a layoutului pentru a obţine caracteristicile dorite pentru tranzistor.

Ecuaţia 1.33 este modelul matematic al tranzistorului MOS pentru regimul liniar de funcţionare. Acest regim liniar, între variaţiile ΔI_{DS} ale curentului de canal şi variaţiile ΔV_{GS} ale tensiunii grilă sursă, se obţine când sunt îndeplinite condiţiile regimului liniar: $V_{GS} > V_{pn}$ şi $V_{DS} < V_{GS} - V_{pn}$.

Când tensiunea dren-sursă creşte, peste valoarea efectivă a tensiunii de comandă a canalului, $V_{DS} > V_{GS} - V_{pn}$, rezultă o diferenţă de potenţial poartă-dren de valoare negativă $V_{GD} = (V_{GS} - V_p) - V_{DS} < 0$, deci se inversează sensul câmpului poartă-dren. Această inversare a sensului câmpului determină o reducere a sarcinii din canal în zona

de dren ($x = L$) ceea ce este echivalent cu creşterea rezistenţei canalului înspre dren, cea mai mare parte a tensiunii V_{DS} se va regăsi pe acest segment de rezistenţă mărită. Lungimea efectivă, notată cu L', a canalului se micşorează ($L' < L$) şi chiar la mărirea tensiunii dren-sursă, V_{DS}, curentul în canal rămâne la o valoare constantă, $I_{DS(sat)}$ (în realitate $I_{DS(sat)}$ are o creştere uşoară în funcţie de V_{DS}). Acesta este regimul de saturaţie al tranzistorului obţinut pentru condiţiile: $V_{GS} > V_{pn}$ şi $V_{DS} \geq V_{GS} - V_{pn}$; iar expresia curentului de saturaţie se obţine din ecuaţia regimului liniar 1.33 în care se face substituţia $V_{DS} = V_{GS} - V_{pn}$:

$$I_{DS(sat)} = \frac{\beta}{2}(V_{GS} - V_{pn})^2 \tag{1.36}$$

În Figura 1.32-b sunt reprezentate caracteristicile statice $I_{DS} = f(V_{DS})_{V_{GS}=\text{const}}$ pentru două tranzistoare nMOS unul cu canal lung $W/L = 60\mu m/6\mu m$ iar altul cu canal scurt $W/L = 6\mu m/0,6\mu m$ realizate în tehnologie de $0,5\mu m$. Se observă din Figura 1.32-c că ecuaţia 1.36 exprimă mult mai exact curentul de saturaţie $I_{DS(sat)}$ pentru tranzistoarele cu canal lung decât pentru cele cu canal scurt.

Separarea zonei liniare de zona de saturaţie în Figura 1.32-b este demarcată printr-o linie întreruptă, care este reprezentarea grafică a ecuaţiei $V_{DS} = V_{GS} - V_{pn}$. Regimul blocat al tranzistorului $I_{DS} = 0$, obţinut când $U_{GS} \leq V_{pn}$, este reprezentat de axa absciselor (în realitate, I_{DS} are valori foarte mici determinat în special de aşa-numitul curent de sub poartă). Tranzistorul se deschide când tensiunea de comandă V_{GS} devine $V_{GS} > V_{pn}$.

În scheme echivalente, în funcţie de regimul de funcţionare, tranzistorul MOS într-o abordare calitativă poate fi substituit în felul următor: regimul blocat - contact deschis, regimul liniar - rezistenţa, regimul saturat - generator de curent.

Modelul matematic al dependenţei curent-tensiune în curent continuu, $I_{DS} = f(V_{DS})_{U_{GS}=\text{const}}$, pentru tranzistorul MOS exprimat de ecuaţia 1.33, pentru regimul liniar şi exprimat de ecuaţia 1.36, pentru regimul în saturaţie, sunt relaţii simple, uşor de utilizat, dar rezultatele pot diferi de cele reale. De exemplu, calculând factorul de câştig al procesului K_n, cu ajutorul ecuaţiei 1.33, rezultă o valoare de două ori mai mică decât valoarea lui K_n măsurată pe tranzistor când este în regimul liniar; explicaţia constă în faptul că V_{pn} şi μ_n nu sunt constante cum se consideră în ecuaţie.

Un model matematic care să exprime exact funcţionarea tranzistorului este necesar atât proiectantului de tranzistoare cât şi proiectantului de circuite pe baza tranzistorului respectiv. Proiectantul de circuite doreşte ca prin simulări pe baza unui model matematic, în care introduce datele de catalog ale tranzistorului , să obţină caracteristicile dinamice la circuitului (τ_{pHL}, τ_{pLH}, τ_{HL}, τ_{LH}). Iar proiectantul de tranzistoare de la turnătoria de Si doreşte ca pe baza unui model în care să introducă valorile parametrilor de proces (V_{pn}, K_n, C_{ox}, D_{ox} etc.) să facă predicţia performanţelor tranzistorului înainte ca acesta să fie produs. Se pot obţine modele mult mai exacte dacă sunt luate în considerare efectele de ordinul doi (neglijate în deducerea ecuaţiilor 1.33 şi 1.36).

Programul de simulare **SPICE** (**S**imulation **P**rogram with **I**ntegrated **C**ircuit **E**mphasis) existent în toate mediile de proiectare automată în electronică, EDA,are la bază diferite modele matematice pentru tranzistorul MOS (simulatoarele comerciale pot avea până la zece modele matematice pentru tranzistor).

SPICE-ul poate simula pentru trei modele, acestea sunt selectabile prin identificatorul de nivel LEVELi. Nivelul LEVEL1 se bazează pe modelul matematic exprimat prin relaţiile 1.33, 1.36 în plus include şi efectele de ordinul doi importante. Nivelul LEVEL2 calculează curenţii pe baza unui model analitic al fizicii dispozitivului. Iar nivelul LEVEL3 este o abordare semi-empirică pe baza parametrilor de proces şi o "potrivire" a expresiei ecuaţiilor astfel încât acestea să se suprapună peste comportamentul real al tanzistorului. Evident, în nivelul 2 şi 3 sunt incluse toate efectele de ordinul doi.

În continuare vor fi expuse succint efectele de ordinul doi [Kang ′96][Weste ′92].

- Efectul de substrat. În Figura 1.32-a s-a considerat că sursa se leagă la substrat, $V_{SB} = V_{SS} = 0V$, deci potenţialul de referinţă este potenţialul substratului. Se poate ca faţă de potenţialul de referinţă (de masă) al sursei, V_{SS}, substratul să fie polarizat cu o tensiune de substrat $V_{SB} \neq 0V$. Polarizarea de substrat duce la o creştere a tensiunii de prag V_p (aproximativ cu $0,5V$ pentru fiecare creştere a V_{BS} cu 2 V). Creşterea tensiunii de prag V_p determină o micşorare a curentului I_{DS} ceea ce duce la un dispozitiv mai lent.

- Modulaţia lungimii canalului. Când $V_{DS} > V_{GS} - V_p$, adică în regim de saturaţie sarcinile din canal din zona de dren dispar, deci lungimea efectivă (electrică) L' a canalului devine mai mică decât lungimea geometrică L a canalului ($L' < L$). Apare astfel o modulaţie a lungimii canalului care este o funcţie de V_{DS}. Odată cu micşorarea lungimii canalului se modifică raportul W/L deci valoarea lui β (relaţia 1.35).La tranzistoarele cu canal lung această variaţie a lui β nu este importantă dar devine importantă la tranzistoarele cu canal scurt (deci odată cu scalarea).

- Variaţia mobilităţii. Mobilitatea atât a electronilor μ_n cât şi a golurilor μ_p nu este constantă, acestea scad cu creşterea concentraţiei de dopare cu impurităţi (deci cu scalarea) şi cu creţerea temperaturii.

- Saturaţia vitezei. Din relaţia 1.31 apare că viteza de drift a purtătorilor poate creşte oricât de mult în funcţie μ şi de E. În realitate viteza ajunge la o valoare de saturaţie, de exemplu viteza electronului ajunge la valoarea $v_{max} = 10^5 m/s$ care se atinge pentru $E = 10^6 V/m$ (o cădere de 1V pe lungimea de $1\mu m$) şi care nu se poate depăşi chiar dacă E_x creşte în continuare.

- Conducţia de subprag I_{OH}. În modul de aproximare graduală a canalului când $V_{GS} < V_p$ curentul $I_{DS} = 0$. Dar atunci când V_{DS} are valori ridicate chiar dacă $V_{GS} < V_p$ apare totuşi un curent de valoare relativ scăzută în canal deci $I_{DS} \neq 0$; care are următoarele două componente: un curent de polarizare inversă prin joncţiunile formate între zonele difuzate şi substrat; un curent de (conducţie) subprag. Odată cu scalarea dimensională (şi a tensiunii V_{DD}, scalare completă) trebuie micşorată şi valoarea tensiunii de prag V_p pentru a asigura viteză ridicată de comutaţie a tranzistorului şi o margine de zgomot suficientă. Curentul de subprag este proporţional cu $e^{-\frac{V_p}{T}}$, deci are o creştere odată cu micşorarea tensiunii de prag şi cu creşterea temperaturii (T).

- Injecţia de purtători fierbinţi. Când, prin scalare, dimensiunile tranzistorului sunt foarte mici şi V_{DS} este de valoare mare, componentele orizontale şi verticale

(E_x, E_y) ale câmpului în canal au valori ridicate sub a căror efect purtătorii mobili de sarcină pot atinge energii cintetice ridicate (devin purtători "fierbinţi"). Electronii fierbinţi pentru nMOS pot pătrunde: în stratul de oxid de sub poartă modificând distribuţia de sarcină de la interfaţa SiO_2-substrat sau în zona de dren dislocând goluri care sunt apoi atraşi înspre substrat. Efectul de injectie de purtători fierbinţi este dezastruoas mai ales pentru dispozitivele deja în funcţionare care, în timp, se distrug scoţând din funcţiune sistemul respectiv.

- Scalarea, care nu este un efect de ordinul doi pentru tranzistor dar, prin reducerea dimensiunilor şi a tensiunilor de alimentare, prin creşterea nivelurilor de dopare cu impurităţi donoare N_D sau acceptoare N_A crează condiţii ca efectele explicate mai sus să apară mult mai pronunţat.

Tabelul 1.10 Reducerea mărimii caracteristice (minime) la un proces tipic de matrice de porţi CMOS

ANUL	1980	1983	1985	1987	1989	1991	1993	1995	1997	1999	2001	2003
Marimea caractersitica minima [µm]	5.0	3.5	2.5	1.75	1.25	1	0.8	0.6	0.35	0.25	0.18	0.13

Reducerea dimensiunilor în tehnologia de integrare este referită prin termenul de **scalare**. Prin îmbunătăţiri, în timp, procesul tehnologic îşi micşorează caracterstica de proces (dimensiunea minimă care poate fi definită şi realizată pentru procesul respectiv); astfel s-a constatat că mărimea caracteristică (minimă) a procesului se scalează/reduce aproximativ cu $1,3 \div 1,5$ în decurs de 2-3 ani, cum rezultă din *Tabelul 1.10*. Scalarea este caracterizată prin factorul de scalare s, mărimea cu care se divid toate dimensiunile. Sunt referite două cazuri de scalare: scalarea completă şi scalarea la tensiune constantă.

Scalarea completă reduce toate dimensiunile cu factorul s, reduce tensiunile cu factorul s iar valorile densităţilor de dopare sunt multiplicate cu s, toate acestea pentru a menţine câmpul electric în dispozitiv nemodificat. Acest mod de scalare apare atractiv din punct de vedere al puterii disipate, *Tabelul 1.11*, care se reduce de s^2 ori, deşi puterea disipată pe unitatea de arie nu se modifică.

Scalarea la tensiune constantă reduce dimensiunile cu factorul s, măreşte densităţile de dopare cu s^2, dar tensiunile de alimentare rămân nemodificate. Acest tip de scalare este preferat, uneori, pentru a se putea interfaţa circuitul scalat cu alte circuite (anterioare) care nu au fost scalate (se utilizează aceeaşi valoare a tensiunii de alimentare). Se observă că puterea disipată creşte de s ori, puterea disipată pe unitatea de arie de s^3 ori, iar densitatea de curent j tot de s^3 ori. Creşterea densităţii de curent pe traseele metalice poate provoca, prin electromigraţie, întreruperea acestora. **Electromigraţia** metalului este efectul prin care, la densităţi foarte mari, într-un conductor atomii metalului sunt antrenaţi în sensul curentului. Prin scalare la tensiune constantă în unele puncte ale traseelor metalice pot apare îngustări, unde densitatea de curent depăşeşte $10^5 A/cm^2$ (pentru Aluminiu), iar în timp, prin electromigraţie, traseul se subţiază continuu până se întrerupe în acele puncte.

1.5.2 Inversorul CMOS

Inversorul bipolar, din Figura 1.21-b, este compus dintr-un element de comutaţie, tranzistorul T, şi o rezistentă de sarcină R_C. În aceeaşi idee, se poate realiza un circuit inversor la care un tranzistor nMOS este elementul de comutaţie iar sarcina un tranzistor pMOS. Dar la fel, tranzistorul pMOS poate fi privit ca un element de comutaţie într-un circuit inversor care are ca sarcină tranzistorul nMOS. Această structură de circuit, formată din două tranzistoare n şi pMOS înseriate în opoziţie, conectate între potenţialele V_{DD} şi V_{SS}, având un singur terminal de poartă, este referită ca inversor Complementar MOS, Figura 1.33-a. Cele două tranzistoare sunt realizate pe aceeaşi plachetă, Figura 1.35, dar au substraturi diferite, zonele de tip p^+ ale tranzistorului pMOS (sursa şi drenul) sunt implantate într-un substrat de tip n, care este de fapt o insulă difuzată în substratul de pe plachetă, iar zonele n^+ (sursa şi drenul) ale tranzistorului nMOS sunt implantate în substratul de tip p al plachetei. Avantajele pe care le prezintă inversorul CMOS în raport cu alte tipuri de inversoare sunt:

- puterea disipată, în regim static, teoretic este zero;
- caracteristica de transfer $V_O = f(V_I)$ se apropie cel mai mult de caracteristica ideală a unui inversor, Figura 1.14-b, cu tensiunea de prag (logic) de comutaţie la $V_{DD}/2$, vezi Definiţia 1.14.

Tabelul 1.11 Valorile unor parametrii electrici şi geometrici după scalare

Mărimea	Înainte de scalare	După scalare	
		Scalare completă	Scalare la tensiune constantă
Lungimea de canal	L	$L' = \frac{L}{s}$	$L' = \frac{L}{s}$
Lăţimea de canal	W	$W' = \frac{L}{s}$	$W = \frac{L}{s}$
Grosimea oxidului de sub poartă	D_{ox}	${D_{ox}}' = \frac{D_{ox}}{s}$	${D_{ox}}' = \frac{D_{ox}}{s}$
Tensiunea de alimentare	V_{DD}	${V_{DD}}' = \frac{V_{DD}}{s}$	${V_{DD}}' = V_{DD}$
Tensiunea de prag	V_p	${V_p}' = \frac{V_p}{s}$	${V_p}' = V_p$
Câmpul electric în oxid	E	$E' = E$	$E' = E \cdot s$
Concentraţia de impurităţi donoare	N_D	${N_D}' = N_D \cdot s$	${N_D}' = N_D \cdot s^2$
Concentraţia de impurităţi acceptoare	N_A	${N_A}' = N_A \cdot s$	${N_A}' = N_A \cdot s^2$
Capacitatea specifică a porţii	C_{ox}	${C_{ox}}' = C_{ox} \cdot s$	${C_{ox}}' = C_{ox} \cdot s$
Curentul de dren-sursă	I_{DS}	${I_{DS}}' = \frac{I_{DS}}{s}$	${I_{DS}}' = I_{DS} \cdot s$
Densitatea de curent	j	$j' = j$	$j' = j \cdot s^3$
Puterea disipată	P_d	${P_d}' = \frac{P_d}{s^2}$	$P_d = P_d \cdot s$
Puterea disipată/unitate de arie	$\frac{P_d}{arie}$	$\frac{{P_d}'}{arie'} = \frac{P_d}{arie}$	$\frac{{P_d}'}{arie'} = s^3(\frac{P_d}{arie})$

1.5.2.1 Caracteristica statică de transfer $V_O = f(V_I)$

În opoziţie faţă de tranzistorul nMOS (prezentat anterior), care are zonele difuzate de tip n^+ şi la care drenul se polarizează cu o tensiune pozitivă faţă de sursă iar tensiunea de deschidere $\boldsymbol{V_{pn}}$ **este pozitivă**, la tranzistorul pMOS zonele difuzate sunt de tip p^+, drenul se polarizează cu tensiune negativă faţă de sursă iar tensiunea de deschidere $\boldsymbol{V_{pp}}$ **este negativă**, Figura 1.33-b. În structura de inversor CMOS prin conectarea, pentru ambele tranzistoare, a zonelor de sursă la substraturile corespunzătoare nu se va lua în considerare efectul de polarizare a substratului, deci se pot deduce următoarele relaţii:

$$\begin{aligned} V_{GSn} &= V_I \\ V_{DSn} &= V_O \end{aligned} \tag{1.37}$$

şi de asemenea

$$\begin{aligned} V_{GSp} &= -(V_{DD} - V_I) \\ V_{DSp} &= -(V_{DD} - V_O) \end{aligned} \tag{1.38}$$

Caracteristica statică, Figura 1.33-c, se va analiza pentru variaţia tensiunii de intrare pornind de la $V_I = V_{SS} = 0V$ până la $V_I = V_{DD}$, iar ieşirea inversorului se consideră în gol, deci pentru curenţii celor două tranzistoare există relaţia:

$$I_{DSn} = -I_{DSp} \tag{1.39}$$

Regimul liniar de funcţionare este definit de relaţiile:

$$\begin{aligned} V_I > V_{pn} \quad \text{şi} \quad V_{DS_n} < V_I - V_{pn} &\text{pentru nMOS} \\ V_I < V_{DD} + V_{pp} \quad \text{şi} \quad V_{DS_p} > V_{GS_p} - V_{pp} &\text{ pentru pMOS} \end{aligned} \tag{1.40}$$

iar tranzistoarele sunt echivalente unor rezistenţe.

Regimul de funcţionare în saturaţie este definit de relaţiile:

$$\begin{aligned} V_I > V_{pn} \quad \text{şi} \quad V_{DS_n} \geq V_{GS_n} - V_{pn} \Longleftrightarrow V_O \geq V_I - V_{pn} &\text{ pentru nMOS} \\ V_I < V_{DD} + V_{pp} \quad \text{şi} \quad V_{DS_p} \leq V_{GS_p} - V_{pp} \Longleftrightarrow V_O \leq V_I - V_{pp} &\text{ pentru pMOS} \end{aligned} \tag{1.41}$$

iar tranzistoarele sunt echivalente unor generatoare de curent.

În caracteristica statică de transfer $V_O = f(V_I)$, în funcţie de regimul de funcţionare al fiecăruia din tranzistoarele nMOS şi pMOS se disting cinci regiuni de funcţionare (Ⓐ, Ⓑ, Ⓒ, Ⓓ, Ⓔ) pentru inversor. Pe această caracterstică se pot calcula valorile tensiunii de intrare în stare L, V_{IL}, şi de intrare în starea H, V_{IH}.

Regiunea **Ⓐ**. $0 \leq V_I < V_{pn}$ deci nMOS este blocat, $I_{DSn} = 0$ şi este echivalent unui contact deschis. Tranzistorul pMOS are $V_{GSp} < Vpp$ şi este în regiunea liniară cu $I_{DSp} = 0$, conform relaţiei 1.39. Fiind în regiunea liniară cu $I_{DSp} = 0$ rezultă că şi $V_{DSp} = 0$ ceea ce din relaţia a doua 1.38 determină tensiunea de ieşire în stare H:

$$V_O = V_{OH} = V_{DD} \tag{1.42}$$

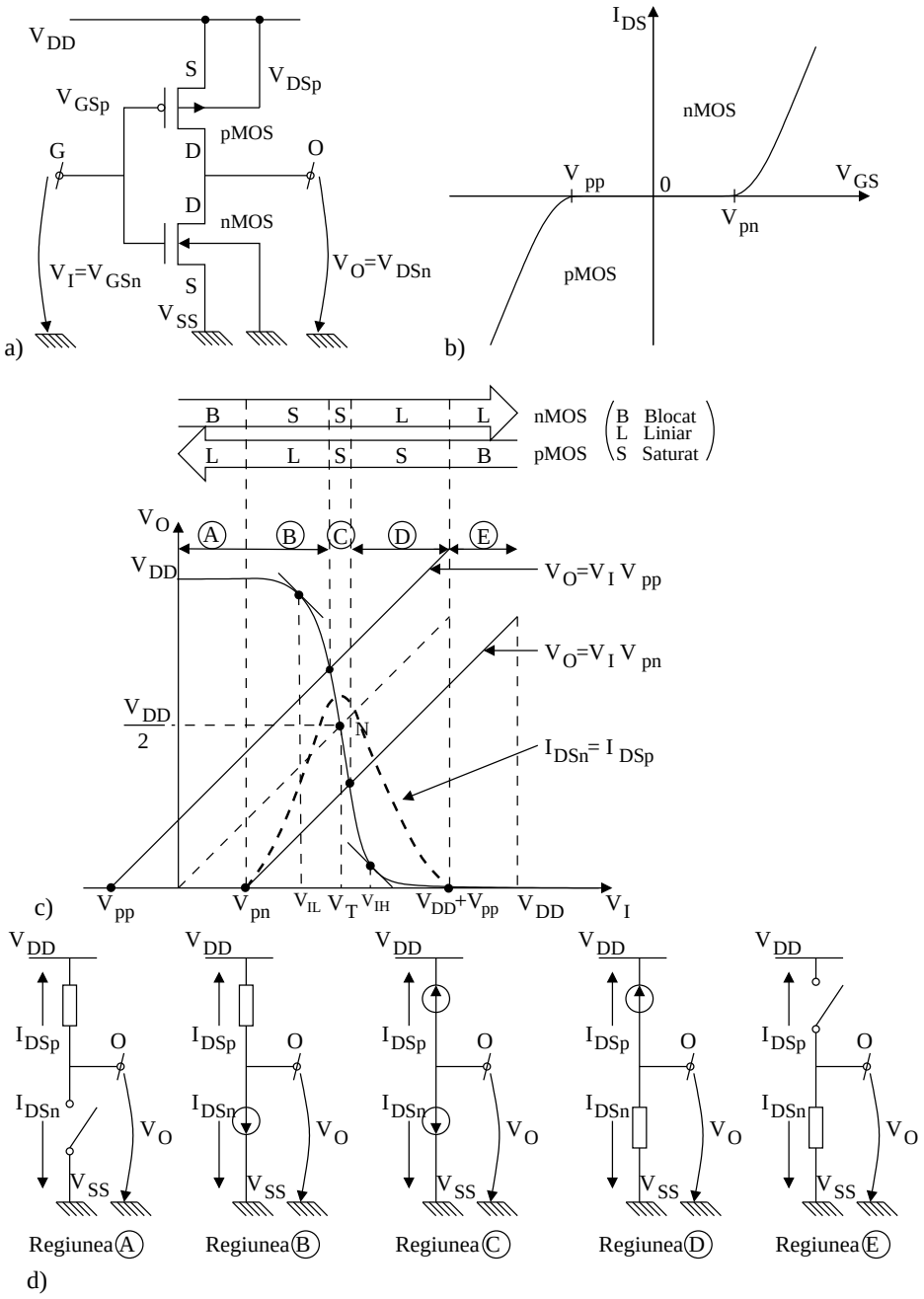


Figura 1.33 Inversorul CMOS: a) schema electrică; b) caracteristicile de comandă $I_{DS} = f(V_{GS})$ pentru nMOS şi pMOS; c) caracterstica statică $V_O = f(V_I)$; d) circuitele echivalente pentru cele cinci regiuni ale caracteristicii statice.

Regiunea (**B**). $V_I \geq V_{pn}$, $V_{DSn} \approx V_{DD} > V_I - V_{pn}$ deci nMOS intră în saturaţie iar pMOS continuă în regimul liniar. În caracteristica de transfer $V_O = f(V_I)$ tensiunea începe să scadă şi, prin definiţie, când panta caracteristicii de transfer este egală cu -1 ($dV_O/dV_I = -1$), se consideră pentru tensiunea de intrare $V_I = V_{IL}$. Introducând în relaţia 1.39 pentru nMOS exprimarea de regim de saturaţie relaţia 1.36, iar pentru pMOS exprimarea de regim liniar se obţine:

$$\frac{\beta_n}{2}(V_I - V_{pn})^2 = \frac{\beta_p}{2}\left[2\left((V_I - V_{DD}) - V_{PP}\right)(V_O - V_{DD}) - (V_O - V_{DD})^2\right] \quad (1.43)$$

Din această expresie, prin derivarea lui V_O în raport cu V_I şi apoi efectuarea substituirilor $dV_O/dV_I = -1$, $V_I = V_{IL}$, se obţine expresia pentru V_{IL}:

$$V_{IL} = \frac{2V_O + V_{pp} - V_{DD} + \frac{\beta_n}{\beta_p}V_{pn}}{1 + \frac{\beta_n}{\beta_p}} \quad (1.44)$$

Prin rezolavarea ecuaţiilor 1.44 şi 1.43 se obţine valoarea lui V_O pentru V_{IL} în punctul caracteristicii cu panta -1.

Regiunea (**C**). Tranzistorul nMOS rămâne în saturaţie şi după punctul de intersecţie al caracteristicii $V_O = f(V_I)$ cu dreapta $V_O = V_I - V_{pp}$ când intră în saturaţie şi tranzistorul pMOS, $V_O \leq V_I - V_{pp}$ (relaţia 1.41). Teoretic, în această regiune caracteristica ar trebui să fie verticală dar pentru că tranzistoarele în saturaţie nu sunt generatoare de curent ideale, $I_{DS(sat)}$ are o mică creştere cu V_{DS}, această caracteristica are o mică abatere de la o pantă infinită. În această regiune pentru variaţii mici ΔV_I se obţin variaţii mari ΔV_O, există un punct când tensiunea V_I care creşte devine egală cu tensiunea V_O care descreşte şi corespunde punctului (N) de intersecţie al caracteristicii cu prima bisectoare (vezi şi Figura 1.14-b).

Definiţia 1.14 Pragul (logic) de comutaţie V_T al unei porţi este aceea valoare a tensiunii de intrare care produce la ieşire o tensiune de valoare egală, adică $V_T = V_I = V_O$. $\diamond$

Prin introducerea expresiilor curenţilor de saturaţie în relaţia 1.39 se obţine:

$$\frac{\beta_n}{2}(V_I - V_{pn})^2 = \frac{\beta_p}{2}\left[(V_I - V_{DD}) - V_{pp}\right]^2 \quad (1.45)$$

şi prin substituţia $V_I = V_T$ rezultă

$$V_T = \frac{V_{pn} + \sqrt{\frac{\beta_p}{\beta_n}}(V_{DD} + V_{pp})}{1 + \sqrt{\frac{\beta_p}{\beta_n}}} \quad (1.46)$$

Regiunea (**D**). Când caracteristica de ieşire intersectează dreapta $V_O = V_I - V_{pn}$, sub care $V_O < V_I - V_{pn}$, tranzistorul nMOS intră în zona liniară de funcţionare iar pMOS rămâne în saturaţie. În caracteristica de transfer tensiunea V_O începe să nu mai descrească puternic şi, prin definiţie, când panta are valoarea $\frac{dV_O}{dV_I} = -1$, se consideră pentru tensiunea de intrare $V_I = V_{IH}$. În mod asemănător, ca în regiunea (**B**), şi

aici, egalând expresia curentului de saturaţie prin canalul p cu expresia în regim liniar din canalul n se obţine:

$$\frac{\beta_n}{2}\left[2(V_I - V_{pn})V_o - V_o^2\right] = \frac{\beta_p}{2}\left[(V_I - V_{DD}) - V_{pp}\right]^2 \tag{1.47}$$

în care substituind $V_I = V_{IH}$ şi $dV_O/dV_I = -1$ rezultă expresia pentru V_{IH} :

$$V_{IH} = \frac{V_{DD} + V_{pp} + \frac{\beta_n}{\beta_p}(2V_O + V_{pn})}{1 + \frac{\beta_n}{\beta_p}} \tag{1.48}$$

Prin rezolvarea sistemului de ecuaţii 1.47 şi 1.48 se obţin valorile pentru V_{IH} şi V_O în punctul caracteristicii cu panta -1.

Regiunea (**E**). $V_I \geq V_{DD} - V_{pp}$ deci pMOS este blocat, $I_{DS_n} = 0$ şi este echivalent unui contact deschis. Tranzistorul nMOS are $V_{GS} > V_{pn}$ şi este în regiunea liniară cu $I_{DS_n} = 0$, conform relaţiei 1.39. Fiind în regiunea liniară cu $I_{DS_n} = 0$ rezultă că şi $V_{DS_n} = 0$ adică :

$$V_O = V_{OL} = 0 \tag{1.49}$$

1.5.2.2 Proiectarea inversorului CMOS

Proiectarea, sau sinteza, unui inversor CMOS constă în determinarea dimensiunilor geometrice ale canalelor $(W/L)_n$, $(W/L)_p$ necesare realizării layoutului, pornind de la parametrii caracteristici de proces (V_{pn}, V_{pp}, D_{ox}, C_{ox} etc.), astfel încât să se obţină o caracteristică de transfer $V_O = f(V_I)$ şi anumite performanţe dinamice cerute. Poate fi parcurs şi traseul invers, adică analiza, pentru un inversor deja realizat, într-un anumit proces şi cu un anumit layout, să se determine caractersitica de proces şi performanţele dinamice (eventual acestea să fie comparate cu valorile obţinute experimental).

Fiind date V_{DD}, V_{pn} şi V_{pp} şi impusă o anumită valoare a tensiunii de prag de comutaţie V_T din relaţia 1.46 se deduce expresia pentru raportul β_n/β_p:

$$\frac{\beta_n}{\beta_p} = \left(\frac{V_{DD} + V_{pp} - V_T}{V_T - V_{pn}}\right)^2 \tag{1.50}$$

Considerând că inversorul este ideal, adică are tensiunea de prag logic de comutaţie la jumătatea tensiunii de alimentare, ca în Figura 1.33-c,

$$V_{T_{ideal}} = \frac{1}{2}V_{DD} \tag{1.51}$$

se obţine:

$$\left(\frac{\beta_n}{\beta_p}\right)_{ideal} = \left(\frac{0,5V_{DD} + V_{pp}}{0,5V_{DD} - V_{pn}}\right)^2 \tag{1.52}$$

Caracteristica statică $V_O = f(V_I)$ poate fi simetrică dacă tensiunile de prag ale tranzistoarelor sunt egale în valoare absolută $V_{pn} = |V_{pp}|$. Astfel din relaţia 1.52 se obţine valoarea raportului β_n/β_p pentru inversorul ideal şi cu o caracteristică simetrică:

$$\frac{\beta_n}{\beta_p} = 1 \tag{1.53}$$

Expresia explicită a raportului β_n/β_p este :

$$\frac{\beta_n}{\beta_p} = \frac{\mu_n \cdot C_{ox} \left(\frac{W}{L}\right)_n}{\mu_p \cdot C_{ox} \left(\frac{W}{L}\right)_p} = \frac{\mu_n \left(\frac{W}{L}\right)_n}{\mu_p \left(\frac{W}{L}\right)_p} \tag{1.54}$$

în care s-a considerat că grosimea stratului de oxid de sub poartă D_{ox}, în consecinţă şi $C_{ox} = \varepsilon_{ox}/D_{ox}$, au aceleaşi valori pentru ambele tranzistoare. Mobilitatea purtătorilor, mai mare cam de două ori a electronilor faţă de cea a golurilor, scade odată cu doparea cu impurităţi a substratului şi cu creşterea temperaturii. Luând valori tipice pentru mobilităţi $\mu_n = 580cm^2/V \cdot s$, $\mu_p = 230cm^2/V \cdot s$ şi introduse în raportul unitar din relatia 1.54 se obţine:

$$\frac{\left(\frac{W}{L}\right)_n}{\left(\frac{W}{L}\right)_p} = \frac{\mu_p}{\mu_n} = \frac{230cm^2/V \cdot s}{580cm^2/V \cdot s} \tag{1.55}$$

Rezultă relaţia cantitativă între coeficienţii de formă ai celor două canale:

$$\left(\frac{W}{L}\right)_p \approx 2,5 \left(\frac{W}{L}\right)_n \tag{1.56}$$

iar în cazul când se realizează aceeaşi lungime pentru ambele canale, $L_n = L_p$, rezultă relaţia între lăţimea canalelor $W_p = 2,5W_n$.

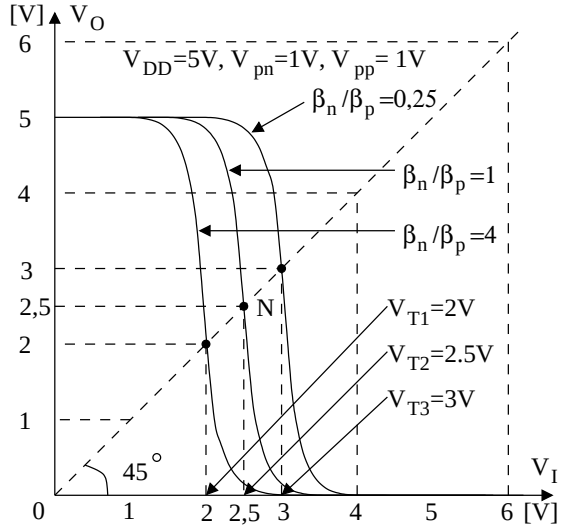


Figura 1.34 Dependenţa tensiunii de prag (logic) de comutaţie a inversorului în funcţie de valoarea raportului β_n/β_p.

Din relaţia 1.50 se observă că există o dependenţă între raportul β_n/β_p şi valoarea tensiunii de prag logic de comutaţie a inversorului. Pentru a obţine caracteristici statice cu prag de comutaţie V_T mai ridicat trebuie micşorată lăţimea W_n a canalului n în raport cu lăţimea W_p a canalului p (considerând că lungimile de canal rămân neschimbate), Figura 1.34.

Pentru inversorul ideal şi caracteristica simetrică ($\beta_n/\beta_p = 1$, $V_{pn} = |V_{pp}|$) din relaţia 1.44 se obţine expresia pentru V_{IL}:

$$V_{IL} = \frac{1}{8}\left(3V_{DD} - 2V_{pn}\right) \tag{1.57}$$

iar din relaţia 1.49 relaţia pentru V_{IH}:

$$V_{IH} = \frac{1}{8}\left(5V_{DD} - 2V_{pn}\right) \tag{1.58}$$

iar suma lor rezultă

$$V_{IH} + V_{IL} = V_{DD} \tag{1.59}$$

deci într-un inversor simetric suma $V_{IH} + V_{IL}$ este constantă.

Se pot calcula marginile de zgomot în curent continuu M_L, M_H:

$$\begin{aligned} M_L &= V_{IL} - V_{OL} \\ M_H &= V_{OH} - V_{IH} = V_{DD} - V_{IH} \end{aligned} \tag{1.60}$$

care sunt egale :

$$M_H = M_L = V_{IL} \tag{1.61}$$

Exemplul 1.15 Pentru inversorul CMOS realizat cu următorii parametrii: $V_{DD} = 5V$, $V_{pn} = 1V$, $V_{pp} = -1,2V$, $\beta_n = 100\mu A/V^2$, $\beta_p = 40\mu A/V^2$ să se calculeze valorile pentru marginile de zgomot în curent continuu M_H, M_L.

Soluţie. Problema nu este de sinteză ci este de analiză. Se observă că nu este un inversor simetric $V_{pn} \neq |V_{pp}|$; $\beta_n/\beta_p = 2,5$.

Valorile pentru $V_{OL} = 0V$, $V_{OH} = V_{DD} = 5V$ rezultă din relaţiile 1.42 şi 1.49 iar cele pentru V_{IL} şi V_{IH} se calculează în felul următor: din relaţia 1.44 rezultă V_{IL}

$$V_{IL} = 0,57V_O - 1,06$$

prin introducerea expresiei lui V_{IL} în relaţia 1.43 se obţine:

$$2,5(0,57V_O - 1,06 - 1)^2 = 2(0,57V_O - 1,06 - 5 + 1,2)(V_O - 5) - (V_O - 5)^2$$

$$0,66V_O^2 - 0,46V_O - 13 = 0$$

Numai soluţia pozitivă a ecuaţiei corespunde fizic problemei ($V_O > 0$) $V_O = 4,8V$ cu care se calculează $V_{IL} = 0,57V_O - 1,06 = 1,68V$.

Din relaţia 1.49 rezultă V_{IH}

$$V_{IH} = 1,43V_O + 1,8$$

prin introducerea expresiei V_{IH} în relaţia 1.47 se obţine:

$$(1,43V_O - 2)^2 = 2,5\left[2(1,43V_O + 1,8 - 1)V_O - V_O^2\right]$$

$$2,61V_O^2 + 9,72V_O - 4 = 0$$

La fel, numai soluţia pozitivă a ecuaţiei corespunde fizic problemei $V_O = 0,37V$ cu care se calculează $V_{IH} = 1,43 \cdot 0,37 + 1,8 = 2,33V$.

Şi ia în final :

$$M_L = V_{IL} - V_{OL} = 1,68 - 0 = 1,68V$$
$$M_H = V_{OH} - V_{IH} = 5 - 2,33 = 2,67V$$

Pe intervalul $V_{pn} \leq V_I \leq V_{DD} + V_{pp}$ se observă din Figura 1.33-c că ambele tranzistoare conduc, deci există un curent de scurtcircuit între V_{DD} şi V_{SS}. Durata acestor impulsuri de curent de scurtcircuit, curba $I_{DSn} = -I_{DSp}$ trasată punctat în Figura 1.33-c, depinde de durata de excursie a tensiunii de intrare în sens crescător între valorile V_{pn} şi $V_{DD} + V_{pp}$ şi în sens descrescător. Durata de excursie este cu atât mai scurtă cu cât panta fronturilor semnalelor de comandă, aplicate pe poartă, este mai mare.

Puterea disipată P_d la inversorul CMOS are cele trei componente explicate în secţiunea 1.3.

$$P_d = P_{dcc} + P_{dca} = P_{dcc} + P_{dsc} + P_{dc} \tag{1.62}$$

În regim static, regiunile (**A**) şi (**E**) din caracteristica statică, Figura 1.33-c, teoretic, puterea disipată P_{dcc} este zero, deoarece fie tranzistorul nMOS, fie pMOS sunt blocate, deci $I_{DSn} = -I_{DSp} = 0$. În realitate există o putere disipată de valoare redusă datorită unui curent rezidual I_{DS} $(= I_{DDQ})$ care are două cauze. Prima, când $V_{GS} < |V_p|$ există un curent în canal de ordinul μA datorită conducţiei de sub prag I_{off}. A doua, prin joncţiunile polarizate invers, formate între zonele difuzate n^+ şi substratul de tip p la nMOS şi zonele difuzate p^+ şi substratul de tip n la pMOS, există un curent de conducţie spre substrat. Acest curent rezidual, notat prin I_{DDQ} - curentul între V_{DD} şi V_{SS} în regim staţionar, poate fi utilizat pentru o primă metodă simplă de testare a unui circuit integrat CMOS. Valoarea normală a lui I_{DDQ} se poate estima sau se poate măsura la un circuit (verificat) care are o funcţionare normală. Dacă la un circuit prin măsurare se determină, în regim staţionar, pentru curentul absorbit de la sursa de alimentare o valoare mai mare decât cea normală a lui I_{DDQ} atunci acel circuit prezintă o cale de scurtcircuit de la V_{DD} la V_{SS}, deci este defect.

Puterea disipată în regim dinamic P_{dca} apare pe durata când inversorul are punctul de funcţionare în regiunile (**B**), (**C**), (**D**), Figura 1.33-c, ca o putere de scurtcircuit P_{dsc} şi ca o putere consumată pentru încărcarea şi descărcarea condensatoarelor interne şi de sarcină, P_{dc}. Într-o schemă echivalentă, similară celei din Figura 1.18, toate capacităţile sunt incluse într-o singură capacitate de sarcină C_L conectată la ieşire, iar puterea disipată pe această capacitate echivalentă, conform relaţiei 1.24, este egală cu :

$$P_{dca} = C_L V_{DD}^2 f \tag{1.63-a}$$

Reducerea puterii disipate se poate realiza prin reducerea oricăruia din factorii produsului din relaţia 1.63-a. Cea mai indicată modalitate de reducere a puterii este prin scalare completă, *Tabelul 1.11* (actual, s-a ajuns la tensiuni de alimentare de $V_{DD} = 0,8V$).

Puterea disipată de scurtcircuit P_{dsc}, apare, ca şi P_{dc}, pe durata tranziţiilor de la $H-L$ şi $L-H$ când V_{DD} este scurtcircuitată la V_{SS}. Reducerea valorii medii a puterii P_{dsc} se poate obţine prin comanda inversorului cu semnale cu fronturi bine formate

(abrupte). Această componentă P_{dsc} de putere disipată este greu de calculat, practic se estimează ca un procent din P_{dc} (uzual $< 0,2P_{dc}$), în consecinţă se consideră că puterea totală disipată în regim dinamic este $P_{dca} \approx 1,2P_{dc}$.

O relaţie utilizată cu succes în evaluarea puterii disipate P_d este:

$$P_d = \alpha C_L V_{DD}^2 f + I_{off} V_{DD} \tag{1.63-b}$$

care ţine cont şi de componenta de putere disipată în curent continuu produsă de curentul de subprag $P_{dcc} = I_{off}V_{DD}$; la dimensiuni sub $0,1\mu m$ şi $V_{DD} < 1,8V$ curentul I_{off} are o creştere pronunţată. α -este coeficientul (mediu) al activităţii de comutaţie (procentajul de timp în care dispozitivul este în funcţiune).

Micşorarea frecvenţei nu este o cale de reducere a puterii disipate deoarece se opune tendinţei de creştere a vitezei sistemelor. Dar printr-o analiză a sistemului se pot identifica anumite componente care pot funcţiona şi la frecvenţe mai mici decât frecvenţa maximă fără a reduce din performanţele de viteză ale sistemului. În plus, uneori, chiar componentele care funcţionează la frecvenţe ridicate pot fi oprite pe anumite intervale de timp $(1 - \alpha)$.

1.5.2.3 Tehnologia de fabricaţie a inversorului CMOS

Tehnologia de fabricaţie a inversorului CMOS, pentru explicaţie, poate fi considerată o extensie a tehnologiei de realizare a tranzistorului MOS, Figura 1.31. Dificultatea care apare acum constă în faptul că, de data aceasta, cele două tranzistoare complementare nMOS şi pMOS ar trebui implementate în acelaşi substrat. Soluţia pentru această incompatibilitate, de realizare în acelaşi substrat, este difuzarea în substratul iniţial a unei insule (well) care va constitui un al doilea substrat şi astfel unul din tranzistoare se implementează în substratul iniţial iar celălalt în al doilea substrat (insulă). Dacă substratul iniţial este de tip p se difuzează o zonă (insulă) de tip n în care va fi implementat tranzistorul pMOS, ca în Figura 1.35-b, iar dacă substratul iniţial este de tip n se difuzează o zonă de tip p în care va fi implementat tranzistorul nMOS.

După realizarea insulei de substrat, etapele de fabricare a fiecărui tranzistor sunt cele descrise în secţiunea Tehnologia de fabricaţie a tranzistorului MOS (1.5.1.1) cu diferenţa că atunci când se difuzează sursa şi drenul tranzistorului nMOS se difuzează şi o zonă n^+ de contact V_{DD} în fereastra pentru tranzistorul pMOS. Iar atunci când se difuzează sursa şi drenul tranzistorului pMOS se difuzează în plus şi o zonă p^+ de contact V_{SS} în fereastra pentru tranzistorului nMOS. Aceste difuzii de contact (cu V_{SS} la tranzistorul nMOS şi cu V_{DD} la tranzistorul pMOS) nu formează o joncţiune cu substratul respectiv, de aceeaşi conductivitate, ci un contact ohmic. Apoi, pe deasupra stratului gros de oxid de câmp se realizează prin trasee metalice conexiunile între :

1- zona de dren nMOS cu zona de dren pMOS care constituie ieşirea inversorului;

2- sursa nMOS cu zona p^+ de contact V_{SS} şi împreună la trasa pentru potenţialul V_{SS};

3- sursa pMOS cu zona n^+ de contact V_{DD} şi împreună la trasa pentru potenţialul V_{DD}.

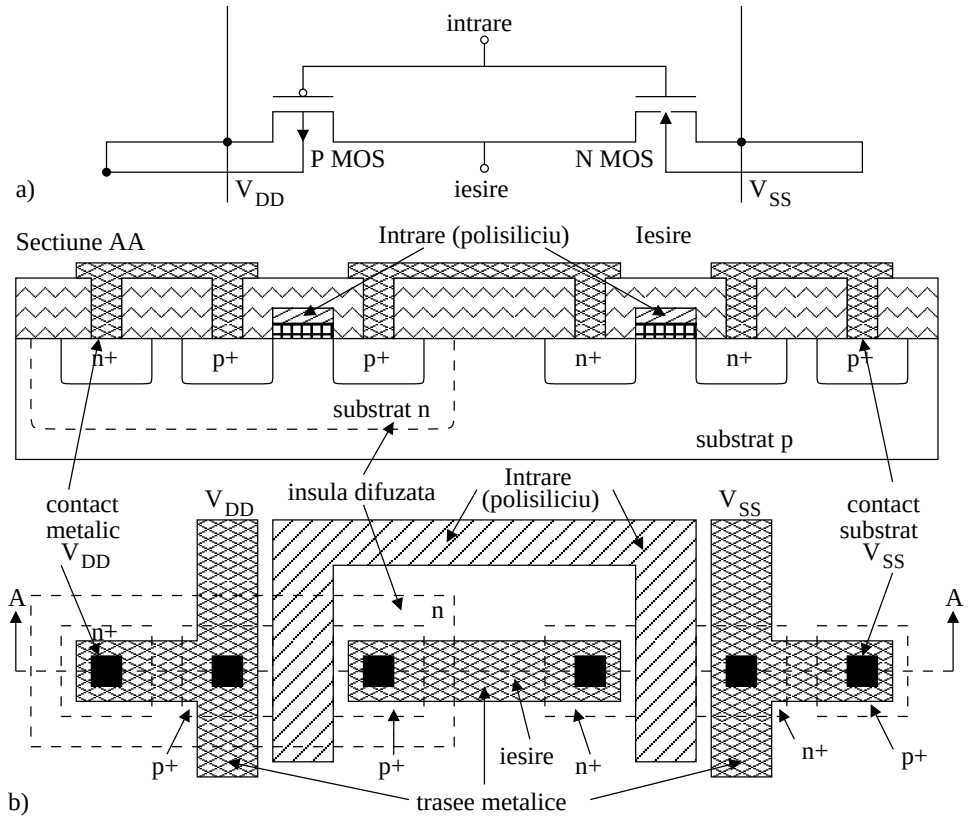


Figura 1.35 Inversorul CMOS: a) schema electrică; b) layout-ul pe substratul de Si şi secţiunea verticală AA în substrat (tranzistorul pMOS este realizat în substratul (insula) de tip n).

De asemenea, înainte de difuziile de dren şi sursă de tip n^+ şi de tip p^+, pentru cele două tranzistoare, se realizează din polisiliciu traseul pentru poarta comună a inversorului care va constitui intrarea.

Problema pe care o are proiectantul în siliciu este de a transforma schema electrică a circuitului în layout. Dar cum se poate face simplu această trecere? Liniile între terminalele tranzistoarelor de pe schema electrică vor reprezenta trasee metalice pe layout, aceste trasee metalice au contacte metalizate la cele două capete ca terminale. La două linii de conexiune în desenul electric, care nu trebuie să se intersecteze, le corespund două trasee, în planuri diferite, de metalizare.

Un exemplu simplu de trecere de la schema electrică la layout este dat în Figura 1.36 pentru un inversor CMOS. Linia de conexiune între terminalele de dren ale canalelor n şi p de pe schema electrică, Figura 1.36-a, este substituită cu o trasă metalică pe suprafaţa superioară a oxidului de câmp evident, cu contacte la cele două capete (pătratele înegrite de la capetele trasei metalice), Figura 1.36-b. La fel, conexiunea zonelor de sursă la liniile de V_{SS} şi V_{DD} se realizează prin trase metalice, în

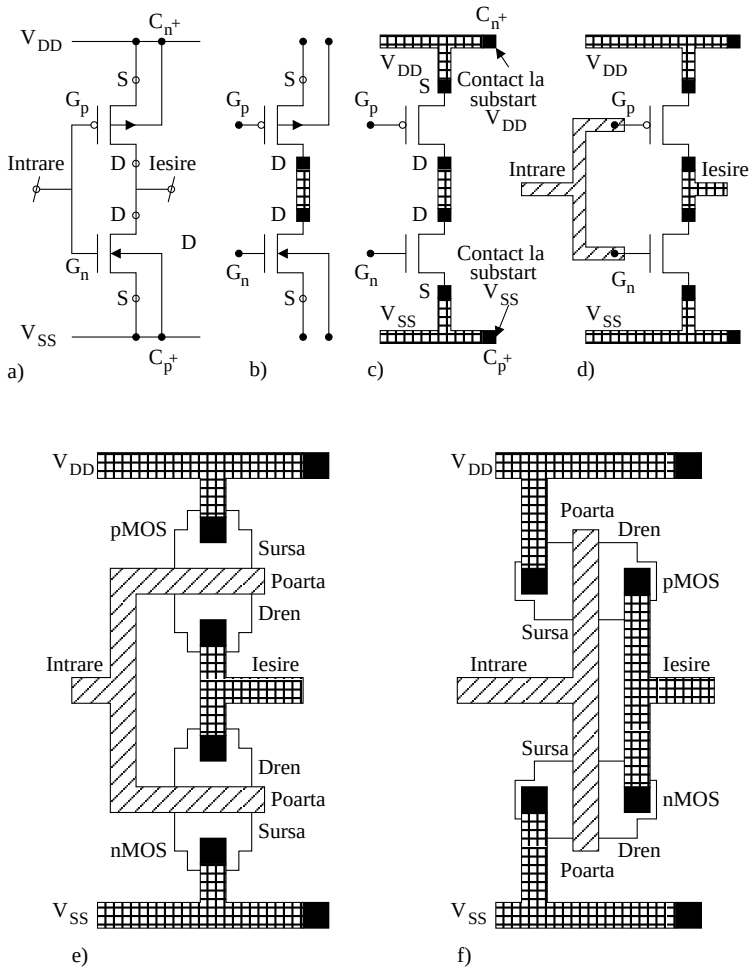


Figura 1.36 Succesiunea etapelor de transformare a schemei electrice în layout pentru un inversor CMOS

acelaşi timp se conectează şi contactele de substrat p^+ şi n^+ respectiv la trasele V_{SS} şi V_{DD}, Figura 1.36-c. Şi în final, conexiunea comună de poartă este substituită cu trasa de siliciu policristalin, Figura 1.36-d. Layout-ul este complet dacă şi simbolurile de tranzistoare sunt substituite cu geometria acestora pe siliciu, Figura 1.36-e şi 1.36-f (s-au realizat două variante de layout pentru tranzistoarele inversorului CMOS). Pentru circuite mai complexe există reguli de trecere care generează părţi de layout cu un grad mare de repetabilitate (repetabilitatea este o caracteristică dorită în obţinerea unui layout deoarece determină un cost mai scăzut şi poate duce la un circuit cu fiabilitate mai ridicată).

1.5.2.4 Regimul dinamic al inversorului

Pe lângă nivelurile de tensiune, definite în regimul static pe caracteristica $V_O = f(V_I)$, în practica sistemelor digitale sunt necesari şi parametrii de timp definiţi în regimul dinamic. Parametrii de timp pentru semnalul de ieşire: timpul de creştere τ_{LH} şi timpul de de descreştere τ_{HL} sunt definiţi ca în Figura 1.15-a iar timpul de propagare $H - L$, τ_{pHL}, şi timpul de propagare $L - H$, τ_{pHL}, sunt definiţi ca în Figura 1.15-b. Timpul de propagare pe un nivel inversor se calculează cu relaţia 1.20, $\tau_{\text{p}} = (\tau_{\text{pHL}} + \tau_{\text{p LH}})/2$.

Se consideră două inversoare CMOS înseriate cu reprezentarea tuturor capacităţilor parazite ale fiecărui tranzistor, Figura 1.37-a. În această reprezentare C_{GD} şi C_{GS} sunt capacităţile grilă-dren şi grilă-sursă datorate suprapunerii parţiale a trasei de grilă peste zonele difuzate dren şi sursă, C_{DB} şi C_{SB} sunt capacităţile dependente de tensiune ale joncţiunilor dren-substrat şi sursă substrat, C_{GB} capacitatea grilă-substrat este capacitatea stratului de oxid de sub poarta fiecărui tranzistor iar C_{cox} este capacitatea datorită conexiunilor (traselor) în polisiliciu sau metalice între ieşirea primului inversor şi intrarea următorului inversor. Calculul tensiunii de ieşire v_O, luând în considerare toate aceste capacităţi, devine foarte complicat chiar şi pentru un circuit simplu. De foarte multe ori, în practică, pentru simplificare calculului, toate aceste capacităţi sunt înglobate într-o singură capacitate echivalentă C_L, considerată ca o capacitate de sarcină, conectată la iesirea inversorului, Figura 1.37-b, conform relaţiei:

$$C_L = C_{GD_n} + C_{GD_p} + C_{DB_n} + C_{DB_p} + C_{cox} + C_{GB} \tag{1.64}$$

În capacitatea de sarcină echivalentă C_L nu sunt incluse C_{SBn} şi C_{SBp} deoarece ambele surse sunt conectate la substraturile corespunzătoare deci nu au un efect în regim dinamic. De asemenea, nu sunt incluse în C_L capacităţile C_{GSn} şi C_{GSp} deoarece acestea sunt conectate între nodul de intrare şi masă respectiv între nodul de intrare şi V_{DD}.

Studiul regimului dinamic se face pe circuitul echivalent din Figura 1.37-b aplicând la intrare un semnal cu variaţia dreptunghiulară v_I şi determinând variaţia în timp a tensiunii de ieşire v_O. Pentru variaţia $0 \rightarrow 1$ la intrare, condensatorul C_L încărcat la tensiunea V_{DD} se va descărca prin tranzistorul nMOS (pMOS este blocat), tensiunea v_O având o variaţie exponenţială de la V_{OH} la V_{OL}. Din variaţia în timp a tensiunii v_O se pot determina parametrii de timp τ_{HL} şi τ_{pHL}. Iar pentru variaţia $1 \rightarrow 0$ la intrare, C_L se încarcă până la tensiunea V_{DD} prin tranzistorul pMOS (nMOS este blocat), tensiunea v_O având o variaţie exponenţială de la V_{OL} la V_{OH}; din variaţia în

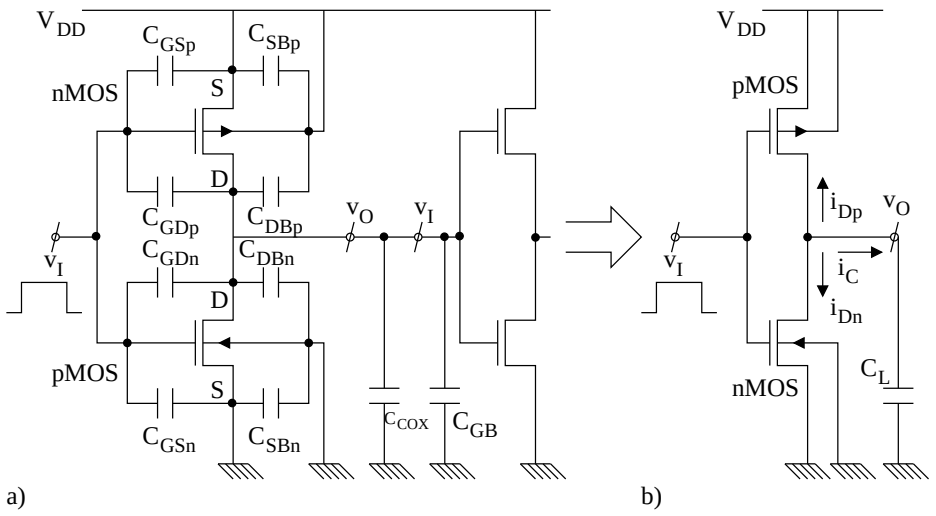


Figura 1.37 Înserierea a două inversoare CMOS: a) reprezentarea capacităţilor interne (parazite) pentru un inversor; b) schemă echivalentă simplificată .

timp a tensiunii v_O se pot determina parametrii de timp τ_{LH} şi τ_{pLH}. Pentru calculul acestor parametri de timp se vor expune succint următoarele patru metode:

1- Simulare în SPICE;

2- Metoda curentului constant;

3- Metoda analitică;

4- Metoda empirică.

1. Simularea în SPICE (NIVEL2 şi NIVEL3, vezi 1.5.1.2 şi exemplul de simulare, ANEXA1 din vol. II) se poate face pe baza circuitului din Figura 1.37-a, cu luarea în considerare a tuturor efectelor de ordinul al doilea în modelul matematic $I_{DS} = f(V_{DS}, V_{GS})$ al tranzistorului şi a parametrilor de proces, în consecinţă valorile obţinute pentru parametrii de timp au abateri minime faţă de cele reale. În practică, de multe ori, pentru determinări fără precizie ridicată dar efectuate simplu şi rapid, sunt utilizate una din celelalte (următoare) trei metode.

2. Metoda curentului constant. Această metodă, după cum şi denumirea indică, consideră valori constante pentru curentul de încărcare şi descărcare ale condensatorului C_L, aproximate prin valori medii I_{mHL} şi I_{mLH}. În consecinţă, cu aceste valori medii, se pot calcula simplu parametrii τ_{pHL} şi τ_{pLH} în felul următor:

$$\tau_{\text{pHL}} = \frac{C_L \cdot \Delta V_{HL}}{I_{mHL}} = \frac{C_L \cdot (V_{OH} - V_{50\%})}{I_{mHL}} \tag{1.65}$$

$$\tau_{\text{pLH}} = \frac{C_L \cdot \Delta V_{LH}}{I_{mLH}} = \frac{C_L \cdot (V_{50\%} - V_{OL})}{I_{mLH}} \tag{1.66}$$

Iar valorile I_{mHL} şi I_{mLH} se calculează ca medie aritmetică ale curenţilor prin tranzistoarele nMOS respectiv pMOS în punctele de început şi de sfârşit ale tranziţiei:

$$I_{mHL} = \frac{1}{2}\left[i_C(\text{pentru } v_I = V_{OH}, v_O = V_{OH}) + i_C(\text{pentru } v_I = V_{OH}, v_O = V_{50\%})\right] \tag{1.67}$$

$$I_{mLH} = \frac{1}{2}\left[i_C(\text{pentru } v_I = V_{OL}, v_O = V_{50\%}) + i_C(\text{pentru } v_I = V_{OL}, v_O = V_{OL})\right] \tag{1.68}$$

3. Metoda analitică. Variaţia tensiunii v_O în timp poate fi determinată prin rezolvarea ecuaţiei de stare pentru nodul de ieşire al circuitului din Figura 1.37-b

$$C_L \cdot \frac{dv_O}{dt} = i_C = i_{D_p} - i_{D_n} \tag{1.69}$$

în care pentru i_{D_p} şi i_{D_n} sunt considerate exprimările date prin relaţiile 1.33 şi 1.36. La excursia $H - L$ a tensiunii de ieşire (pMOS este blocat) tranzistorul nMOS la început este în regimul de saturaţie iar când $v_O \leq V_{DD} - Vpn$ trece în regimul liniar. Pentru excursia $L - H$ a tensiunii de ieşire (nMOS este blocat) tranzistorul pMOS la început este în regim de saturaţie şi apoi când $v_O \geq V_{DD} + V_{pp}$ trece şi în regimul liniar. În rezolvarea ecuaţiei 1.69 se joncţionează intervalul de timp când tranzistorul este în regim liniar de funcţionare, se utilizează relaţia 1.33, cu intervalul de timp când tranzistorul este în regimul de saturaţie, se utilizează relaţia 1.36. Se obţin următoarele expresii pentru τ_{HL} şi τ_{LH} [Weste '92].

$$\tau_{\text{HL}} \approx \frac{K}{V_{DD}} \cdot \frac{C_L}{\beta_n} \tag{1.70}$$

$$\tau_{\text{LH}} \approx \frac{K}{V_{DD}} \cdot \frac{C_L}{\beta_p} \tag{1.71}$$

în care $K = 3 \div 4$ pentru $V_{DD} = 3 \div 5V$, şi V_{pn} şi $|V_{pn}|$ au valori între $(0,5 \div 1)V$.

Pentru un inversor la care factorii de formă ai celor două tranzistoare sunt egali $\left(\frac{W}{L}\right)_n = \left(\frac{W}{L}\right)_p$ (rezultă din relaţia 1.54 că $\beta_n = 2\beta_p$, pentru că $\mu_n = 2\mu_p$) se obţine din 1.70 şi 1.71 o relaţie uzuală cunoscută în practica proiectării:

$$\tau_{\text{HL}} = \frac{\tau_{\text{LH}}}{2} \tag{1.72}$$

adică în semnalul de ieşire $v_O(t)$ tranziţiile nu sunt egale, durata tranziţiei de la $L \rightarrow H$ este aproximativ de două ori mai lungă decât durata tranziţiei de la $H \rightarrow L$.

Pentru obţinerea unui semnal de ieşire cu tranziţii $H - L$ şi $L - H$ simetrice, la ieşirea inversorului, trebuie ca $\beta_n/\beta_p = 1$ ceea ce implică, pentru lungimi egale de canal, să se realizeze canalul p cu o lăţime cam de două până la trei ori mai mare decât lăţimea canalului n ($\mu_n \approx 2\mu_p$).

$$W_p \approx (2 \div 3)W_n \tag{1.73}$$

4. Metoda empirică. Prin simulare în SPICE, pentru o variantă de inversor, se determină valoarea exactă a constantei K din relaţiile 1.70 şi 1.71. Apoi,utilizând această valoare determinată pentru K, pentru alte variante de inversor, realizate în aceeaşi tehnologie, se calculează τ_{HL} şi τ_{LH} cu relaţiile 1.70 şi 1.71.

Exemplul 1.16 Pentru o turnătorie de siliciu care are un proces cu următorii parametrii: $\mu_n C_{ox} = 30\mu A/V^2$, $\mu_p C_{ox} = 10\mu A/V^2$, $L = 1\mu m$ atât pentru canal n cât şi pentru canal p, $V_{pn} = 1,0V$, $V_{pp} = -1,5V$, $W_{min} = 2\mu m$ să se dimensioneze un inversor CMOS, lăţimile de canal W_n şi W_p, încât să se obţină următoarele caracteristici:

1. $V_T = 2V$ pentru $V_{DD} = 5V$;
2. Durata timpului de cădere τ_{HL} să fie de $2ns$ când tensiunea v_O are variaţia de la $4V$ la $1V$.

Soluţie. Considerând saltul în v_I instantaneu de la 0 la 5V rezultă că atunci când $v_O = 4V$ tranzistorul nMOS trece din regimul de saturaţie în regimul liniar ($V_{DS} = 4V \leq V_{GS} - V_P = 5V - 1V = 4V$) deci în ecuaţia 1.69 se introduce exprimarea din relaţia 1.33:

$$C_L \frac{dv_O}{dt} = -\frac{1}{2}\mu_n \cdot C_{ox} \cdot \frac{W_n}{L_n}\left[2(V_{OH} - V_{pn})v_O - v_O^2\right]$$

iar prin integrare se obţine

$$\begin{aligned}\tau_{\text{HL}} &= 2 \cdot 10^{-16} = -2C \int_{v_O=4}^{v_O=1} \frac{dv_O}{\mu_n \cdot C_{ox} \cdot \frac{W_n}{L_n}\left[2(V_{OH} - V_{pn})v_O - v_O^2\right]} = \\ &= \frac{1 \times 10^{-12}}{30 \times 10^{-16} \times \frac{W_n}{L_n} \times 4}\end{aligned}$$

din care se obţine pentru canalul n

$$\frac{W_n}{L_n} = 8,1$$

şi pentru $L_n = 1\mu m$ rezultă $W_n = 8,1\mu m$

Condiţia impusă $V_T = 2V$ ne ajută să obţinem dimensiunea inversorului. Conform relaţiei 1.46 se poate scrie:

$$2 = \frac{V_{pn} + \sqrt{\frac{\beta_p}{\beta_n}}(V_{DD} + V_{pp})}{1 + \sqrt{\frac{\beta_p}{\beta_n}}}$$

iar raportul β_n/β_p se obţine ca fiind

$$\frac{\beta_n}{\beta_p} = \frac{\mu_n C_{ox} \frac{W_n}{L_n}}{\mu_p C_{ox} \frac{W_p}{L_p}} = \frac{9}{4}$$

rezultă lăţimea W_p (pt $L_p = 1$)

$$W_p = \frac{\beta_p}{\beta_n} \times \frac{\mu_n W_n}{\mu_p} = \frac{4}{9} \times \frac{3}{1} \times 8,1 = 10,8\mu m$$

Deci inversorul cu dimensiunile $L = 1\mu m$, $W_n = 8,1\mu m$ şi $W_p = 10,8\mu m$ satisface condiţiile impuse.

Exemplul 1.17 Pentru un oscilator în inel să se determine frecvenţa de oscilaţie.

Soluţie. Un oscilator în inel, după cum şi denumirea indică, se obţine prin conectarea în inel a unui număr n impar de inversoare. Închiderea buclei peste un singur inversor $v_I = v_O$ determină o instabilitate deoarece având variaţii permanent opuse se comandă ca intrarea să se modifice din $1 \rightarrow 0$ şi invers, la fel şi ieşirea din $0 \rightarrow 1$ şi invers. Singurul punct de funcţionare când $v_I = v_O$, este la tensiunea de prag de comutaţie a inversorului V_T, dar

după cum s-a văzut din Figura 1.33-c acesta nu este un punct stabil de funcţionare. Acelaşi raţionament se poate extinde când sunt cuprinse în buclă un număr impar de inversoare, Figura 1.38-a. Considerând inversoarele identice cu $\tau_{\text{HL}} = \tau_{\text{LH}}$ şi $\tau_{\text{pHL}} = \tau_{\text{pLH}} = \tau_{\text{p}}$ variaţiile tensiunilor v_{O1}, v_{O2} şi v_{O3} sunt reprezentate în Figura 1.38-b. Perioada T a oscilaţiilor se poate calcula simplu :

$$T = \tau_{\text{pHL1}} + \tau_{\text{pLH1}} + \tau_{\text{pHL2}} + \tau_{\text{pLH2}} + \tau_{\text{pHL3}}\tau_{\text{pLH3}} = 2\tau_{\text{p}} + 2\tau_{\text{p}} + 2\tau_{\text{p}} = 6\tau_{\text{p}}$$

iar frecvenţa oscilaţiilor rezultă (pentru un număr n de inversoare)

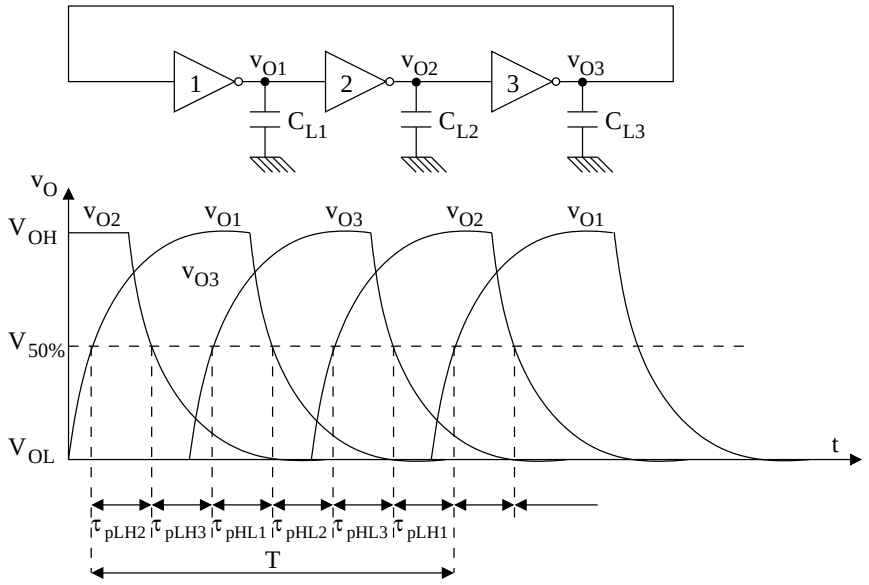


Figura 1.38 Oscilatorul în inel: a) structură cu trei inversoare; b) formele de variaţie ale tensiunilor v_{O1}, v_{O2}, v_{O3}

$$f = \frac{1}{T} = \frac{1}{2n\tau_{\text{p}}} \tag{1.74}$$

Această relaţie poate fi utilizată pentru determinarea timpului de propagare τ_{p} pentru un inversor. În tehnologia respectivă, pe o plachetă se realizează pentru test un număr impar de inversoare, în general un număr mai mare de trei, care se conectează în inel. În urma măsurării frecvenţei de oscilaţie se poate determina, cu relaţia 1.74, timpul de propagare. Pentru tehnologia respectivă, apoi, se poate exprima timpul de propagare al circuitelor mai complexe ca multiplii de τ_{p} (vezi metoda efortului logic).

1.5.3 Familia de porţi logice CMOS

Circuitul inversor CMOS, ca şi inversorul bipolar pentru porţile TTL, este celula de bază în structurarea porţilor CMOS; porţile CMOS pot fi obţinute prin extensia circuitului inversor, respectiv porţile CMOS pot fi reduse la o structură echivalentă de inversor.

Inversorul CMOS prin cele două canale n şi p are o complementaritate în modul de a fi comandat, o comandă de 1 logic pentru canalul n este o comandă de 0 logic pentru canalul p şi invers, ceea ce permite ca ramura p a inversorului să fie privită ca duala ramurii n şi invers. Dar, realizând conexiuni (reţele) care pot fi serie, paralel, serie-paralel şi paralel-serie cu ramuri de tip n în conceptul dualităţii înseamnă că reţeaua corespunzătoare formată din ramuri p trebuie să fie respectiv paralel, serie, paralel-serie şi serie-paralel. Duala unei relaţii logice se obţine conform relaţiei 1.2 iar pentru axiomele şi teoremele algebrei booleene formele duale sunt prezentate în *Tabelul 1.2* (duala lui AND este OR şi invers).

Mai multe tranzistoare în conducţie, fie cu canal n fie cu canal p, toate având aceeaşi tensiune de prag şi neglijând efectul de polarizare a substratului, când sunt într-o reţea, conectate în paralel sau în serie, pot fi substituite cu un singur tranzistor echivalent în conducţie. Deoarece conductanţa canalului este proporţională cu coeficientul de formă al tranzistorului W/L, dimensiunile canalului tranzistorului echivalent al reţelei se calculează cu relaţii similare conectării în serie sau în paralel a conductanţelor. Astfel coeficientul de formă al tranzistorului echivalent pentru conectarea a k tranzistoare în paralel se calculează cu relaţia:

$$\left(\frac{W}{L}\right)_{\text{echivalent}} = \sum_{i=0}^{k} \left(\frac{W}{L}\right)_i \tag{1.75}$$

respectiv pentru conectarea a k tranzistoare în serie:

$$\left(\frac{W}{L}\right)_{\text{echivalent}} = \frac{1}{\sum_{i=0}^{k} \frac{1}{\left(\frac{W}{L}\right)_i}} \tag{1.76}$$

1.5.3.1 Poarta NOR şi NAND cu două intrări

Amintind faptul că operatorul OR poate fi modelat prin conexiunea paralelă a elementelor de comutaţie iar operatorul AND prin conexiunea serie apare foarte simplă modalitatea de a structura porţile NOR şi NAND.

Structura porţii NOR cu două intrări (NOR2), Figura 1.39-a, constă din două inversoare la care ramurile canalelor n formează o reţea paralelă iar ramurile canalelor p formează o reţea serie. Când cel puţin una din intrările A sau B este în starea H reţeaua n crează o cale de conducţie a nodului de ieşire spre masă, V_O=V_{OL}, reţeaua complementară p nu este în conducţie. Iar când ambele intrări sunt în starea L, reţeaua p creează o cale de conducţie între V_{DD} înspre nodul de ieşire, $V_O = V_{OH}$, reţeaua n nu este în conducţie. Pentru determinarea tensiunii de prag (logic) de comutaţie a porţii V_T ($V_A = V_B = V_O = V_T$) poarta NOR cu două intrări este substituită cu structura echivalentă a unui inversor cu coeficienţii $\beta_n/2$ şi $2\beta_p$, Figura 1.39-b. Rescriind relaţia 1.46 pentru aceşti coeficienţi rezultă expresia pentru tensiunea de prag de comutaţie a porţii NOR2.

$$V_{T(NOR2)} = \frac{V_{pn} + \sqrt{\frac{\beta_p}{4\beta_n}}\,(V_{DD} - |V_{pp}|)}{1 + \sqrt{\frac{\beta_p}{4\beta_n}}} \tag{1.77}$$

Dacă $\beta_n = \beta_p$ şi $V_{pn} = |V_{pp}|$ tensiunea de prag de comutaţie a inversorului CMOS este $V_{DD}/2$ pe când a porţii NOR2 din relaţia 1.77 rezultă

$$V_{T(NOR2)} = \frac{V_{DD} + V_{pn}}{3} \tag{1.78}$$

care este diferită de $V_{DD}/2$. De exemplu, pentru $V_{DD} = 5V$ şi $V_{pn} = |V_{pp}| = 1V$ se obţine $V_{T(INVERSOR)} = 2,5V$ şi $V_{T(NOR2)} = 2V$.

Relaţia 1.77 pote fi utilizată pentru proiectarea porţii NOR2 în care dacă se impune tensiunea de prag (logic) de comutaţie V_T rezultă relaţia între β_n şi β_p. De exemplu, pentru $V_T = V_{DD}/2$ şi $V_{pn} = |V_{pp}|$ rezultă $\beta_p = 4\beta_n$ (cu $L_n = L_p \rightarrow W_p = 4W_n$).

Layoutul porţii NOR2 este compus dintr-o "linie de difuzie" de tip p^+ pentru zonele de sursă şi dren ale celor două tranzistoare T3 şi T4 realizate în insula difuzată de tip n şi la fel o linie de difuzie de tip n^+ în substrat care realizează zonele de dren şi sursă ale tranzistoarelor T1 şi T2. Poarta comună pentru nMOS T1, pMOS T3 şi poarta comună pentru nMOS T2 şi pMOS T4 sunt sub forma a două bare (trase) verticale realizate din polisiliciu. La extremitatea de sus şi extremitatea de jos a layoutului sunt trasele metalice pentru V_{DD} şi V_{SS} (masă) cu conexiunile metalizate (pătrăţele înegrite) corespunzătoare la zonele (liniile) de difuzie p^+ şi n^+ respectiv din insula difuzată n şi din substrat. Trasa de ieşire V_O, metalică sau din polisiliciu, conectează între ele cele două linii difuzate de canal n^+ şi p^+. Realizarea tranzistoarelor în linii de difuzie paralele simplifică atât layoutul cât şi tehnologia; toate zonele de tip n^+ se realizează cu o singură difuzie la fel şi toate zonele de tip p^+, iar barele verticale ale porţilor de polisiliciu servesc şi ca măşti cu autoaliniere în procesul de difuzie. Acest mod de organizare structurată constituie o condiţie în conceperea unor metode de generare automată a layoutului.

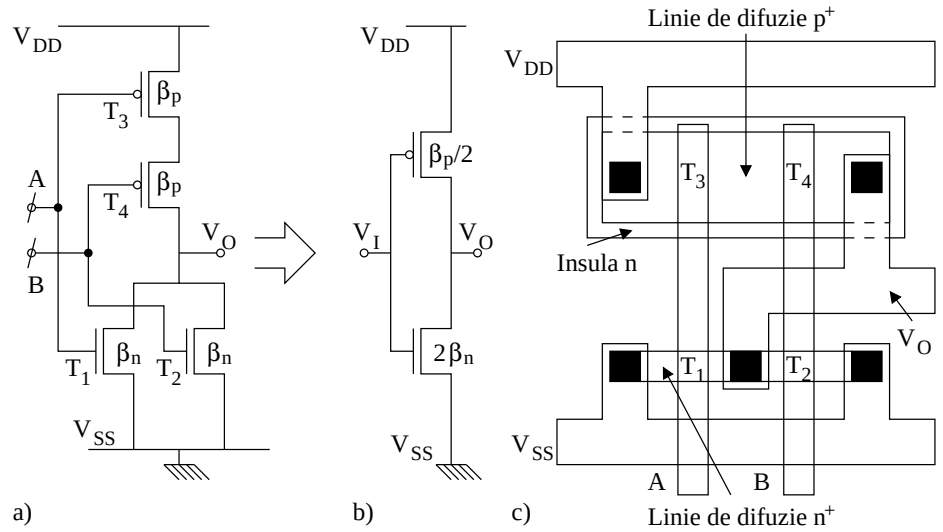


Figura 1.39 Poarta NOR cu două intrări: a) stuctură; b) schema inversorului echivalent porţii; c) layoutul porţii structurat pe o linie de difuzie n^+ şi o linie de difuzie p^+.

Poarta NAND cu două intrări (NAND2), Figura 1.40-a, poate fi privită ca fiind formată din două inversoare la care canalele n formează o reţea serie iar canalele p formează o reţea paralelă. Se creează o cale de conducţie de la nodul de ieşire spre masă, $V_O = V_{OL}$, prin tranzistoarele T1 şi T2 în serie numai când ambele intrări A,B sunt în nivelul H, iar tranzistoarele T3 şi T4 ale reţelei complementare p sunt blocate. Pentru toate celelalte trei combinaţii ale nivelurilor intrărilor A,B unul sau ambele tranzistoare ale reţelei p conduc, reţeaua n nu conduce, iar tensiunea de ieşire are valoarea $V_O = V_{OH}$.

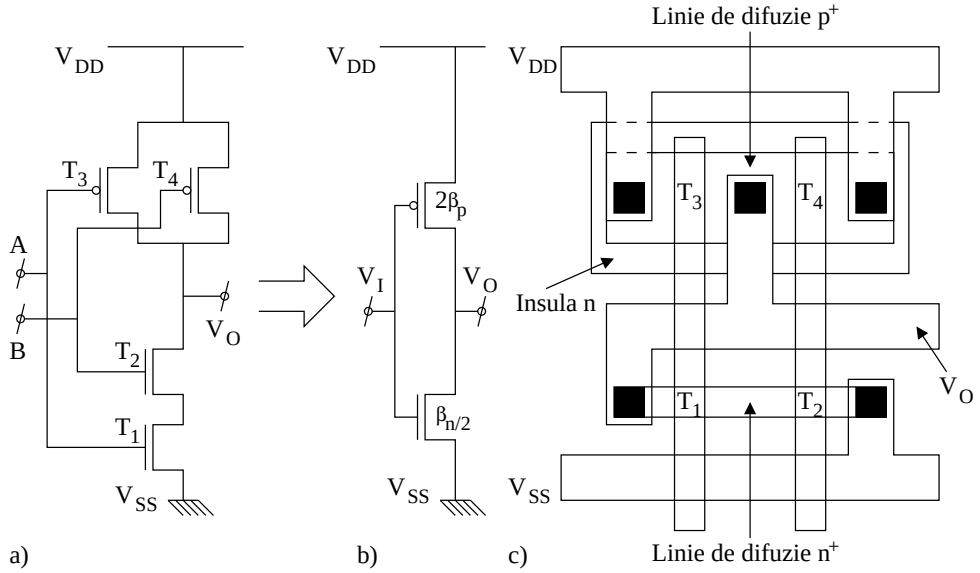


Figura 1.40 Poarta NAND cu două intrări: a) stuctură; b) schema inversorului echivalent porţii; c) layoutul porţii structurat pe o linie de difuzie n^+ şi o linie de difuzie p^+.

Tensiunea de prag (logic) de comutaţie a porţii, relaţia 1.46, aplicată pentru inversorul echivalent din Figura 1.40-b, are expresia:

$$V_{T(NAND2)} = \frac{V_{pn} + 2\sqrt{\frac{\beta_p}{\beta_n}}\,(V_{DD} - |V_{pn}|)}{1 + 2\sqrt{\frac{\beta_p}{\beta_n}}} \tag{1.79}$$

Pentru valorile $\beta_n = \beta_p$, $V_{pn} = |V_{pn}|$, la care pragul logic de comutaţie al inversorului CMOS este $V_{DD}/2$, porţii NAND2 îi corespunde o tensiune de prag logic de comutaţie:

$$V_{T(NAND2)} = \frac{2V_{DD} - |V_{pp}|}{3} \tag{1.80}$$

Din relaţia 1.79 rezultă că pentru a obţine un prag logic de $V_{DD}/2$ când $V_{pn} = |V_{pp}|$ este necesară îndeplinirea condiţiei $\beta_n = 4\beta_p$; relaţia 1.80 poate fi utilizată pentru dimensionarea layoutului porţii NAND2 când se impune o anumită valoare pentru V_T.

Layoutul porţii NAND2, Figura 1.40-c, este structurat şi realizat pe două linii de difuzie, în mod asemănător cu cel al porţii NOR2 din Figura 1.39-c.

Trecerea de la schema electrică a circuitului la layout se poate face conform succesiunii de paşi prezentaţi în Figura 1.36. Totuşi, pentru circuite complexe, această trecere directă poate fi dificilă în consecinţă se utilizează iniţial o formă simplificată/intermediară de layout (stick diagram). Forma simplificată conţine informaţii despre plasarea relativă a tranzistoarelor şi a conexiunilor dintre acestea, Figura 1.41. În aceste forme simplificate suprafeţele difuzate (liniile de difuzie n^+ şi p^+) sunt reprezentate sub forma unor dreptunghiuri (W_n şi W_p), traseele metalice sunt simple linii de conexiune având pentru contacte punctele evidenţiate, iar barele de polisiliciu pentru porţi sunt coloane haşurate. Conform acestor reguli pentru trecerea din Figura 1.39, de la circuitul porţii NOR2 la layoutul corespunzător, se poate realiza iniţial layoutul simplificat din Figura 1.41-a, iar pentru trecerea din Figura 1.40, corespunzător porţii NAND2, este realizat iniţial layoutul simplificat din Figura 1.41-b. Apoi de la layoutul simplificat se trece la forma completă de layout.

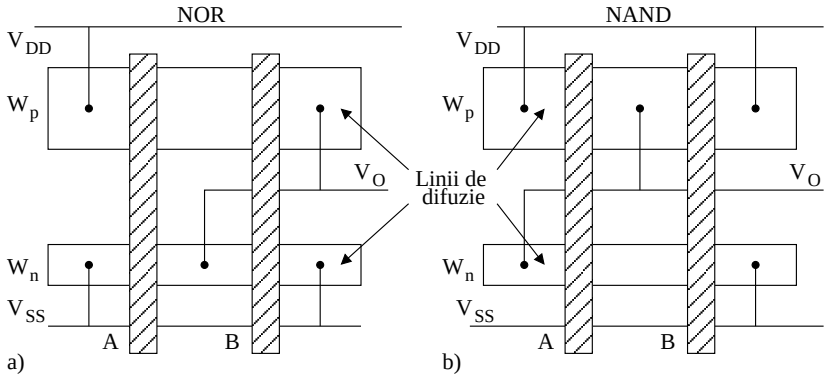


Figura 1.41 Layoutul simplificat / intermediar: a) pentru poartă NOR2; b) pentru poartă NAND2.

1.5.3.2 Porţi logice complexe

La implementarea porţilor complexe, care conţin multe variabile de intrare, aspectele implicite care se impun a fi optimizate sunt: micşorarea numărului de tranzistoare folosite şi micşorarea ariei utilizate pe placheta de siliciu.Evidenţierea acestor aspecte va rezulta prin exemplificarea implementării unei porţi care realizează următoarea relaţie logică:

$$Z = \overline{A(D+E)+BC}$$

Pentru expresia nenegată a relaţiei se va construi o reţea/graf similar ca la modelarea acesteia cu contacte (prin structura sa o poartă CMOS modelează o expresie negată). (De fapt se poate porni de la desenarea unei reţele cu contacte, care modelează funcţia respectivă, ca în Figura 1.9, şi din care se deduce graful expresiei nenegate). Deoarece în locul contactelor se utilizează tranzistoare (canale) n sau p, pentru o linie ce ar conţine un contact (în reţeaua de contacte), acum în graf, se va desena un simplu arc pe care se notează variabila de comandă a porţii tranzistorului respectiv. (Un

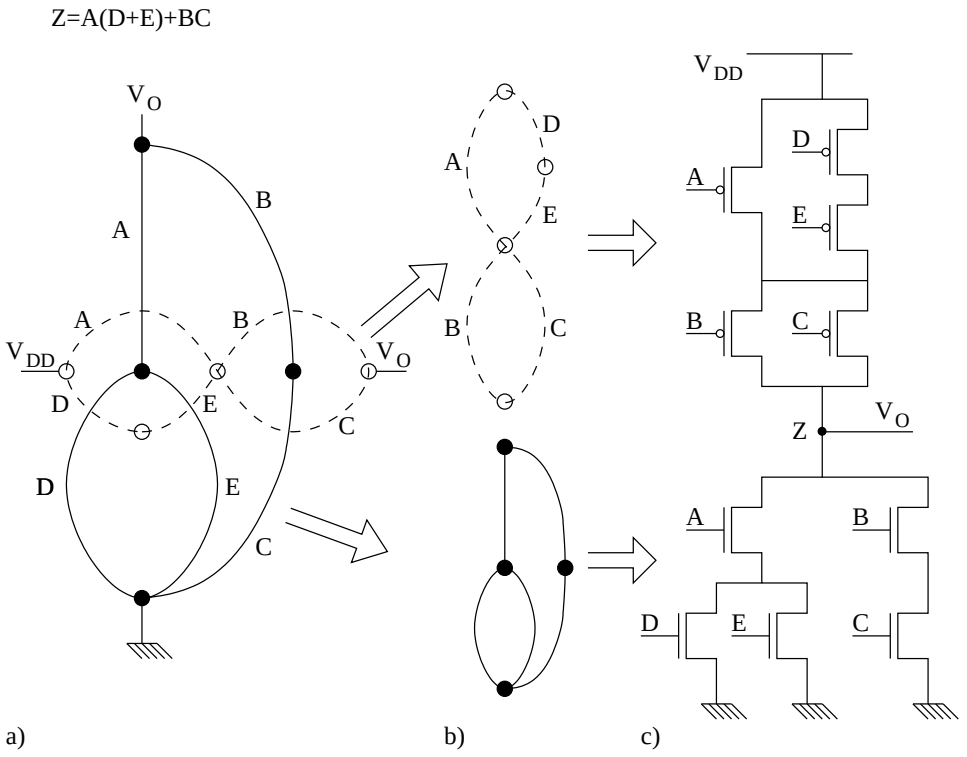


Figura 1.42 Explicativă pentru obţinerea unei structuri de reţea CMOS pentru relaţia $Z = \overline{A(D+E)+BC}$: a,b) realizarea grafului pentru reţeaua n şi deducerea grafului pentru reţeaua duală p; c) structura porţii obţinute prin maparea grafurilor pentru reţeaua n şi reţeaua p.

tranzistor este echivalent unui contact). Conexiunea dintre liniile cu contacte va reprezenta acum vârfurile (nodurile) grafului. Astfel se obţine graful pentru reţeaua n, conturul îngroşat din Figura 1.42-a.

Pe baza grafului reţelei n se va construi graficul reţelei duale p în felul următor: în fiecare suprafaţă închisă sau semiînchisă formată de graful reţelei n se fixează un nou vârf (punctele cerculeţe), se unesc câte două din aceste vârfuri prin câte un nou arc astfel încât aceste arce noi să intersecteze doar o singură dată un arc al reţelei iniţiale n, fiecărui arc nou i se asignează aceeaşi variabilă ca aceea a arcului pe care l-a intersectat - graful nou obţinut este graful reţelei duale p, desenat cu linie întreruptă în Figura 1.42-a. Având, acum, cele două grafuri desenate separat, în Figura 1.42-b, se poate, printr-o mapare unu-la-unu, trece de la aceste grafuri la reţele de tranzistoare n şi p, Figura 1.42-c. Fiecărui arc din graf îi corespunde un tranzistor pe poarta căruia se aplică variabila înscrisă pe acel arc; punctelor cerculeţe le corespund conexiunile între tranzistoarele respective.

Urmează transformarea circuitului electric în layout pe siliciu. Se obţine o arie minimă când, atât pentru toate tranzistoarele n cât şi pentru toate tranzistoarele

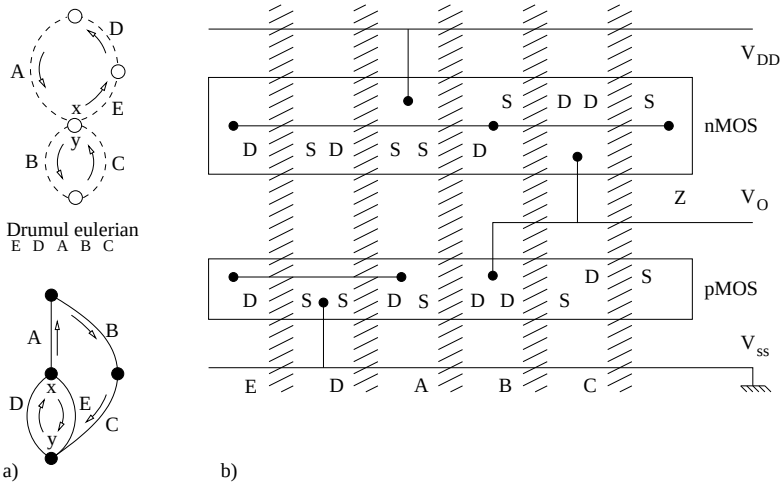


Figura 1.43 Explicativă pentru layoutul porţii $Z = \overline{A(D+E)+BC}$**:** a) identificarea drumului eulerian în graful reţelei n şi reţelei p; b) pe liniile continue de difuzie de tip n^+ şi p^+ s-au ordonat porţile din polisiliciu (barele verticale haşurate) ale tranzistoarelor în ordinea găsită la parcurgerea drumului eulerian.

p, se poate realiza câte o difuzie în linie neîntreruptă pentru canalele n şi pentru canalele p. Pentru realizarea unei difuzii în linie neîntreruptă, în care o zonă din această linie de difuzie dintre două bare de polisiliciu (porţi) să fie un terminal comun la două tranzistoare vecine, este necesar a se găsi ordinea de înlănţuire (plasare) a fiecărui tranzistor al reţelei în linia difuzată. Această ordine poate fi determinată prin identificarea unui traseu eulerian comun atât în graful reţelei p cât şi în graful reţelei n.

Definiţia 1.15 Un **drum** care parcurge neîntrerupt o singură dată toate arcele dintr-un graf se numeşte **eulerian**. $\diamond$

Pentru grafurile duale din Figura 1.42-b s-a identificat drumul eulerian comun $E-D-A-B-C$ din Figura 1.43-a. Cunoscând acum ordinea de joncţionare/alăturare a tranzistoarelor (care este aceeaşi cu ordinea de parcurgere în drumul eulerian) se poate realiza succesiunea porţilor din polisiliciu (barele haşurate) pe liniile de difuzie n^+ şi p^+ ca în layoutul simplificat din Figura 1.43-b. Dacă în grafuri nu se poate identifica un singur drum eulerian distinct, pentru toate tranzistoarele circuitului, ci mai multe drumuri distincte care acoperă graful, atunci sunt necesare atâtea linii de difuzie n^+ şi p^+ întrerupte câte drumuri euleriene distincte au fost identificate.

O funcţie logică poate fi sub forma sumă de produse (FD) sau produse de sumă (FC); forma negată a acestor funcţii este potrivită pentru implementarea în tehnologia CMOS deoarece se pot realiza uşor porţi de tipul AND-OR-NOT şi OR-AND-NOT. Porţile AND-OR-NOT au pentru reţeaua n o structurare paralel-serie şi o structurare serie - paralel pentru reţeaua p dual\u0103, Figura 1.44-a, iar porţile OR-AND-NOT au o structurare serie-paralel pentru reţeaua n şi paralel-serie pentru reţeaua p, Figura 1.44-b.

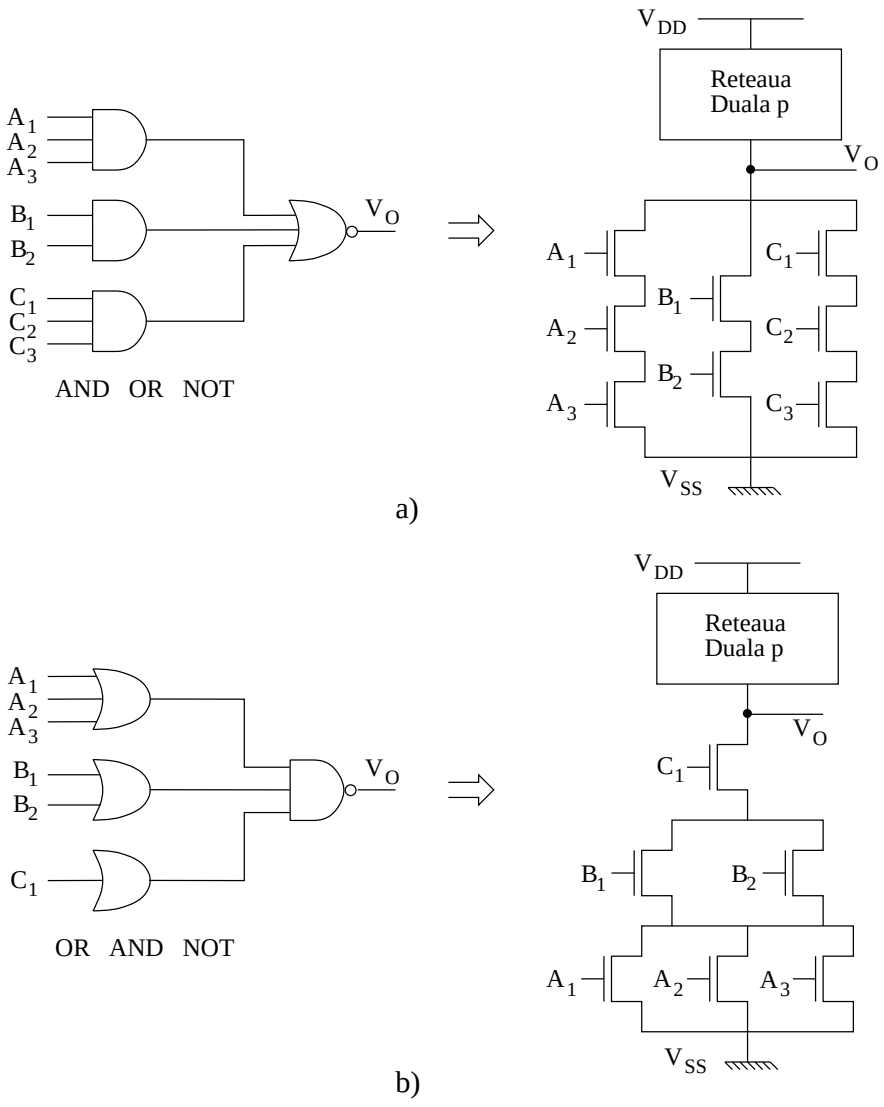


Figura 1.44 Modalităţi de structurare a porţilor CMOS complexe: a) structurare pentru implementarea funcţiilor logice de forma AND-OR-NOT; b) structurare pentru implementarea funcţiilor logice de forma OR-AND-NOT.

Exemplul 1.18 Pentru circuitul cu layoutul desenat în Figura 1.45 să se deducă structura de circuit apoi să se determine circuitul inversor CMOS echivalent pentru cazul când toate intrările comută, presupunând că: $(W/L)_p = 15$ pentru toate tranzistoarele pMOS şi $(W/L)_n = 10$ pentru toate tranzistoarele nMOS.

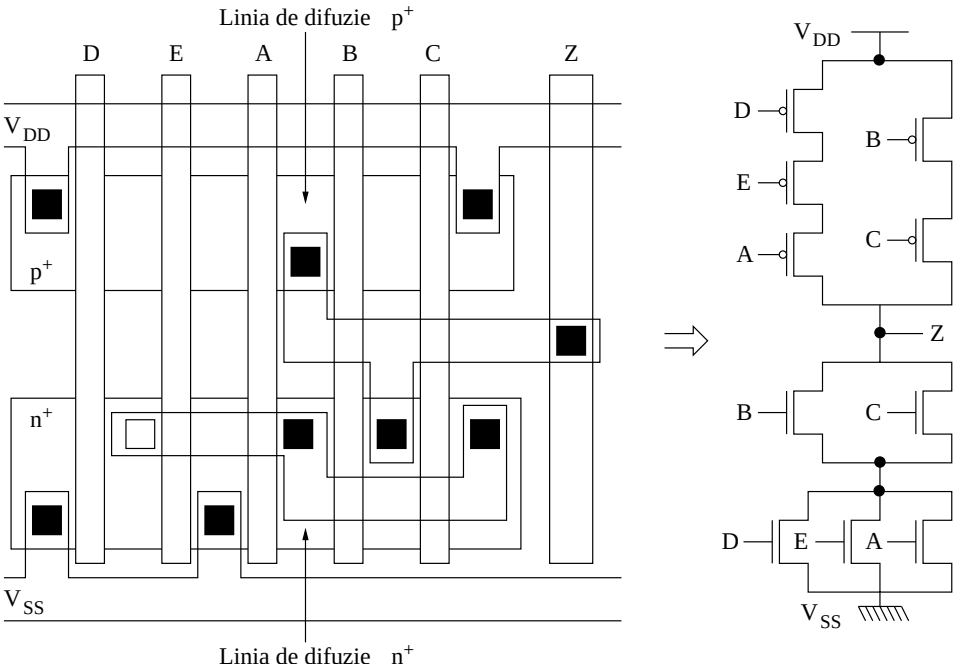


Figura 1.45 Trecere de la layout de circuit la schema electrică.

Soluţie. Faţă de prezentările anterioare aici se face o trecere inversă, de la layout la structura de circuit. Inspectând în linia de difuzie n^+ trasele şi contactele metalice rezultă că există grupul de tranzistoare D, E, A legate în paralel şi grupul de tranzistoare B, C legate în paralel iar cele două grupuri sunt înseriate. Reţeaua n a porţii modelează următoarea relaţie $(D+E+A)(B+C)$, deci ieşirea Z este $\overline{(D+E+A)(B+C)}$ pentru care corespunde circuitul poartă CMOS din Figura 1.45-b.

Rapoartele echivalente, $(W/L)_{n,INV}$ şi $(W/L)_{p,INV}$, ale inversorului echivalent când comută toate intrările se obţin cu relaţiile 1.75 şi 1.76 în felul următor:

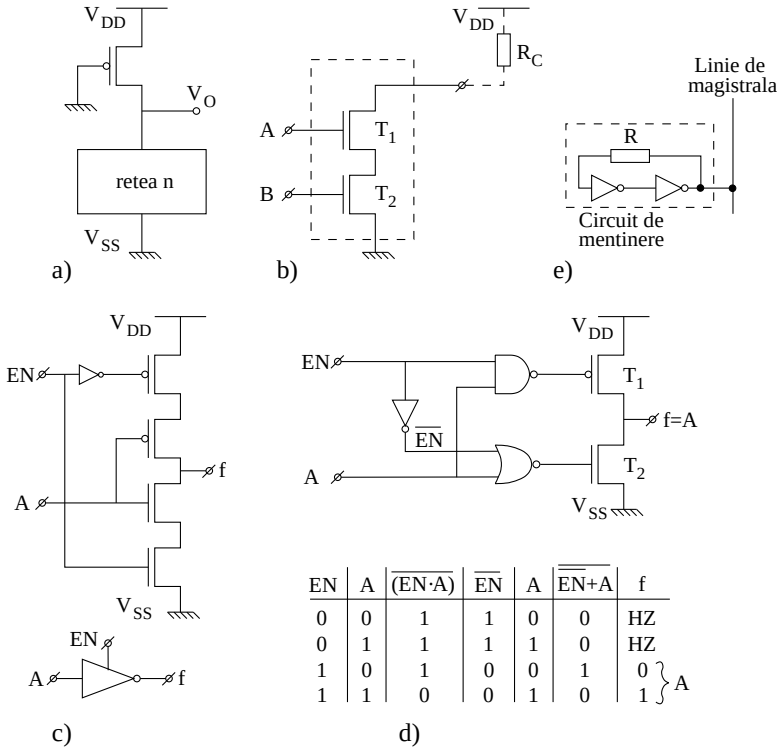
$$\left(\frac{W}{L}\right)_{n,INV} = \frac{1}{\frac{1}{\left(\frac{W}{L}\right)_D+\left(\frac{W}{L}\right)_E+\left(\frac{W}{L}\right)_A} + \frac{1}{\left(\frac{W}{L}\right)_B+\left(\frac{W}{L}\right)_C}} = \frac{1}{\frac{1}{30}+\frac{1}{20}} = 12$$

$$\left(\frac{W}{L}\right)_{p,INV} = \frac{1}{\frac{1}{\left(\frac{W}{L}\right)_D} + \frac{1}{\left(\frac{W}{L}\right)_E} + \frac{1}{\left(\frac{W}{L}\right)_A} + \frac{1}{\left(\frac{W}{L}\right)_B} + \frac{1}{\left(\frac{W}{L}\right)_C}} =$$

$$= \frac{1}{\frac{1}{15}+\frac{1}{15}+\frac{1}{15}} + \frac{1}{\frac{1}{15}+\frac{1}{15}} = 12,5$$

Pseudo poarta CMOS. Porţile CMOS complexe, cu n intrări, necesită pentru fiecare intrare două tranzistoare, deci în total $2n$ tranzistoare plus încă două tranzistoare (un inversor) pentru fiecare din intrările negate. Rezultă că reducerea numărului de tranzistoare, când numărul de intrări n este mare, se impune. O variantă de reducere a numărului de tranzistoare, de la $2n$ la $n+1$, se obţine la structura denumită pseudo poartă CMOS. Pseudo poarta CMOS, pentru un anumit operator, se obţine din poarta normală, care implementează acel operator, prin substituirea reţelei p cu un singur tranzistor pMOS a cărui poartă se conectează la potenţialul de masă (deci este permanent în conducţie), Figura 1.46-a. De fapt, tranzistorul pMOS, are rolul de sarcină (rezistenţă) pentru reţeaua nMOS.

Dezavantajul pseudo porţii CMOS constă în consumul mare de putere în regim static datorat unui permanent curent prin canalul p atât timp cât $V_O < V_{DD}$. De asemenea, V_{OL} şi marginea de zgomot sunt determinate de raportul între transconductanţa tranzistorului de sarcină supra transconductanţa echivalentă din reţeaua n.



EN	A	$\overline{(EN \cdot A)}$	$\overline{EN}$	A	$\overline{\overline{EN}+A}$	f
0	0	1	1	0	0	HZ
0	1	1	1	1	0	HZ
1	0	1	0	0	1	0 } A
1	1	0	0	1	0	1 } A

Figura 1.46 Porţi CMOS speciale: a) structurarea unei pseudo porţi CMOS; b) structură pentru poarta cu drenul în gol; c) structura şi simbolul de reprezentare ale unei porţi de tip TSL; d) structura unui buffer TSL neinversor; e) evitarea apariţiei unui potenţial flotant pe o linie de magistrală prin conectarea acesteia la un circuit (celulă) activ de menţinere a nivelului (bus holder).

Poarta cu drenul în gol. Restricţia impusă de a nu se conecta împreună ieşirile porţilor TTL, Figura 1.23-a, are valabilitate şi pentru porţile CMOS. Conectarea ieşirilor împreună a mai multor porţi poate crea o cale de scurtcircuit între V_{DD} şi V_{SS} prin reţelele n sau p de la diferite porţi.

Eliminarea acestei restricţii se poate obţine pentru porţile care sunt realizate, în etajul de ieşire, fără reţeaua p, numai cu reţeaua n, aceste porţi sunt denumite cu drenul în gol, Figura 1.46-b (similare porţilor cu colectorul în gol). Poarta devine funcţională numai când drenul în gol este conectat la tensiunea V_{DD} printr-o rezistenţă R_D ataşată exterior porţii. Porţile cu drenul în gol (open-drain) sunt necesare pentru următoarele aplicaţii: comanda unor sarcini externe (LED-uri, relee, rezistenţe, bobine etc), realizarea conexiunii SI-cablat şi comanda unor linii de magistrală, Figura 1.24-a; pentru calculul rezistenţei R_D ataşată în exteriorul porţii se utilizează relaţiile 1.26.

Poarta TSL. Această poartă prezintă pe lângă cele două stări logice normale H şi L, existente la o poartă obişnuită, şi starea când ieşirea este în înaltă impedanţă, HZ. Structura unei porţi TSL se obţine din cea a unei porţi normale la care se înseriază câte un tranzistor n şi p respectiv cu reţeaua n şi p, Figura 1.46-c. Când semnalul de validare EN (ENable) este activ, EN=1, poarta are funcţionarea unei porţi normale, iar când validarea nu este activată, EN=0, cele două tranzistoare înseriate sunt blocate, atât calea de conducţie prin reţeaua n cât şi cea prin reţeaua p nu sunt în conducţie, deci ieşirea este în HZ. O altă variantă de poartă CMOS TSL este cea cu structura din Figura 1.68-a. La o poartă CMOS TSL când este în HZ ieşirea sa este forţată, de o altă poartă care comandă în acel moment linia de magistrală, pe nivelul H sau pe nivelul L şi generează respectiv absoarbe curenţi la ieşire până la $10\mu A$. Pentru cazurile când sarcina ce trebuie comandată este relativ mare sunt utilizate buffere; o astfel de structură de buffer TSL este prezentată în Figura 1.46-d. Un buffer TSL se compune din două tranzistoare complementare T1 şi T2 comandate, prin porţiile NAND şi NOR, fie ambele tranzistoare simultan în blocare (starea HZ), fie un tranzistor în blocare şi celălalt în conducţie sau invers (ca la inversorul CMOS). Din tabelul de adevăr, ataşat bufferului, se observă că pentru EN=0 ieşirea este în HZ, iar pentru EN=1 ieşirea este identică cu intrarea (buffer neinversor).

Bufferele CMOS TSL sunt utilizate pentru comanda liniilor de magistrale; o linie de magistrală poate fi comandată, la un moment, de cel mult un buffer (emiţător). Dacă toate ieşirile bufferelor conectate la o linie de magistrală sunt în HZ atunci potenţialul pe linie este flotant, iar dacă acest potenţial este aproape de valoarea de prag de comutaţie (Definiţia 1.14) al porţilor receptoare acestea consumă un curent de valoare relativ mare sau pe linie pot apărea oscilaţii. Se pot elimina aceste inconveniente dacă linia de magistrală se conectează la V_{DD} printr-o rezistenţă R_{pu}, deci când linia devine flotantă potenţialul acesteia va fi "tras" în sus (pull-up) în intervalul de tensiune H. Dar această soluţie prezintă unele inconveniente:

1. dacă R_{pu} este de valoare mare, atunci când linia din L rămâne în stare flotantă, şi forţată imediat în H de R_{pu}, datorită faptului că prezintă o constantă de timp mare $R_{pu} \cdot C$, timpul de creştere τ_r devine lung. Un τ_r lung face ca durata excursiei, în zona tensiunilor interzise, pentru tensiunile de intrare ale porţilor receptoare să fie de asemenea lungă deci un consum mărit de putere;

2. dacă R_{pu} este de valoare mică atunci bufferul care va comanda linia în starea L

va trebui să absoarbă un curent mai mare.

Inconvenientele anterioare pot fi eliminate prin conectarea la linia de magistrală a unui circuit (celulă) activ de menţinere a nivelului (**bus holder**) cu structura din Figura 1.46-e. Se va vedea în secţiunea 3.3.1 că această celulă activă de menţinere nu este altceva decât un circuit latch. Când linia de magistrală rămâne în HZ, deci va trece din L sau din H în starea flotantă, celula va forţa menţinerea liniei în aceeaşi stare L sau H pe care a avut-o anterior. Bufferul de magistrală care comută linia din starea H în starea L sau din starea L în H va absorbi sau va genera un surplus de curent pentru comanda celulei de menţinere în starea L sau H, dar numai pe durata comutaţiei stărilor. În general, un buffer de magistrală are integrat pe ieşirea sa o astfel de celulă de menţinere a nivelului pe linia de magistrală.

Celula de menţinere a nivelului pe linia de magistrală nu este eficientă când la magistrală sunt conectaţi receptori de tip TTL. Porţile TTL necesită curenţi de intrare de valoare ridicată, mai ales în starea L, I_{IL}, care nu pot fi generaţi de către celula de menţinere dacă la această celulă rezistenţa R nu este de valoare mică, iar o rezistenţă de valoare mică duce la o încărcare puternică a liniei de magistrală.

1.5.3.3 Seriile de porţi ale familiei CMOS

În cadrul tehnologiei CMOS, porţile, elemente de bază pentru realizarea unor sisteme, pot fi celulele cu care se realizează sistemul sub formă de circuit integrat (monolitic) sau pot fi celule discrete (independente, integrate pe plachete separate) cu care se realizează sistemul pe o placă de circuit imprimat. Parametrii unei porţi logice, secţiunea 1.3, sunt mult mai restrictive pentru o poartă implementată ca circuit independent decât pentru o poartă inclusă într-un circuit integrat. Pentru porţile discrete aceşti parametri sunt specificaţi de fabricant în fişa tehnică ce însoţeşte poarta. În cadrul familiei de porţi CMOS discrete există mai multe serii de porţi, aceste serii au apărut în funcţie de optimizarea parametrilor pentru anumite aplicaţii şi mai ales ca urmare a perfecţionării în timp a tehnologiei de integrare.

Prima serie de porţi discrete a familiei CMOS a fost **seria 4000**. În prezent, porţile din această serie nu se mai utilizează deoarece au apărut alte serii mult mai performante. Există în cadrul fiecărei serii varianta civilă ($0° \div 70°$C) şi varianta militară ($-55° \div 125°$C), notate respectiv cu 74 sau 54. Cuvântul de cod al unei porţi este de forma 74SERXX/54SERXX, unde SER sunt două sau trei litere (abreviaţie) din denumirea SERiei în care este implementată poarta, iar XX este codul porţii. De exemplu, aceeaşi poartă NAND cu două intrări (NAND2) care are codul 7400 poate fi specificată prin cuvintele 74HC00, 74HCT00, 74VHC00, 74VHCT00 în funcţie de seria căreia îi aparţine (la prima privire, se poate spune, dacă poarta este din familia CMOS deoarece în abreviaţia seriei intră totdeauna litera C).

Seriile HC şi HCT. Seria **HC** (**High-speed CMOS**) este optimizată pentru realizarea în special de sisteme numai cu porţi CMOS. Utilizează o alimentare în gama de la $2 \div 6V$; valorile mai mici ale tensiunii de alimentare sunt recomandate când se doreşte o putere disipată mai mică iar valorile mai ridicate când este necesară o viteză mai ridicată. O comparaţie a seriei HC cu seriile din familia TTL arată că între acestea nu există compatibilitate a tensiunilor de ieşire şi a celor de intrare.

Pentru a se putea intermixa, în sisteme, porţile CMOS cu porţile TTL s-a conceput seria **HCT** (**High-speed CMOS**, **TTL compatible**). Valorile tensiunilor de ieşire

garantate (V_{OHmin}, V_{OLmax}) şi de intrare admise (V_{IHmin}, V_{ILmax}) de la seria HCT sunt identice cu cele de la familia TTL, deci porţile pot fi interconectate. Seriile HC şi HCT au aceiaşi parametrii pe ieşire dar parametrii diferiţi pe intrare, această diferenţă a fost creată la HCT pentru ca această serie să devină compatibilă cu TTL. Ambele serii au comanda pe ieşire simetrică, adică există egalitate între curentul absorbit de poartă în stare L cu cel generat de poartă în starea H (simetria aceasta de curenţi pe ieşire nu există la TTL).

Seria FCT (FCT-T). Prin introducerea seriei **FCT** (**Fast-CMOS**, **TTL compatible**) la începutul anilor 1990 s-a urmărit realizarea în CMOS a următoarelor performanţe: atingerea unei capabilităţi de comandă pe ieşire (valori mărite pentru I_{OHmax}, I_{OLmax}), viteză ridicată ca la cele mai performante porţi TTL dar în acelaşi timp reducerea puterii consumate şi, evident, o completă compatibilitate (pe nivelurile de tensiune) cu porţile TTL. Totuşi seria FCT realizată cu aceste performanţe era afectată de două deficienţe: fiind alimentată la $V_{DD} = 5V$, saltul tensiunii de ieşire la comutaţie era de aproape 5V, ceea ce ducea la o putere disipată ($C_L V^2 f$) foarte mare la frecvenţe de peste $25MHz$ şi, în plus, aceste salturi rapide generează zgomot în sistem. Pentru înlăturarea acestor deficienţe s-au introdus anumite perfecţionări în structura de circuit FCT obţinându-se astfel seria FCT-T (Fast CMOS, TTL compatible with TTL V_{OH}). Cele două deficienţe amintite anterior, de la FCT au fost atenuate la seria FCT-T în primul rând prin reducerea $V_{OH} \approx 5V$ la valoarea tipică

Tabelul 1.12 Parametrii porţilor familiei CMOS (prezentare simplificată)

Denumire parametru		Conditia de test	Seria				
			HC	HCT	VHC	VHCT	FCT−T
Timpul de propagare tipic	τ_p [ns]		9	10	5,2	5,5	5,8
Curentul in regim static	I_{DDQ} [μA]	V_{in}=0 sau V_{DD}	2,5	2,5	5	5	200
Puterea disipata in regim static	P_{dcc} [mW]	V_{in}=0 sau V_{DD}	0,0125	0,0125	0,025	0,025	1
Capacitatea interna a circuitului	[pF]		22	15	19	17	−
Puterea disipata in regim dinamic	P_{dca} [mW/MHz]		0,55	0,38	0,48	0,43	0,6
Puterea disipata totala	P_d [mW]	f=100KHz	0,068	0,050	0,073	0,068	0,60
		f=1MHz	0,56	0,39	0,50	0,45	1,06
		f=10MHz	5,5	3,8	4,8	4,3	1,6
Factorul de merit	P_d τ_p [pJ]	f=100KHz	0,61	0,50	0,38	0,37	6,15
		f=1MHz	5,1	3,9	2,6	2,5	9,3
		f=10MHz	50	38	2,5	24	41
Tensiunea de intrare permisa	V_{ILmax} [V]		1,35	0,8	1,35	0,8	0,8
Tensiunea de intrare permisa	V_{IHmin} [V]		3,85	2,0	3,85	2,0	2,0
Curentul de iesire in starea L	I_{OLmax} [mA]	incarcare CMOS	−0,02	0,02	0,05	0,05	−
		incarcare TTL	4,0	4,0	8,0	8,0	64
Curentul de iesire in starea H	I_{OHmax} [mA]	incarcare CMOS	−0,02	−0,02	−0,05	−0,05	−
		incarcare TTL	−4,0	−4,0	−8,0	−8,0	−15
Tensiunea de iesire in starea L	V_{OLmax} [V]	$I_{out} \leq I_{OLmax\ CMOS}$	0,1	0,1	0,1	0,1	
		$I_{out} \leq I_{OLmax\ TTL}$	0,33	0,33	0,44	0,44	0,55
Tensiunea de iesire in starea H	V_{OHmin} [V]	$\lvert I_{out} \leq I_{OLmax\ CMOS} \rvert$	4,4	4,4	4,4	4,4	
		$\lvert I_{out} \leq I_{OLmax\ TTL} \rvert$	3,84	3,84	3,80	3,80	2,4

de $V_{OH} = 3,3V$. Varianta FCT-T datorită performanţelor sale este, în prezent, foarte populară. Aplicaţii uzuale pentru porţile FCT-T sunt comanda liniilor de magistrală sau comenzi pentru sarcini mari (comparativ cu alte porţi CMOS poate absorbi şi genera pe ieşire valori foarte mari de curenţi, $I_{OLmax} \approx 60mA$). Seria FCT (FCT-T) nu are implementate porţile simple ci numai porţi complexe (zeci de tranzistoare). Selectiv, unii din parametrii porţilor logice din seriile familiei CMOS sunt prezentaţi în *Tabelul 1.12*. Pentru seriile HC, HCT, VHC şi VHCT (abrevierea VH este de la Very High-speed) au fost selectate unele din valorile parametrilor porţilor NAND2 (74XX00) iar pentru seria FCT-T au fost selectate unele din valorile parametrilor porţii 74FCT138T, care este un circuit decodificator $3:8$.

1.5.3.4 Interfaţarea TTL-CMOS şi CMOS-TTL

Interfaţarea între cele două familii se poate face cu condiţia respectării compatibilităţii nivelurilor de tensiune şi a factorilor de încărcare la ieşire, fan-out. Respectarea nivelurilor de tensiune înseamnă realizarea unor margini de zgomot în curent continuu M_H, M_L, calculate cu relaţia 1.18, care pot avea anumite valori pozitive acceptabile dar în nici un caz valori negative. Valorile nivelurilor de tensiune pe intrare şi pe ieşire pentru ambele familii sunt reprezentate în Figura 1.47 [Wakerly ′00]. Evident, că există compatibilitate pe nivelurile de tensiune între seriile HCT, VHCT, FCT şi familia TTL dar nu există comptibilitate completă între HC, VHC şi familia TTL. De exemplu, la interfaţarea HC sau VHC cu TTL rezultă $M_L = 0,8 - 0,33 = 0,47V$ şi $M_H = 3,84 - 2,0 = 1,84V$, iar la interfaţarea TTL cu HC sau VHC rezultă $M_L = 1,35 - 0,4 = 0,95$, $M_H = 2,7 - 3,85 = -1,15V$!; ar trebui ca V_{OHmin} de la TTL să fie ridicată cu cel puţin $1,15V$. O soluţie de compromis în realizarea şi a acestei interfaţări ar fi ridicarea valorii tensiunii V_{OHmin} prin conectarea unei rezistenţe R între ieşirea TTL şi V_{CC} (trebuie verficat dacă această rezistenţă de "tragere în sus" nu distruge prin căderea de tensiune $R \times I_{OLmax}$ nivelul de tensiune garantat V_{OLmax} când poarta TTL este comandată pe ieşire în L).

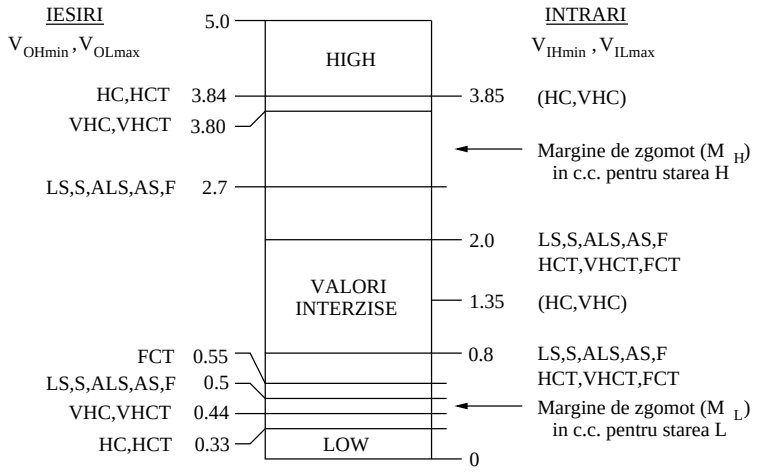


Figura 1.47 Nivelurile de tensiune pentru seriile familiei TTL şi familiei CMOS

Respectarea factorului de încărcare (în curent) la ieşire, relaţia 1.22, impune ca valoarea maximă a curentului absorbit sau generat la ieşirea porţii să fie mai mare sau egal cu suma curenţilor respectiv generaţi sau absorbiţi de intrările tuturor porţilor comandate. Interfaţarea TTL-CMOS nu ridică nici un fel de problemă, deoarece porţile CMOS aproape că nu necesită curent de intrare, atât în H cât şi în L curenţii de intrare au valori maxime $< 50\mu A$. În schimb pentru interfaţarea CMOS-TTL trebuie calculată valoarea de fan-out a porţii CMOS, de exemplu, porţile HC sau HCT pot comanda 10 porţi 74LSXXTTL, dar pot comanda numai două porţi 74SXXTTL.

La interconectarea porţilor CMOS factorul de încărcare se calculează nu în funcţie de un curent de intrare, ca sarcină standard, ci în funcţie de o sarcină capacitivă standard prezentată pe intrare de o poartă. În general, se consideră capacitatea de $5pF$ ca sarcină standard, care aproximează capacitatea de intrare medie la o poartă CMOS. Factorul de încărcare la ieşire se exprimă prin numărul de sarcini standard (nr de intrări) pe care le comandă la ieşire.Cu mărirea sarcinii capacitive conectate pe ieşire timpul de propagare al porţii creşte (aproximativ cu $1ns$ pentru fiecare sarcină de $5pF$ adăugată) .

Interfaţarea CMOS de tensiune redusă. Două sunt argumentele pentru care tensiunea de alimentare V_{DD} a circuitelor CMOS tinde a fi redusă:

1. Puterea disipată se reduce ($P_d = CV^2f$)
2. Scalarea determină şi micşorarea grosimii D_{ox} (Figura 1.32-a) a oxidului de sub poartă care, evident, pentru evitarea străpungerii, impune şi o tensiune de valoare mai mică aplicabilă pe poartă.

Au fost selectate tensiunile de alimentare: $3,3V \pm 0,3V$; $2,5V \pm 0,2V$, $1,8V \pm 0,15V$ ca valori pentru viitoarele standarde. Evident, în cadrul fiecărei tensiuni de alimentare au fost definite şi nivelurile logice de intrare şi de ieşire. Migrarea spre valori mai reduse de tensiuni de alimentare se face treptat, în etape, aceasta impunând ca unele din noile porţi logice CMOS discrete de $3.3V$ care apar să poată tolera încă tensiunile mai mari de intrare şi de ieşire TTL şi CMOS de la seriile alimentate la $5V$, această toleranţă fiind necesară pentru realizarea de sisteme cu porţi cu tensiuni diferite de alimentare. Necesitatea de intermixare a circuitelor CMOS de tensiuni reduse cu alte circuite de tensiuni mai ridicate există şi în cazul circuitelor integrate cum sunt μP şi **ASIC**-urile (**A**pplication **S**pecific **I**ntegrated **C**ircuits) numai că rezolvarea se face într-un alt mod. Aceste circuite fiind mari justifică alimentarea cu două tensiuni de alimentare, de exemplu cu $2,5V$ $(1,8V)$şi $3,3V$, vezi secţiunile 4.5. şi 4.6. Tensiunea scăzută de $2,5V$ alimentează nucleul de procesare, iar tesiunea ridicată componentele de interfaţare cu circuitele exterioare care sunt alimentate cu tensiunea de $3,3V$. În interiorul circuitului integrat există circuite buffer speciale, alimentate la cele două tensiuni $3,3V$ şi $2,5V$, care fac deplasările de nivel de la tensiunea coborâtă la cea ridicată şi invers.

Nivelurile logice de tensiune pentru familia CMOS la alimentare cu $5V$ precum şi la valorile reduse sunt prezentate în Figura 1.48. Nivelurile logice de tensiune, Figura 1.48-a, corespund seriilor HC şi VHC ale familiei CMOS de $5V$ adică acelor porţi CMOS care să fie interconectate numai cu porţi CMOS. În Figura 1.48-b sunt prezentate nivelurile logice de tensiune pentru seriile HCT, VHCT, FCT şi TTL (compatibile între ele ca niveluri de tensiune), care evident pot fi substituite între ele dacă sunt satisfăcute şi condiţiile de încărcare la intrare şi ieşire.

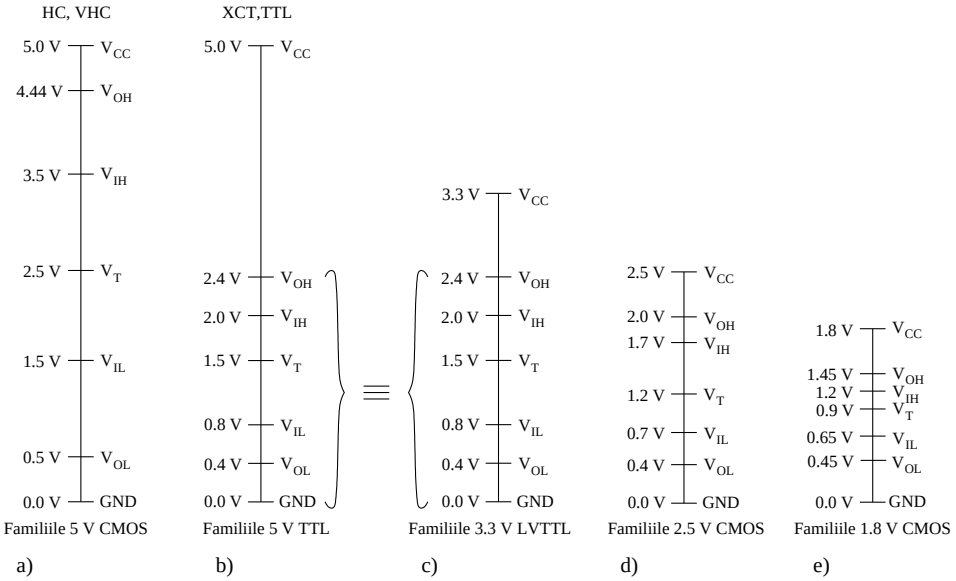


Figura 1.48 Nivelurile logice de tensiune pe intrare şi ieşire pentru toate familiile CMOS

Familia CMOS la $3,3V$**.** Pentru alimentarea cu $3,3V$ se definesc două seturi de niveluri logice de tensiune. Primul set de niveluri, abreviat prin **LVCMOS** (**L**ow-**V**oltage **CMOS**), este definit pentru porţile CMOS care se interconectează numai cu porţi CMOS. Aceasta înseamnă că valorile de încărcare în regim static sunt foarte reduse (mai puţin de $100\mu A$) astfel că V_{OL} şi V_{OH} se menţin la diferenţe de maximum $0,2V$ faţă de $0V$ respectiv faţă de $V_{DD} = 3,3V$. De fapt, nivelurile LVCMOS corespund seriilor HC şi VHC la $3,3V$.

Al doilea set de niveluri logice de tensiune abreviat prin **LVTTL** (**L**ow-**V**oltage **TTL**) reprezentat în Figura 1.48-c este definit pentru porţile CMOS care sunt utilizate în aplicaţii cu sarcini mărite şi care pot produce pentru tensiunea de ieşire valorile $V_{OL} = 0,4V$, $V_{OH} = 2,4V$. Deoarece în timp asignarea nivelurilor de tensiune pentru porţile TTL, respectiv şi pentru cele CMOS compatibile TTL, s-au ales valori de lucru sub $2,4V$ (fără a se lua în considerare evoluţia ulterioară spre tensiuni standard de alimentare sub $5V$), această potrivire a făcut ca mai târziu să fie posibilă asignarea şi pentru LVTTL a aceloraşi niveluri de tensiune ca şi pentru TTL, a se compara reprezentările din Figura 1.48-b şi 1.48-c. Astfel porţile CMOS cu nivelurile LVTTL (cu $V_{CC} = 3,3V$) pot comanda la ieşire porţi TTL (cu $V_{CC} = 5V$) atât timp cât încărcarea nu depăşeşte valorile I_{OLmax}, I_{OHmax} şi, la fel, ieşirile porţilor TTL pot comanda intrările LVTTL (dacă aceste intrări sunt realizate tolerante la $5V$). De fapt, nivelurile LVTTL corespund cu nivelurile de lucru de la seriile HCT, VHCT şi FCT ale familiei CMOS de $5V$.

De ce porţile LVTTL trebuie să fie realizate tolerante $5V$ la intrare? În general, pe intrare porţile au conectate diodele D1 şi D2, care au rolul de a shunta supratensiunile ce pot apărea la intrare, Figura 1.49-a. Dioda D1 shuntează la masă supratensiunile care au amplitudinea negativă iar D2 shuntează la bara de alimentare V_{DD} supraten-

siunile care au amplitudinea mai mare de $3,3V$. Dar la porţile logice TTL valorile tipice pentru V_{OH} depăşesc $3,3V$ ceea ce înseamnă că o poartă TTL ce comandă o poartă LVTTL poate avea ieşirea scurtcircuitată prin dioda D2 la bara $V_{DD} = 3,3V$ (deci un curent foarte mare). Soluţia? La structura de poartă LVTTL netolerantă pe intrare la $5V$, Figura 1.49-a, se elimină dioda D2 şi se obţine structura tolerantă pe intrare la 5V, Figura 1.14-b. Evident, tranzistorul acestei structuri tolerante trebuie să reziste la o tensiune de străpungere de minimum $5V$.

Dar porţile TSL de tip LVTTL trebuie să fie tolerante la 5V şi pe ieşire. Să considerăm că un buffer LVTTL de tip TSL are ieşirea conectată la o linie de magistrală la care sunt conectate şi alte porţi TTL de tip TSL, Figura 1.49-c. Bufferul LVTTL este în HZ când tensiunea aplicată pe poarta tranzistorului pMOS este $V_{DD} = 3,3V$ iar pe poarta tranzistorului nMOS este $0V$. Dacă linia de magistrală este comandată de o poartă TTL în stare H înseamnă că tensiunea de ieşire V_O a bufferului LVTTL este fixată de linia de magistrală la 5V, tensiune care este aplicată şi pe drenul tranzistorului pMOS notat cu Q. Tranzistorul Q având o tensiune pe dren de

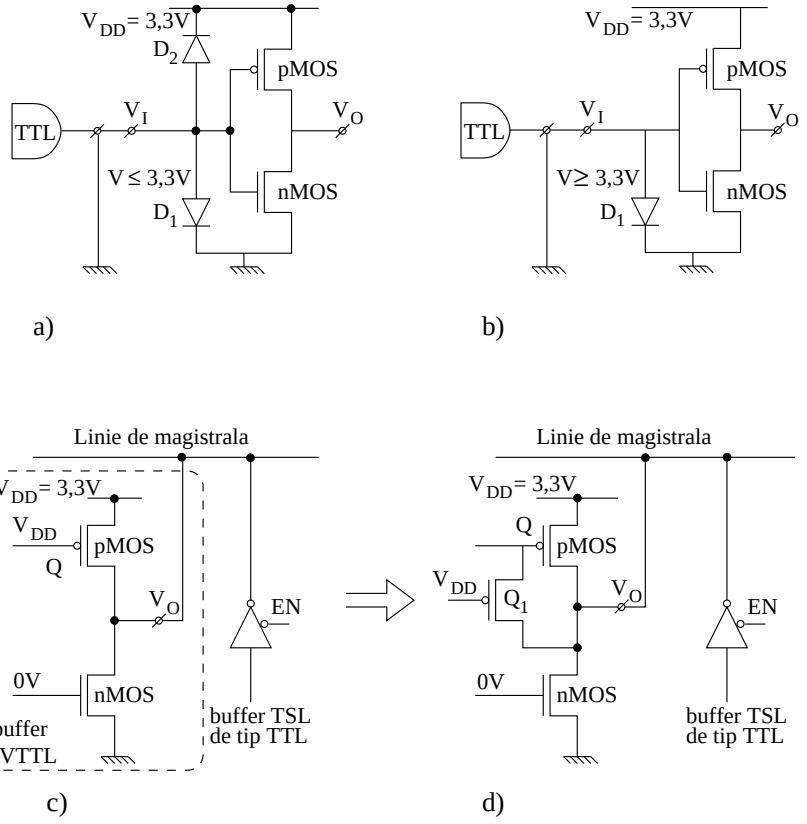


Figura 1.49 Porţi CMOS de $3.3V$**:** a) structură de poartă netolerantă la $5V$ pe intrare şi tolerantă la $5V$ (b); structură de buffer TSL netolerant la $5V$ pe ieşire (c) şi tolerant pe ieşire la $5V$ (d).

5V, faţă de tensiunea de $3,3V$ aplicată pe poarta sa, va conduce deci crează o cale de scurtcircuit de la sursa de 5V la sursa de $3,3V$. Se poate împiedica intrarea în conducţie a tranzistorului Q dacă între poarta sa şi dren se introduce un tranzistor pMOS notat cu Q1, Figura 1.49-d. Pe poarta tranzistorului Q1 se aplică o tensiune constantă $V_{DD} = 3,3V$. Când tensiunea de ieşire $V_O > V_{DD}$ tranzistorul Q1 intră în conducţie realizând o cale de impedanţă mică între ieşire şi poarta tranzistorului Q. Rezultă că potenţialul pe poarta tranzistorului Q nu poate să scadă sub potenţialul V_O aplicat pe drenul său, deci este blocat. O astfel de structură aplicată circuitului buffer LVTTL de tip TSL (alimentat la $V_{DD} = 3,3V$) îl face tolerant la $5V$ pe ieşire.

Concluzionând, interfaţarea TTL/LVTTL se poate realiza în condiţiile:

1. Interfaţarea LVTTL-TTL se poate face direct cu respectarea condiţiilor de încărcare la ieşire (nedepăşirea valorilor pentru I_{OHmax} şi I_{OLmax});

2. Interfaţarea TTL-LVTTL se poate realiza dacă intrările pe partea de LVTTL sunt intrări tolerante la 5V;

3. Porţile TTL şi LVTTL de tip TSL pot comanda împreună linii de magistrală dacă ieşirile LVTTL sunt tolerante la 5V.

CMOS la $2,5V$ şi $1,8V$. Migrarea de la $3,3V$ la $2,5V$ nu va fi simplă. Ieşirile de la familia CMOS de $3,3V$ pot comanda intrările de la familia CMOS de $2,5V$ atât timp cât intrările pe partea de $2,5V$ sunt tolerante la $3,3V$. Dar, comparând nivelurile logice de tensiune din Figura 1.48-c şi 1.48-d se observă că tensiunea $V_{OH} = 2V$ de la $V_{DD} = 2,5V$ este egală cu $V_{IH} = 2V$ de la $V_{DD} = 3,3V$ deci $M_H = 0$ când CMOS de $2,5V$ comandă CMOS de $3,3V$. Soluţionarea acestei deficienţe ar fi integrarea împreună cu poarta de $2,5V$ a unui circuit de deplasare de nivel spre $3,3V$, soluţie ce se aplică în prezent doar la μP şi ASIC-uri. Probabil când familia de porţi logice CMOS la $2,5V$ va deveni populară atunci şi porţile discrete vor avea înglobat şi o componentă standard - circuitul de deplasare de nivel.

Următorul pas va fi tranziţia de la $2,5V$ la $1,8V$. Analizând nivelurile logice de tensiune din Figura 1.48-d şi 1.48-e, când CMOS de $1,8V$ comandă CMOS de $2,5V$, rezultă o valoare negativă pentru $M_H = 1,45 - 1,7 = -0,25V$ deci, de asemenea, este necesar un circuit de deplasare de nivel.

1.5.4 Poarta de transmisie CMOS

Poarta de transmisie CMOS este compusă din două tranzistoare complementare, nMOS şi pMOS, având conectate în comun drenurile, la fel şi sursele, iar porţile lor sunt comandate separat. Semnalele de comandă pe cele două porţi ale tranzistoarelor sunt complementare, deci semnalul $\overline{S}$ pentru poarta tranzistorului pMOS se obţine de la ieşirea unui inversor la a cărui intrare s-a aplicat semnalul S pentru comanda porţii tranzistorului nMOS. Această structură cu reprezentările simbolice din Figura 1.50 are o funcţionare de comutator bidirecţional care este trecut în starea deschisă, ambele canale sunt blocate, prin semnalul de comandă $S = 0$, $\overline{S} = 1$, respectiv trecut în stare inchisă, cel puţin un canal conduce, prin semnalul de comandă $S = 1$, $\overline{S} = 0$. Pentru înţelegerea funcţionării acestui dispozitiv compus dintr-un **tranzistor** (**de trecere**) nMOS şi un tranzistor (de trecere) pMOS, conectate între potenţialele V_{in} şi V_O se va analiza funcţionarea separată a fiecărui tranzistor. Se va considera

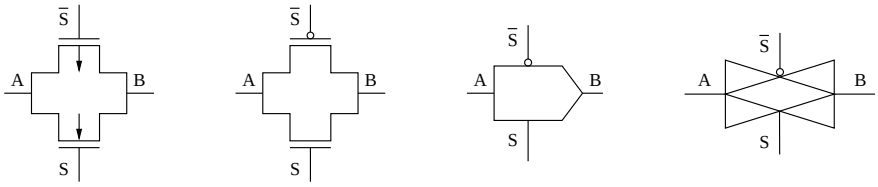


Figura 1.50 Diferite simboluri de reprezentare pentru poarta de transmisie CMOS.

potenţialul de alimentare V_{DD} ca fiind 1 logic iar potenţialul de masă V_{SS} ca 0 logic. Reamintim că ambele tranzistoare sunt blocate când $|V_{GS}| \leq |V_p|$ şi sunt în conducţie când $|V_{GS}| \geq |V_p|$. Tranzistoarele în conducţie sunt în zona liniară când $|V_{DS}| < |V_{GS} - V_p|$ şi în zona de saturaţie când $|V_{DS}| \geq |V_{GS} - V_p|$, Figura 1.32-b (dacă relaţiile sunt exprimate în modul atunci sunt corecte atât pentru nMOS cât şi pentru pMOS).

Pentru tranzistorul nMOS, Figura 1.51-a, când semnalul pe poartă are valoarea $S = 0$ canalul este blocat şi este trecut în conducţie pentru $S = 1$. Pentru comanda trecerii în conducţie, $S = V_{DD}$, la momentul $t = 0$, perechea de tensiuni $v_{I(0)}$, $v_{O(0)}$ poate fi: V_{DD}, V_{SS}; V_{SS}, V_{DD}; V_{SS}, V_{SS}; V_{DD}, V_{DD}. Considerând că la terminalele tranzistorului este aplicată prima pereche de valori, $v_I(0) = V_{DD}$, $v_O(0) = V_{SS}$, atunci la început canalul conduce în saturaţie $v_{DS} = V_{DD} - v_O(0) = V_{DD} - V_{SS} = V_{DD} > V_{GS} - V_{pn} = V_{DD} - V_{pn}$ condensatorul de sarcină C_L se încarcă până la tensiunea $v_O(\infty) = V_{DD} - V_{pn}$ când tranzistorul se blochează $v_{GS} = V_{DD} - v_O(\infty) = V_{DD} - (V_{DD} - V_{pn}) \leq V_{pn}$. Valoarea 1 logic de la intrare este transmisă degradată la ieşire $v_O(\infty) = V_{DD} - V_{pn}$; se spune că **tranzistorul nMOS transmite slab 1 logic**. Pentru $v_I(0) = V_{SS}$ şi $v_O(0) = V_{DD} - V_{pn}$ canalul intră în conducţie în regim liniar $v_{GS} = V_{DD} - v_I(0) = V_{DD} - V_{SS} = V_{DD}$, $v_{DS} = (V_{DD} - V_{pn}) - V_{SS} = V_{DD} - V_{pn} < v_{GS} = V_{DD}$ până când se ajunge la $v_{DS} = V_{SS}$, $v_{GS} = V_{DD}$ şi $I_{DS} = 0$; condensatorul C_L se descarcă până la $v_O(\infty) = V_{SS}$. Rezultă că valoarea 0 logic la intrare este transmisă fără degradare la ieşire; se spune că **tranzistorul nMOS transmite bine 0 logic**. Valorile tensiunilor de ieşire $v_O(t = \infty)$ pentru toate cele patru combinaţii iniţiale ale perechii $v_I(0)$, $v_O(0)$ sunt concentrate în tabelul din Figura 1.51-b.

Tranzistorul de trecere pMOS, Figura 1.51-c, este blocat pentru $\overline{S} = 1$ şi comandat în conducţie pentru $\overline{S} = 0$. Dacă la comanda în conducţie, $ovS = V_{SS}$, la terminalele tranzistorului perechea $v_I(0)$, $v_O(0)$ are valorile V_{DD}, V_{SS} atunci la început canalul conduce în saturaţie $|V_{DS}| = |V_{SS} - V_{DD}| > |V_{GS}| = |V_{SS} - V_{pp}|$ până când v_O creşte la valoarea $|V_{DD} - V_{pp}|$ şi apoi în regim liniar până când $v_O(\infty) = V_{DD}$, deci C_L se încarcă până la tensiunea $v_O(\infty) = V_{DD}$. Nivelul **1 logic este transmis prin canalul pMOS fără degradare**. Pentru cazul când $v_I(0) = V_{SS}$, $v_O(0) = V_{DD}$ condensatorul se descarcă prin rezistenţa canalului în conducţie până când $v_O = V_{pp}$, sub această valoare $|V_{GS}| < |V_{pp}|$ canalul se blochează $V_O(\infty) = |V_{pp}|$. Canalul **pMOS transmite cu degradare nivelul de 0 logic**. Tabelul din Figura 1.51-d prezintă modul cum se transmite semnalul prin tranzistorul pMOS pentru toate cele patru combinaţii de perechi $v_I(0)$, $v_O(0)$.

Poarta de transmisie CMOS, care este un comutator format din cele două canale

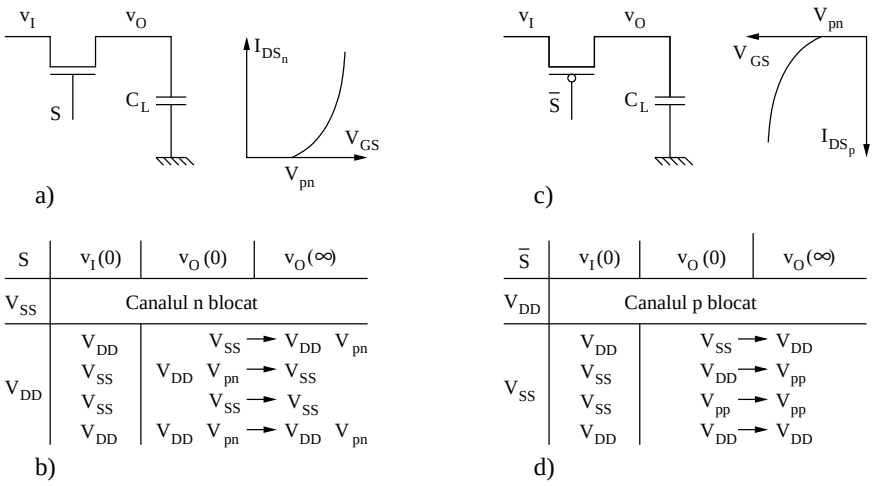


Figura 1.51 Tranzistoare de trecere: a) tranzistorul de trecere nMOS şi caracterstica de comandă $I_{DSn} = f(V_{GS})$; b) analiza modului de transfer al semnalului printr-un canal nMOS; c) tranzistorul de trecere pMOS şi caracteristica de comandă $I_{DSp} = f(V_{GS})$; d) analiza modului de transfer al semnalului printr-un canal pMOS.

n şi p conectate în paralel comandate cu semnale complementate, elimină dezavantajul de transmisie, degradarea de semnal, a fiecărui tranzistor de trecere, astfel că realizează o transmisie bună atât pentru 1 logic, prin canalul p, cât şi pentru 0 logic prin canalul n, Figura 1.52-b.

Deşi inversorul CMOS transmite bine nivelurile de 1 şi 0 logic, într-un lanţ de transmisie pentru un transfer corect, este necesar ca şi semnalele să fie suficient de puternice. În acest sens să considerăm că semnalul logic V_m, din punctul A de pe condensatorul C_m (de valoare mică) vrem să-l transmitem prin intermediul porţii de

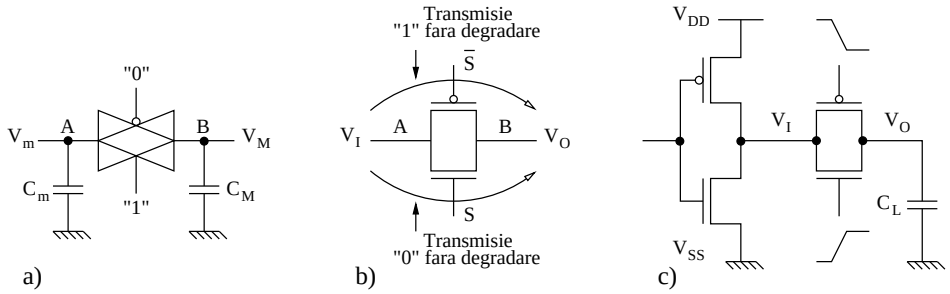


Figura 1.52 Poarta de transmisie CMOS: a) utilizarea porţii pentru controlul conexiunii între două puncte; b) poarta transmite fără degradare atât nivelul de 1 logic cât şi nivelul de 0 logic; c) înserierea porţii de transmisie la ieşirea unei porţi CMOS inversor este modalitatea normală de utilizare.

transmisie CMOS în punctul B unde este conectat condensatorul C_M (de valoare mare) pe care există semnalul logic V_M, Figura 1.52-a. Tensiunea rezultantă, când poarta conduce, este :

$$V_R = \frac{C_M \cdot V_M + C_m \cdot V_m}{C_M + C_m} \tag{1.81}$$

Dacă $C_m = 0,02pF$ (o sarcină standard în tehnologia integrată de $0,5\mu m$) şi $V_m = 5V$ iar $C_M = 0,2pF$ (zece sarcini standard) şi $V_m = 0V$, cu relaţia 1.81, rezultă tensiunea pe ambele capacităţi egală cu $V_R = 0,45V$. Aceasta însemnă că semnalul din punctul A cu valoarea logică 1 nu s-a transmis corect în punctul B, mai mult, semnalul din B a deteriorat semnalul din A şi aceasta pentru că semnalul V_m nu a fost suficient de puternic să forţeze semnalul V_M din B. Corectitudinea transferului prin poarta de transmisie se poate realiza prin:

1- izolarea nodului A de nodul B prin introducerea unui buffer;

2- realizarea unui semnal în punctul A suficient de puternic. Uzual, realizarea unui semnal destul de puternic la intrarea unei porţi de transfer CMOS rezultă prin obţinerea semnalului de intrare la poarta de transfer de la ieşirea unui inversor CMOS, ca în Figura 1.52-c. În implementarea sistemelor logice se intermixează porţile de transmisie cu cele pe bază de inversor CMOS; prin utilizarea şi a porţilor de transmisie rezultă un număr mai redus de tranzistoare.

Implementarea operatorilor logici pe bază de porţi de transmisie CMOS se face într-un mod similar ca la realizarea schemelor logice pe bază de contacte. Dacă variabila de intrare în poarta de transmisie (variabila de trecere) este x iar poarta este comandată în deschidere cu variabila de control y se obţine produsul logic xy, iar dacă variabila de control este $\overline{y}$ se obţine produsul logic $x\overline{y}$. În acest mod de implementare, pentru funcţia $x \oplus y = x\overline{y} + \overline{x}y$, considerând x ca variabila de trecere iar y ca variabilă de control, se obţine structura din Figura 1.53-a. Implementările de bază de tranzistoare de trecere se fac la fel de simplu ca şi cele cu relee, de exemplu, în Figura 1.53-b este implementat operatorul XOR. Bazat pe această simplă implementare a operatorului XOR se poate realiza un circuit pentru calculul identităţii a două cuvinte, de exemplu $X = x_3x_2x_1x_0$ şi $Y = y_3y_2y_1y_0$ ca în Figura 1.53-d. Acest circuit realizează funcţia de identificare cuvinte $I_{dc} = \overline{(x_3 \oplus y_3) + (x_2 \oplus y_2) + (x_1 \oplus y_1) + (x_0 \oplus y_0)}$, generând $I_{dc} = 1$ numai când există identitate. Structural este o pseudopoartă NOR (Figura 1.46-a) ale cărei intrări se obţin de la porţi XOR.

Extinzând aceste reguli simple se poate obţine o structură de circuit pe bază de porţi de transmisie care poate modela toate cele 16 funcţii de două variabile x, y. Fiecare funcţie de două variabile $f_i^2(x,y)$ se poate obţine, din tabelul de adevăr din Figura 1.2-a, ca o formă canonică normală disjunctivă, relaţia 1.10, sau pentru o exprimare sintetică se poate folosi relaţia matriceală:

$$\begin{Vmatrix} f_0^2(x,y) \\ f_1^2(x,y) \\ \vdots \\ f_i^2(x,y) \\ \vdots \\ f_{14}^2(x,y) \\ f_{15}^2(x,y) \end{Vmatrix} = \begin{Vmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{Vmatrix} \cdot \begin{Vmatrix} \overline{x_1}\,\overline{x_0} \\ \overline{x_1}\,x_0 \\ x_1\,\overline{x_0} \\ x_1\,x_0 \end{Vmatrix} \Rightarrow \left\|f_i^2\right\| = \|d_{ij}\| \cdot \|P\| \tag{1.82}$$

adică matricea funcţiilor $\|f_i^2\|$ este produsul dintre matricea coeficienţilor $\|d_{ij}\|$, $i = 0, 1, \cdots, 15$, $j = 0, 1, 2, 3$ ai funcţiei cu matricea termenilor produs de două variabile $\|P\|$. Funcţia $f_i^2(x, y)$ are forma:

$$f_i^2(x_1, x_0) = d_{i0}\overline{x_1}\,\overline{x_0} + d_{i1}\overline{x_1}\,x_0 + d_{i2}x_1\,\overline{x_0} + d_{i3}x_1\,x_0 \tag{1.83}$$

Analizând această relaţie se deduce că implementarea constă din patru ramuri în paralel, la fiecare din ramuri se aplică unul din coeficenţii binari ai funcţiei (d_{i0}, d_{i1}, d_{i2}, d_{i3}), iar pe fiecare ramură sunt câte două porţi de transmisie înseriate care realizează unul din termenii canonici produs ($\overline{x_1}\,\overline{x_0}, \overline{x_1}\,x_0, x_1\,\overline{x_0}, x_1\,x_0$), Figura 1.53-c. De exemplu, pentru setul de coeficienţi: ($d_0 = 0, d_1 = 1, d_2 = 1, d_3 = 0$) ai funcţiei f_6^2 (vezi Figura 1.2) circuitul va implementa operatorul XOR, iar pentru setul: $d_0 = 0$, $d_1 = 1$, $d_2 = 1$, $d_3 = 0$ se obţine modelarea operatorului NAND. Pentru fiecare combinaţie de patru biţi aplicată la intrările acestui circuit se va modela una din cele 16 funcţii de două variabile. Se va vedea în secţiunea 2.4.4 că acest circuit este, de fapt, o structură de multiplexor, MUX 4:1.

1.5.5 Circuite logice dinamice

La circuitele logice prezentate până acum funcţia logică realizată corespunde regimului static de funcţionare, adică este asociată cu punctul static de funcţionare. La o poartă logică după un (anumit) timp de propagare (întârziere pe poartă), de la aplicarea intrărilor, se obţine la ieşire o stare logică stabilă care se menţine atât timp cât

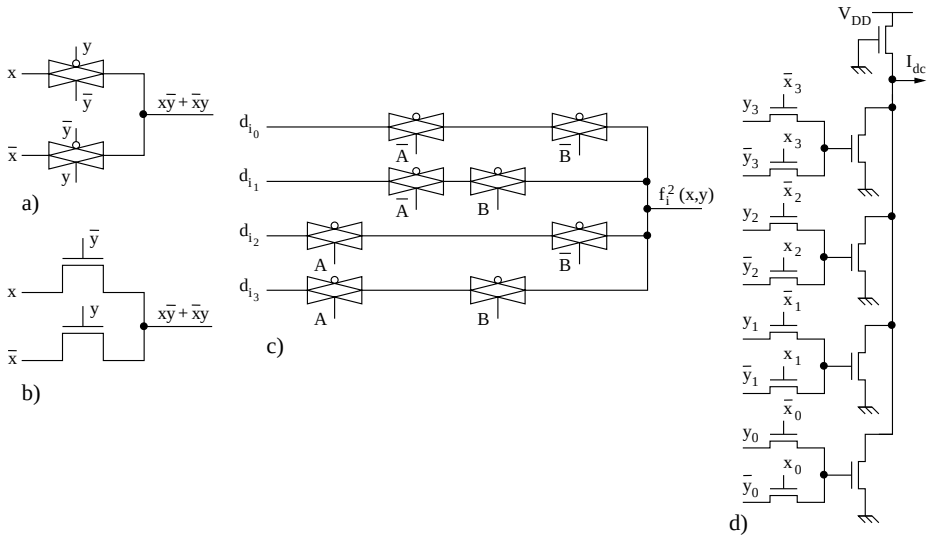


Figura 1.53 Exemple de circuite implementate cu porţi de transmisie CMOS şi tranzistoare de trecere: Operatorul XOR cu porţi de transmisie (a) şi cu tranzistoare de trecere (b); c) structură programabilă pentru toate cele 16 funcţii de două variabile, $f_i^2(x, y)$; d) structură de circuit pentru determinarea identităţii a două cuvinte de patru biţi ($X = x_3x_2x_1x_0$, $Y = y_3y_2y_1y_0$).

se menţin valorile intrărilor şi tensiunea de alimentare se păstrează. Implementările acestea, referite ca statice, în general necesită un număr mare de tranzistoare şi care, în plus, pot determina timpi de întârziere considerabili.

Pentru implementările de înaltă performanţă şi de densitate mare de integrare, unde puterea disipată, întârzierea pe circuit şi aria consumată pe siliciu sunt cerinţe majore, se recomandă aşa numitele circuite logice dinamice care oferă unele avantaje în raport cu circuitele logice statice. Totuşi, aceste avantaje pot fi umbrite de faptul că, în raport cu circuitele statice, imunitatea la zgomot este mai scăzută, ceea ce a determinat ca unele circuite dinamice să fie modificate pentru o funcţionare pseudostatică. Funcţionarea circuitelor logice dinamice se bazează pe stocarea temporară (tranzistorie) a unei sarcini electrice pe o capacitate parazită a unui nod de circuit. Tensiunea tranzitorie pe acea capacitate parazită, în intervalele de timp când se situează în intervalele ΔV_{IH}, ΔV_{IL}, este considerată ca semnal logic de intrare pentru comanda circuitului a cărui intrare este conectată la acel nod (în aceste intervale de timp nu mai este necesar să se aplice semnal de intrare din exterior).

Circuitele logice dinamice de bază, pentru porţile dinamice nMOS şi pentru cele CMOS, sunt reprezentate respectiv în Figura 1.54-a şi 1.54-b. Pentru ambele circuite

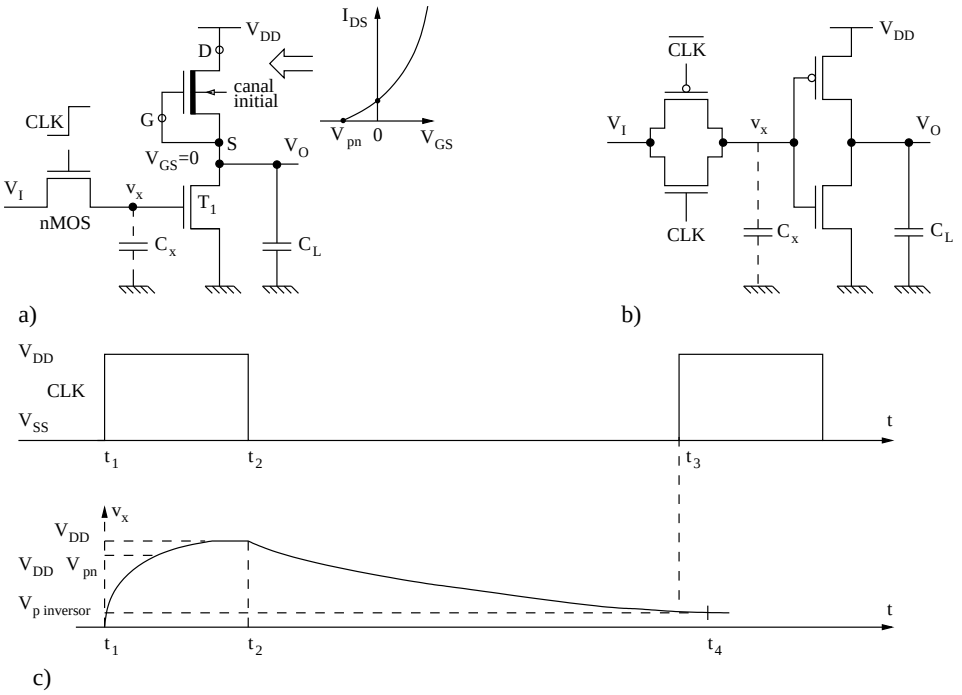


Figura 1.54 Circuite logice dinamice: a) structură de inversor dinamic (cu sarcină tranzistor cu canal iniţial) şi comutator pe intrare – tranzistor de trecere; b) structură de inversor dinamic CMOS cu comutator pe intrare – poartă de trecere; c) diagramele de timp pentru semnalul de ceas, CLK, şi pentru variaţia tensiunii v_x pe capacitatea parazită C_x a nodului de intrare.

tensiunea V_x, care va comanda în intervalele tranzitorii inversorul cu tranzistor de comandă nMOS şi sarcina cu canal iniţial (depletion-load nMOS) sau inversorul CMOS, este tensiunea rezultată pe capacitaţile parazite C_x ale porţilor de intrare. Inversorul cu tranzistor de comandă nMOS (sau pMOS) are o structură de principiu similară ca şi inversorul bipolar, Figura 1.21-d, un element comandat-tranzistorul-şi o sarcină. La fel ca şi cel bipolar, sarcina se poate realiza cu o rezistenţă de sarcină în dren, dar în tehnologia integrată o rezistenţă de sarcină obţinută prin difuzie ocupă o suprafaţă pe siliciu echivalentă cu suprafaţa consumată, pentru implementarea, a zeci de tranzistoare. Evitarea unui astfel de consum mare de suprafaţă de siliciu poate fi obţinută prin utilizarea ca rezistenţă de sarcină rezistenţa unui canal nMOS în conducţie, deci un tranzistor utilizat ca rezistenţă. Există două modalităţi de realizare a sarcinii pe bază de tranzistor: fie cu un tranzistor nMOS cu canal indus în regim de saturaţie, fie un tranzistor nMOS cu canal iniţial.

Inversorul cu rezistenţa de sarcină canal în saturaţie se obţine prin conectarea între tranzistorul inversor (de comandă) şi bara V_{DD} a unui tranzistor (de sarcină) nMOS a cărui poartă se conectează la V_{DD}. Prin această conexiune poarta şi drenul fiind echipotenţiale, $V_{GS\,\text{sarcină}} = V_{DD} - V_S = V_{DD} - V_O$, $V_{DS\,\text{sarcină}} = V_D - V_S = V_{DD} - V_O$ rezultă că totdeauna $V_{GS\,\text{sarcină}} = V_{DS\,\text{sarcină}}$ iar $V_{DS\,\text{sarcină}} > V_{GS\,\text{sarcină}} - V_{pn\,\text{sarcină}}$, deci canalul este permanent în saturaţie, Figura 1.32-b. Căderea de tensiune pe canal $V_{DS\,\text{sarcină}}$ nu poate scădea sub valoarea de prag de deschidere $V_{pn\,\text{sarcină}}$ deoarece atunci canalul s-ar bloca (s-a explicat în 1.5.4 că tranzistorul nMOS transmite deteriorat 1 logic). Aceasta înseamnă că tensiunea maximă de ieşire în starea H nu poate atinge valoarea barei de alimentare V_{DD}, $V_{Omax} = V_{DD} - V_{pn\,\text{sarcină}} < V_{DD}$.

Inversorul cu tranzistor inversor nMOS şi cu sarcină tranzistor nMOS cu canal iniţial poate genera la ieşire o tensiune maximă egală cu V_{DD}. Tranzistorul nMOS cu canal iniţial are, chiar când $V_{GS} = 0$, permanent o sarcină electrică negativă în canal (**canalul iniţial**), deci există un curent dacă $V_{DS} > 0$; această stare se poate observa din caracteristica de comandă $I_{DS} = f(V_{GS})$, desenată lângă tranzistor în Figura 1.54-a. Se observă că din această caracteristică valoarea tensiunii de prag de deschidere a tranzistorului este negativă, $-V_{pn}$, iar când grila este conectată la sursă, $V_{GS} = 0$, canalul este în conducţie, $I_{DS} \neq 0$. Deci atunci când tranzistorul inversor este blocat tensiunea de ieşire V_{OH} nu mai este limitată la valoarea $V_{DD} - V_{pn}$ (ca la inversorul cu sarcină tranzistor saturat) ci poate creşte până la valoarea V_{DD}. Canalul iniţial se obţine în procesul tehnologic printr-o difuzie suplimentară, în zona canalului, de impurităţi donoare care generează (permanent) canalul iniţial; mărind tensiunea de comandă pe poartă V_{GS} peste valoarea zero sarcina negativă a canalului iniţial este întărită suplimentar printr-o sarcină indusă în canal.

Pentru cele două tipuri de inversoare, din Figura 1.54-a şi 1.54-b, tensiunea de intrare V_I se aplică pe grilele de intrare prin intermediul unui comutator, care este un tranzistor de trecere nMOS pentru inversorul cu sarcină cu canal iniţial, iar pentru inversorul CMOS este o poartă de transmisie CMOS. Comutatorul este comandat periodic cu un semnal de ceas, CLK; deci când $CLK = 1$ comutatorul este închis iar tensiunea de intrare V_I va forţa încărcarea sau descărcarea capacităţii parazite C_x până la un potenţial $v_x = V_{DD}$ sau $v_x = 0$ după cum la intrare V_I a avut valorile 1 sau 0 logic. Aceste valori pentru V_x se obţin doar pentru comutatorul poarta de transmisie dar pentru comutatorul tranzistor de trecere se obţin valorile $v_x = V_{DD} - V_{pn}$, $v_x = 0$ (tranzistorul de trecere nMOS transmite cu deteriorare 1 logic

şi fără deteriorare 0 logic, iar poarta de transmisie transmite fără deteriorare atât 1 logic cât şi 0 logic). Când semnalul de ceas devine inactiv $CLK = 0$, comutatorul este deschis, condensatorul C_x rămâne încărcat sau descărcat, iar tensiunea sa V_x va comanda ieşirea inversorului respectiv în starea logică 0, $V_O = V_{OL}$ sau 1, $V_O = V_{OH}$.

Sarcina stocată pe C_x scade în timp datorită unui curent de descărcare. Curentul de descărcare spre masă nu este prin stratul de oxid de sub poarta tranzistorului/tranzistoarelor inversorului, care prezintă o rezistenţă extrem de mare ($\sim 10^{14}\Omega$) ci prin comutatorul de pe intrare. Curentul de descărcare prin acest comutator are două componente: 1- curentul prin joncţiunea pn polarizată invers, formată între zona difuzată a terminalului tranzistorului de trecere legată la C_x şi substratul în care este realizată această zonă; 2- curentul de conducţie de sub prag; dar preponderentă este prima componentă. Când $CLK = 0$ $v_x = V_{DD}$ sau $V_{DD} - V_{pn(\text{tranzistor de trecere})}$ inversoarele generează la ieşire V_{OL} şi această valoare logică se păstrează până când v_x descreşte până la tensiunea de prag a tranzistorului inversor $V_{p(\text{inversor})}$, adică momentul t_4 din Figura 1.54-c. Evident, funcţionarea corectă a inversorului dinamic se realizează numai atunci când semnalul CLK devine activ înainte ca v_x să scadă la $V_{p(\text{inversor})}$, $t_3 < t_4$. Atât pentru comutatorul tranzistor de trecere cât şi pentru comutatorul poartă de transmisie variaţia în timp a tensiunii $v_x(t)$, deci a intervalului când aceasta ajunge la pragul $V_{p(\text{inversor})}$, se poate determina analitic pe baza circuitelor echivalente din Figura 1.51-a, 1.51-b sau printr-o modelare în SPICE. Dar un calcul orientativ se poate efectua foarte rapid; de exemplu pentru $C_x = 0,1pF$, $I_{\text{descărcare}} = 0,1pA$ rezultă o variaţie $\Delta V_x/\Delta t = I/C = 0,1 \cdot 10^{-12}pA/0,1 \cdot 10^{-12}pF = 1V/s$, deci o scădere de la $5V$ la $0V$ în $5s$ ($\frac{5V}{1V/s} = 5s$). La această viteză de scădere de $1V/s$ a tensiunii V_x o frecvenţă de exemplu $500Hz$ ($T = 2ms$) pentru semnalul de ceas, ca să reîmprospăteze sarcina pe condensator, este cu mult peste suficient. Semnalul de ceas mai este referit, în acest caz, şi ca semnal de reîmprospătare (refreshment).

Exemplul 1.19 Pentru lanţurile de tranzistoare de transmisie nMOS din Figura 1.55-a şi 1.55-b să se determine tensiunea la ieşire.

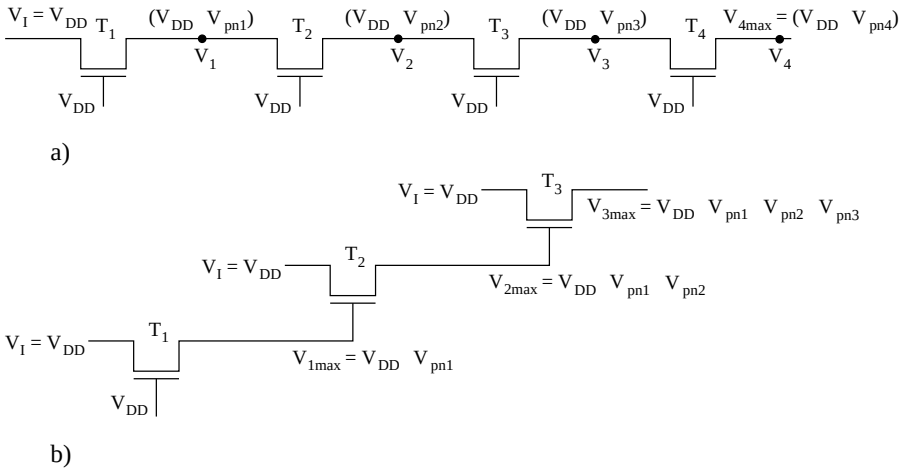


Figura 1.55 Structuri de lanţuri cu tranzistoare de trecere nMOS

Soluţie. Pentru lanţul obţinut din înserierea a patru tranzistoare de trecere nMOS identice ($V_{pn1} = V_{pn2} = V_{pn3} = V_{pn4}$) se consideră că $V_I = V_{DD}$ şi toate cele patru noduri sunt descărcate ($V_1 = V_2 = V_3 = V_4 = 0$). Cu aceste consideraţii tranzistorul T1 operează în saturaţie $V_{DS1} > V_{GS1} - V_{pn1}$ deci căderea de tensiune pe primul canal nu poate fi mai mică decât V_{pn1} rezultă că tensiunea V_1 în nodul 1 nu poate depăşi valoarea $V_{1max} = (V_{DD} - V_{pn1})$. Tranzistorul T2 funcţionează la limita de saturaţie deci tensiunea maximă în nodul 2 va fi $V_{2max} = (V_{DD} - V_{pn2})$. Extinzând la T4 tensiunea în nodul 4 va fi mai mică cu V_{pn4} decât V_{DD}. Rezultă că indiferent câte tranzistoare sunt înseriate când la intrare se aplică $V_I = V_{DD}$ tensiunea pe fiecare nod intern se va stabiliza la o valoare egală cu V_{pn} sub V_{DD} oricare a fost tensiunea iniţială pe nod. Ca o imagine intuitivă se poate considera un singur tranzistor de trecere, cu de n ori lungimea de canal, pe care este o cădere egală cu V_p.

Pentru structura din Figura 1.55-b când ieşirea de la tranzistorul anterior se aplică pe poarta tranzistorului următor, fiecare tensiune de ieşire poate creşte doar până la valoarea egală cu V_{pn} sub tensiunea aplicată pe poartă deci $V_{1max} = V_{DD} - V_{pn}$, $V_{2max} = V_{DD} - V_{pn} - V_{pn}$, $V_{3max} = V_{DD} - V_{pn} - V_{pn} - V_{pn} - \ldots V_{nmax} = V_{DD} - nV_{pn}$.

Circuite CMOS dinamice. Implementarea următoarei funcţii de opt variabile $F = AB + (C + D)(E + F) + GH$ pe o structură CMOS statică convenţională necesită 16 tranzistoare ca în Figura 1.56-a. Aceeaşi implementare pe o structură CMOS

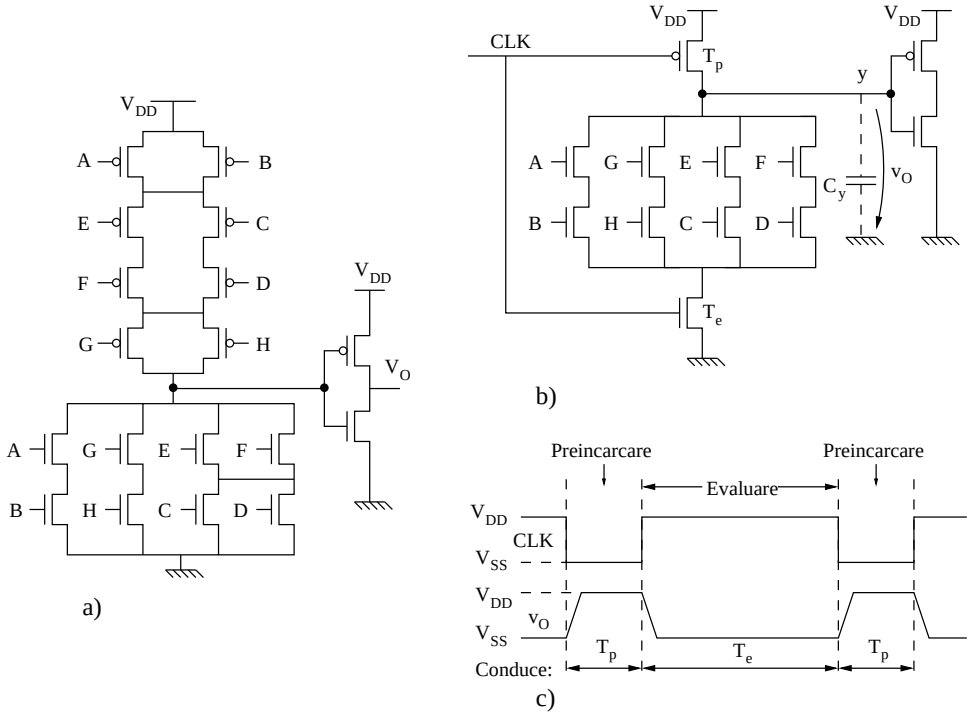


Figura 1.56 Porţi CMOS pentru implementarea funcţiei $F = AB + (C + D)(E + F) + GH$: a) cu o structură statică; b) cu o structură dinamică; c) diagramele de semnale pentru fazele de preîncărcare şi evaluare în funcţionarea porţii dinamice.

dinamică, reprezentată în Figura 1.56-b, necesită doar 10 tranzistoare; structura are asemănare cu cu pseudo-poarta CMOS din Figura 1.46-a. Această poartă dinamică constă dintr-o reţea de n tranzistoare al cărui nod de ieşire y, caracterizat de o capacitate parazită C_y, este preîncărcat la tensiunea V_{DD} prin tranzistorul pMOS, T_p. Apoi condensatorul C_y este descărcat/evaluat condiţionat (de intrările de comandă ale porţii) prin unele din tranzistoarele reţelei nMOS şi prin tranzistorul nMOS, T_e. Alternativ, se poate realiza o structură cu un nMOS tranzistor pentru preîncărcare la V_{DD}, un pMOS tranzistor pentru descărcare la V_{SS} şi o reţea de p de tranzistoare.

Faza de preîncărcare a capacităţii C_y prin tranzistorul T_p până la tensiunea $v_O = V_{DD}$ se realizează pe intrevalul de timp când semnalul de ceas are valoarea $CLK = 0$, Figura 1.55-c, totodată se aplică şi valorile variabilelor pe intrările porţii. Apoi, pe intervalul de timp când semnalul de ceas are valoarea $CLK = 1$, iar T_p este blocat dar T_e conduce, se realizează evaluarea valorii funcţiei. În funcţie de valorile logice ale variabilelor de intrare se poate realiza sau nu o cale de scurtcircuit la masă pentru tensiune $v_O = V_{DD}$, deci valoarea funcţiei este evaluată la 0 sau 1 logic. Pe nodul de ieşire y poate obţine doar o tranziţie de la 1 la 0 sau rămâne în starea 1 logic.

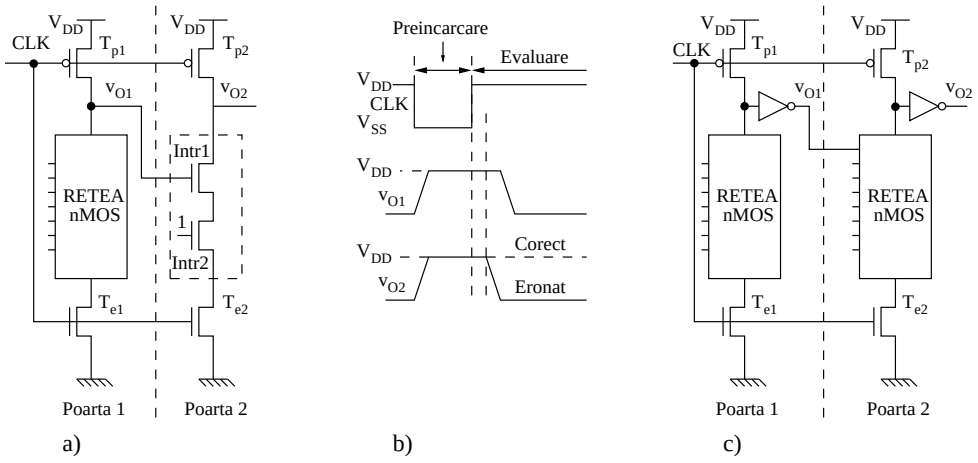


Figura 1.57 Porţi CMOS dinamice: a) înserierea porţilor şi diagrama semnalelor la ieşirea lor (b); c) înserierea porţilor de tip domino.

Funcţionarea unei singure porţi CMOS dinamice, ca în Figura 1.56-b, este corectă, dar când sunt înseriate mai multe porţi ca în Figura 1.57-a poate apare o funcţionare eronată. În această înseriere Poarta2 se consideră a fi un NAND cu două intrări ($Intr1$, $Intr2$). Pe durata fazei de preîncărcare, prin tranzistoarele T_{p1} şi T_{p2}, tensiunile de ieşire v_{O1} şi v_{O2} ajung la valoarea V_{DD} iar valorile variabilelor de intrare sunt aplicate la Poarta1, a doua intrare ($Intr2$) pentru Poarta2 se consideră că are aplicată valoarea 1 (pe prima intrare ($Intr1$) a Porţii2 se aplică v_{O1}). Se presupune că intrările pentru Poarta1 determină pentru aceasta, prin unele tranzistoare în conducţie din reţeaua n şi prin T_{e1}, o cale de scurtcircuit la masă pe durata fazei de evaluare, deci v_{O1} devine 0 logic, valoare evaluată corect pentru Poarta1.

Dar, la începutul fazei de evaluare tensiunea de ieşire, care se aplică pe prima

intrare de la Poarta2, este încă la nivelul $v_{O1} = v_{DD}$ deci ieşirea v_{O2} a porţii NAND va fi comandată în valoarea zero, $v_{O2} = 0$. Această valoare logică 0 de la ieşirea porţii NAND este eronată deoarece la sfârşitul fazei de evaluare ar trebui să fie 1 logic ($v_{O2} = V_{DD}$) pentru că ieşirea corectă de la Poarta1 este 0 logic, Figura 1.57-b. Rezultă că posibilitatea de funcţionare eronată la această poartă când este înseriată cu alte porţi s-ar părea că elimină celelalte avantaje (putere disipată redusă, valoarea ridicată pentru marginea de zgomot, număr redus de tranzistoare şi viteză ridicată). Pentru eliminarea posibilităţii de funcţionare eronată au fost realizate o mulţime de structuri [Weste 2001][Kang 1996] CMOS dinamice de mare performanţă dar care nu prezintă acest dezavantaj, dintre acestea se va prezenta doar structura de poartă CMOS (dinamică) domino.

O **poartă logică CMOS domino** se compune dintr-o poartă CMOS dinamică la care se conectează pe ieşire un inversor CMOS static, Figura 1.57-c. Să explicăm necesitatea introducerii inversorului. Poarta1 poate realiza la ieşire doar o comutaţie de la 1 la 0 şi trebuie să comande tranzistorul Porţii2 numai când are evaluată pentru ieşire valoarea 0. Deoarece tranzistorul nMOS al Porţii2 se comandă cu 1 logic este normal ca între ieşirea Porţii1 şi intrarea Porţii2 să se introducă un inversor CMOS static; cu această completare Poarta2 este comandată numai atunci când, după evaluare, Poarta1 comută din 1 în 0. Când Poarta1 comută din 1 în 0, comandă, prin inversor, ca Poarta2 să poată comuta din 1 în 0, care la fel comandă Poarta3 să comute din 1 în 0 şi aşa mai departe până la poarta n-a (propagare de tip domino!). Evident, această succesiune de "căderi" ale tensiunilor de ieşire ale porţilor din 1 în 0 – similar ca la un lanţ domino – trebuie să se propage într-un interval de timp care să nu fie mai lung decât faza de evaluare, $CLK = 1$. Dar şi poarta domino prezintă inconveniente: primul, poarta este la ieşire (din inversorul static CMOS) o structură neinversoare (necesară pentru un lanţ domino) iar dacă este necesară o inversare trebuie introdus încă un inversor static; al doilea, distribuţia sarcinii C_y pe nodurile intermediare ale tranzistoarelor nMOS înseriate (din reţeaua nMOS) în timpul fazei de evaluare, poate produce o ieşire eronată.

Distribuţia sarcinii nodului de ieşire pe nodurile intermediare din reţeaua nMOS apare în felul următor. Să considerăm poarta domino din Figura 1.58-a. În intervalul de timp de preîncărcare tensiunea v_y în nodul de ieşire y, pe capacitatea parazită C_y, atinge valoarea V_{DD}. Se poate ca unele semnale de comandă pe porţile tranzistoarelor nMOS înseriate să nu fie aplicate corect; semnalele de intrare pe porţile tranzistoarelor trebuie aplicate numai pe durata fazei de preîncărcare, este incorectă modificarea valorii acestor semnale pe durata fazei de evaluare. Să presupunem că în faza de evaluare pe porţile primelor două tranzistoare nMOS de lângă nodul de ieşire y al Porţii1 se aplică semnalele 1 logic care comandă aceste două tranzistoare în conducţie; restul tranzistoarelor înseriate din reţeaua nMOS sunt comandate în blocare, T_{e1} conduce, deci nu există o cale de scurtcircuit a sarcinii de pe C_y la masă, ieşirea Porţii1 nu comută din 1 în 0. Deoarece primele două tranzistoare T1 şi T2 conduc, sarcina de pe C_y se distribuie şi pe condensatoarele parazite C_1 şi C_2 din nodurile dintre tranzistoarele T1 şi T2 respectiv T2 şi T3. În urma distribuirii sarcinii tensiunea în nodul de ieşire va avea valoarea $v_y = \left[\frac{C_y}{C_y+C_1+C_2}\right] V_{DD}$. Pentru $V_{DD} = 5V$, $C_y = C_1 = C_2$ rezultă $V_y = 1,66V$. Valoarea tensiunii $v_y = 1,66V$, când tensiunea la ieşirea Porţii1 ar fi trebuit să rămână la valoarea $v_y = V_{DD} = 5V$, este sub valoarea de prag de comutaţie ($\approx 2,5V$) a inversorului CMOS static, deci ieşirea

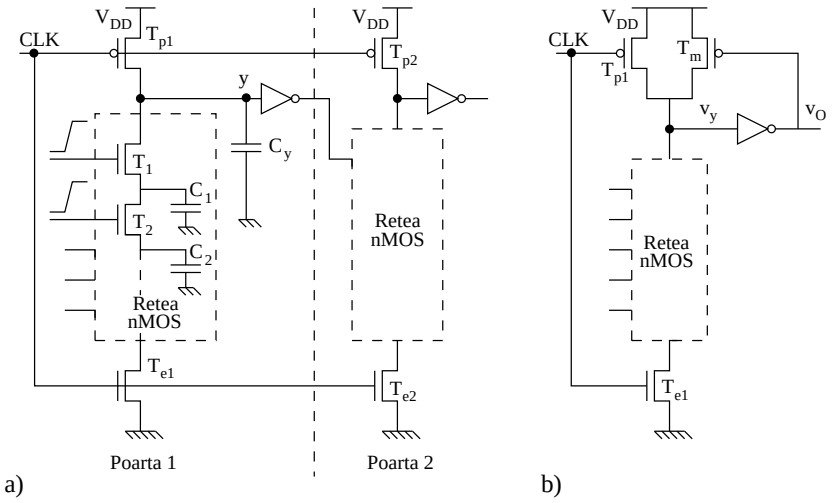


Figura 1.58 Poarta CMOS domino: a) dacă nu se respectă aplicarea semnalelor de intrare, numai în faza de preîncărcare, atunci redistribuţia sarcinii de pe C_y poarta produce o funcţionare eronată pentru o poartă domino; b) eliminarea posibilităţii de funcţionare eronată prin introducerea unui circuit cu reacţie (tranzistorul T_m) pentru menţinerea $v_y = V_{DD}$.

acestuia comută din 0 în 1 şi se comandă intrarea Porţii2. Iată că prin distribuţia de sarcină scăderea tensiunii v_{O1} produce o comandă eronată pentru Poarta2.

Pentru eliminarea comutaţiei eronate datorată distribuţiei sarcinii din nodul de ieşire se poate aplica una din următoarele soluţii:

1. Se realizează inversorul CMOS static cu un prag de comutaţie foarte scăzut;

2. Se utilizează câte un tranzistor de preîncărcare, similar lui T_{p1}, pentru fiecare nod din reţeaua de tranzistoare nMOS, în felul acesta capacitatea din fiecare nod se încarcă la tensiunea V_{DD} (deci sarcina de pe C_y nu se mai distribuie).

3. Se realizează un tranzistor pMOS de menţinere T_m ca în Figura 1.58-b. Acesta este un tranzistor "slab", adică forţează un curent foarte mic în nodul de ieşire deoarece are un raport de forma W/L de valoare foarte mică, deci poate menţine tensiunea v_y în starea H în acest nod numai dacă nu există o cale putenică de scurtcircuit de la nod la masă prin lanţul de tranzistoare din reţeaua nMOS. Se observă că tranzistorul T_m este în conducţie doar când tensiunea în nod este de nivel H; prin reacţia inversă, realizată prin intermediul inversorului static, se aplică un semnal de nivel L pe poarta lui T_m menţinându-l în conducţie.

Performanţele tranzitorii ale porţii domino pot fi îmbunătăţite prin reducerea timpului de descărcare a capacităţii C_y la masă prin linia de tranzistoare nMOS înseriate. O soluţie în acest sens este realizarea layoutului tranzistoarelor înseriate ca în Figura 1.59-b, unde lăţimea de canal creşte în sensul de la tranzistorul conectat la nod spre tranzistorul conectat la masă, deşi această geometrie pare a fi contraintuitivă.

Prin micşorarea raportului de formă W/L al unui tranzistor capabilitatea de a furniza curent scade (rezistenţa echivalentă a canalului creşte), la fel scad şi capacităţile parazite. Explicaţia în cazul prezentei geometrii este: dacă lungimea canalelor nMOS înseriate este destul de mare creşterea rezistenţei prin micşorarea unor lăţimi de canal este neînsemnată pe când scăderea capacităţii parazite este semnificativă şi deci per total rezultă o constantă de timp micşorată pentru descărcare. În plus, se poate realiza pentru inversorul static de pe ieşire, care după faza de evaluare poate comuta de la 0 la 1, o viteză mărită a comutaţiei $L-H$ de la ieşirea sa prin dimensionarea tranzistorului nMOS al său cu W mărit.

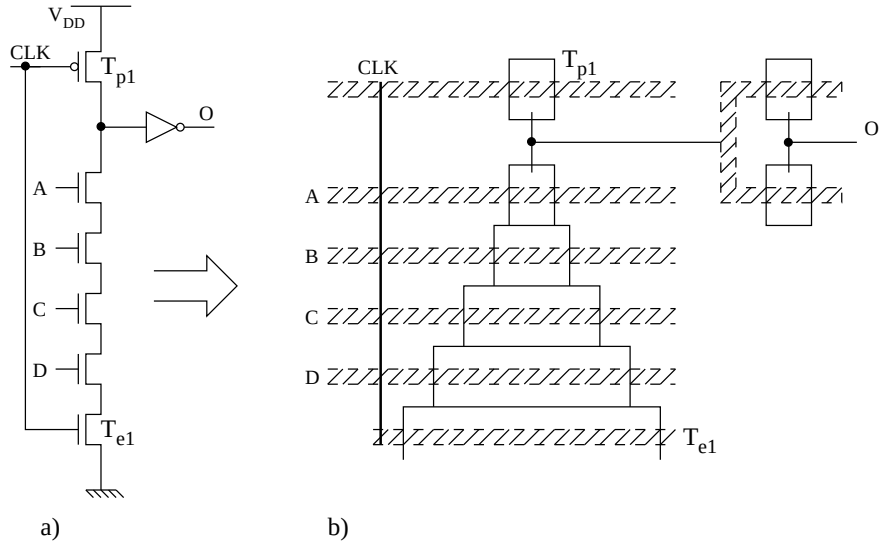


Figura 1.59 Poarta CMOS dinamică: a) structură tipică de poartă domino; b) variantă de layout (simplificat) care poate realiza un timp redus în etapa de evaluare pentru poarta domino.

Poarta domino, de fapt, este realizată din două etaje, poarta dinamică urmată de inversorul CMOS static, Figura 1.59-a. Uneori poarta domino nu este sincronizată în faza de evaluare, deci tranzistorul T_e este eliminat; acest tranzistor încetineşte procesul de descărcare al capacităţii C_y dar elimină în faza de preîncărcare orice cale de conducţie între V_{DD} şi V_{SS}. Tipic o poartă domino este de $1,5 \div 2$ ori mai rapidă decât o poartă CMOS statică.

1.5.6 Metoda efortului logic

Metoda efortului logic a fost concepută de către Ivan Sutherland şi Bob Sproull (1991), ca un instrument în proiectarea circuitelor CMOS în funcţie de întârzierea pe circuit, ulterior a fost publicată o monografie pe acest subiect [Sutherland ′99]. Prezentul material este o sinteză a metodei efortului logic bazată pe această monografie.

Metoda efortului logic în raport cu alte metode de proiectare (simulare şi corectare, fan-out egal pe etaj, optimizarea numerică, întârziere egală pe nivel) este uşor de aplicat pentru că se bazează pe un simplu model RC al porţii, permite o legătură cu structura internă a circuitului, deci se poate alege topologia optimă de circuit. Prin aplicarea metodei se pot proiecta circuite care realizează abateri ale timpului de propagare cu cel mult 10% faţă de timpul minim. Este foarte important faptul că, prin utilizarea metodei efortului logic, se poate selecta în prima etapă cu certitudine care dintre variantele de circuit luate în considerare pentru analiză este mai rapidă; valoarea absolută a întârzierii pe circuit nu este chiar aşa de importantă în prima etapă, deoarece pentru o variantă de circuit selectat se pot face apoi simulări exacte de timp. Deşi metoda efortului logic conduce la proiectarea unui circuit CMOS rapid apare totuşi ca lacunară prin faptul că nu este corelată cu obţinerea pentru acel circuit şi a unui minim pentru aria consumată şi pentru puterea disipată.

1.5.6.1 Determinarea întârzierii pe o poartă logică

Primul pas în modelarea întârzierilor este concentrarea tuturor efectelor de întârziere, ale unui circuit de referinţă realizat într-o anumită tehnologie, sub forma unei unităţi etalon de întârziere. În fiecare tehnologie se alege ca circuit de referinţă, dintre toate porţile realizate în tehnologia respectivă, poarta cea mai simplă, adică poarta inversor. Iar ca **unitate etalon de întârziere**, notată cu τ, este întârzierea introdusă de poarta inversor când se consideră că nu are capacităţi interne parazite şi comandă doar o poarta inversor identică. Astfel se va exprima întârzierea absolută, d_{abs}, a unei alte porţi din aceeaşi tehnologie ca produsul dintre întârzierea etalon τ a inversorului şi o întârziere adimensională d (specifică fiecărei porţi)

$$d_{abs} = d \cdot \tau \tag{1.84}$$

Pentru o tehnologie cu caracteristica de $0,6\mu m$ şi $V_{DD} = 3,3V$ valoarea întârzierii etalon este $\tau = 50ps$. De fapt, caracteristica de viteză a procesului respectiv se exprimă printr-o singură valoare, τ.

Întârzierea pe o poartă logică cuprinde două componente, una constantă, notată cu p, datorată capacităţilor interne parazite şi alta notată cu $\boldsymbol{f}$, **referită ca efortul pe poartă/nivel** (stage effort) dependentă de sarcina comandă la ieşire şi de structurarea/topologia porţii. Rezultă că întârzierea adimensională a porţii, care indică de câte ori este mai mare decât a inversorului, poate fi exprimată prin relaţia:

$$d = f + p \tag{1.85}$$

La efortul pe poartă f contribuie **efortul electric**, **h**, ce caracterizează sarcina comandată de poartă şi **efortul logic**, **g**, ce înglobează particularitatea structurii porţii în raport cu structura porţii etalon (inversorul), acestea două sunt în relaţia:

$$f = h \cdot g \tag{1.86}$$

Efortul electric, h, reflectă cum mărimea sarcinii conectate la ieşire afectează întârzierea pe poarta logică şi cum dimensiunile porţilor tranzistoarelor determină capabilitatea de a produce curenţi pentru sarcina conectată la ieşire şi se exprimă prin raportul:

$$h = \frac{C_O}{C_{in}} \tag{1.87}$$

în care C_{in} este capacitatea prezentată de poartă la intrare, iar C_O este toată capacitatea comandată de poartă la ieşire. Evident, dacă se consideră la ieşire numai capacităţile prezentate de intrările porţilor comandate, efortul electric poate fi referit şi prin fan-out (numărul de intrări de porţi comandate la ieşire de către o poartă). Mai mult, este foarte uzuală încărcarea unei porţi, la ieşire, cu patru sarcini etalon, adică realizarea unei comenzi pentru patru inversoare, notată cu **FO4** (**fan-out 4**), vezi Exemplul 1.20. Dar, în general în valoarea lui C_O pe lângă capacităţile de intrare ale porţilor comandate intră şi capacităţile conexiunilor dintre ieşirea porţii de comandă şi intrările comandate (C_{cox} în relaţia 1.64), deci în cazul metodei efortului logic se va considera valoarea lui h obţinută prin raportul dat de relaţia 1.87 şi care numai în cazuri particulare coincide cu fan-out-ul (când nu se consideră capacităţile conexiunilor). Uneori atât C_{in} cât şi C_O se exprimă în unităţile în care se măsoară lăţimea canalului W, adică în μm, pentru că la aceeaşi lungime de canal L, a tuturor tranzistoarelor, capacităţile fiind proporţionale cu suprafaţa porţilor tranzistoarelor, $W \times L$, rezultă că sunt proporţionale cu W; dar în acest caz şi componenta de capacitate a conexiunilor, C_{cox}, care intră în C_O, trebuie convertită în μm. Pentru capacitatea de poartă a tranzistorului şi pentru capacitatea de conexiune valori uzuale sunt: respectiv $2f\ F/\mu m$ şi $0,2f\ F/\mu m$, valori care rămân cam aceleaşi pentru multe generaţii de proces atunci când se scalează identic dimensiunile dar simultan şi grosimea stratului de oxid (D_{ox}, Figura 1.32-a).

Din relaţiile 1.85, 1.86, 1.87 se obţine expresia întârzierii pe o poartă în unităţi τ:

$$d = gh + p \tag{1.88}$$

care reflectă faptul că efortul electric, h, şi efortul logic,g, contribuie la creşterea întârzierii în aceeaşi manieră. Pentru un inversor care comandă un inversor identic, $h = \frac{C_{in}}{C_{in}} = 1$, şi nu se consideră capacităţile interne parazite, $p = 0$, pentru a se obţine o întârziere adimensională egală cu unitatea $d = 1$, se consideră efortul logic $g = 1$ (rezultă din relaţia 1.88). Considerând $\mu_n = 2\mu_p$ (deşi la tehnologiile adânc submicronice μ_n/μ_p tinde spre $1,5$), pentru un inversor care să comande un curent egal pentru sarcină, atât în tranziţia $H-L$, cât şi în tranziţia $L-H$, este necesar raportul $W_n/W_p = 0,5$. Un inversor cu raportul $W_p = 2W_n$ prezintă la intrare o capacitate C_{inv} proporţională cu 3 unităţi de capacitate (de lăţime de canal W): o unitate de capacitate pentru poarta de canal nMOS şi două unităţi de capacitate pentru poarta de canal pMOS, Figura 1.60; evident s-a considerat capacitatea minimă, cea a porţii nMOS, ca unitate de capacitate.

Efortul logic exprimă informaţia despre topologia porţii – adică reţeaua de tranzistoare care realizează conectarea ieşirii porţii la tensiunea V_{DD} şi la tensiunea V_{SS} (masă) – necesară pentru determinarea întârzierii pe poartă. Se vor prezenta trei definiţii echivalente ale efortului logic, fiecare definiţie exprimând o altă perspectivă de abordare.

Definiţia 1.16 Efortul logic al unei porţi, g, este raportul dintre capacitatea de intrare a unei porţi, C_{in}, şi capacitatea de intrare a unei porţi inversor, C_{inv}, pentru care ambele porţi au aceleaşi valori ale curenţilor pe ieşire:

$$g = \frac{C_{in}}{C_{inv}} \tag{1.89}$$

$\diamond$

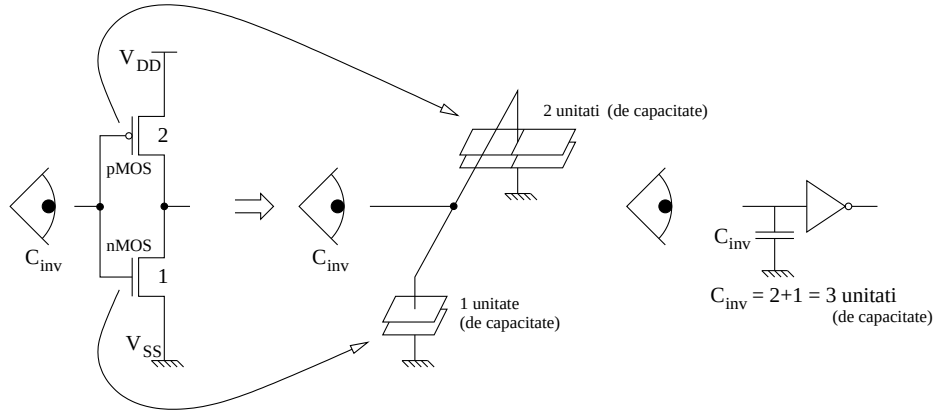


Figura 1.60 Exprimarea capacităţii de intrare a unui inversor C_{inv} în unităţi echivalente de capacitate (minimă).

Conform acestei definiţii, din Figura 1.61-a, 1.61-b şi 1.61-c se calculează pentru porţile inversor, NAND şi NOR valorile capacităţilor pe fiecare intrare, fiind egale respectiv cu $1 + 2 = 3$, $2 + 2 = 4$ şi $4 + 1 = 5$; rezultă deci valorile efortului logic 3/3 pentru inversor, 4/3, pentru NAND şi 5/3 pentru NOR. În *Tabelul 1.13* sunt prezentate valorile efortului logic pentru porţile foarte uzuale în funcţie de numărul de intrări şi de valorile raportului $\gamma = W_p/W_n$.

Definiţia 1.17 Efortul logic al unei porţi, g, este numărul care exprimă de câte ori este mai redus curentul pe ieşirea porţii decât curentul pe ieşirea unei porţi inversor când ambele porţi au aceeaşi capacitate de intrare. $\diamond$

Tabelul 1.13 Valorile efortului logic pentru porţile uzuale

Tipul de poartă	Efortul logic	Formula de calcul	Numărul de intrări			
			$n = 1$	$n = 2$	$n = 3$	$n = 4$
			$\gamma = 2$			
Inversor	1	$\frac{1+\gamma}{1+\gamma}$	1	-	-	-
NAND	Total	$\frac{n(n+\gamma)}{1+\gamma}$	-	8/3	5	8
	Pe intrare	$\frac{n+\gamma}{1+\gamma}$	-	4/3	5/3	2
NOR	Total	$\frac{n(n+n\gamma)}{1+\gamma}$	-	10/3	7	12
	Pe intrare	$\frac{1+n\gamma}{1+\gamma}$	-	5/3	7/3	3
Multiplexor	Total	$4n$	-	8	12	16
	Pe date, selector	$2, 2$	-	$2, 2$	$2, 2$	$2, 2$
XOR, XNOR	Total	$n^2 \cdot 2^{n-1}$	-	8	36	128
	Pe intrare pereche	$n \cdot 2^{n-1}$	-	4	12	32
Latch	Total	4	-			
	Pe date, clock	$2, 2$	-			

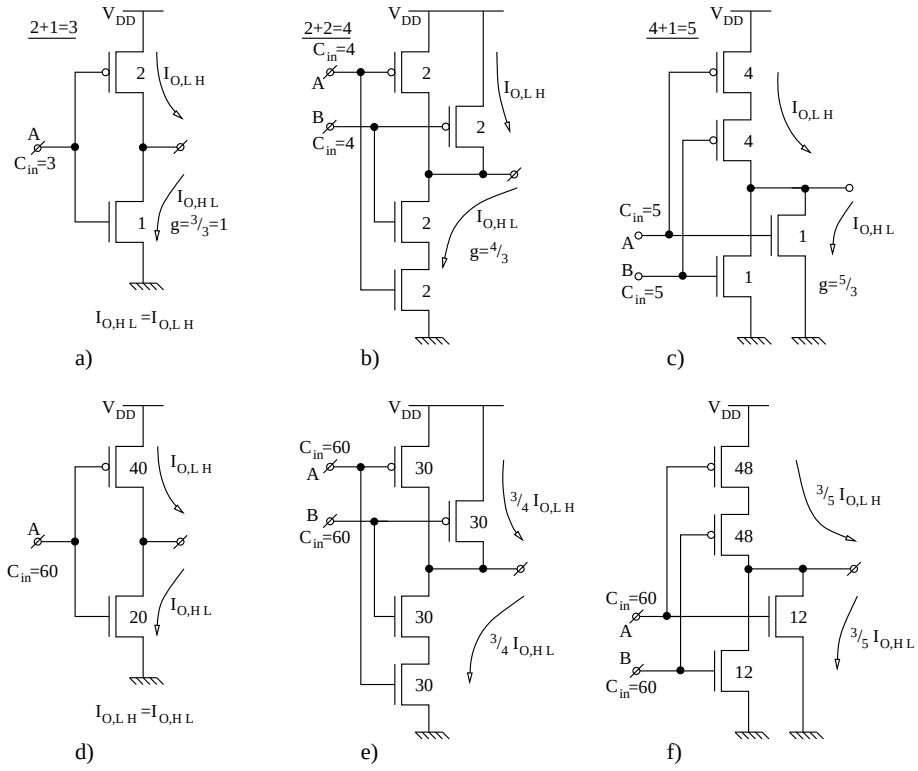


Figura 1.61 Efortul logic pentru porţile inversor, NAND şi NOR: a,b,c) când toate aceste porţi au aceeaşi capabilitate de comandă la ieşire; d,e,f) când toate aceste porţi au aceeaşi capacitate C_{in} pe intrare.

Orice topologie de poartă logică prezintă o mai mică capabilitate de a produce un curent de ieşire decât o poartă inversor când ambele au aceeaşi capacitate pe intrare. Reducerea curentului pe ieşirea porţii înseamnă un timp de propagare mai lung, deci efortul logic exprimă de câte ori este mai lentă poarta în efectuarea comenzii unei sarcini decât poarta inversor în efectuarea aceleiaşi comenzi.

Din Figura 1.61-e şi Figura 1.61-f rezultă de câte ori sunt mai reduşi curenţii la ieşirea porţilor NAND şi NOR decât curenţii la ieşirea porţii inversor, Figura 1.61-d, când toate trei porţile au aceeaşi capacitate la intrare, egală cu 60 de unităţi.

Definiţia 1.18 Efortul logic al unei porţi, g, este egal cu raportul dintre panta dreptei care reprezintă dependenţa întârzierii pe poartă în funcţie de fan-out şi panta dreptei care reprezintă dependenţa întârzierii porţii inversor în funcţie de fan-out. ◇

Această definiţie sugerează şi o metodă de măsurare a efortului logic din dependenţa grafică întârziere - fan-out obţinută experimental sau prin simulare, Figura 1.62.

Se poate defini un efort logic g_b şi pentru un grup de b intrări, analog relaţiei 1.89,

în felul următor:

$$g_b = \frac{C_b}{C_{inv}} = \frac{\sum_{i=1}^{b} C_i}{C_{inv}} \tag{1.90}$$

în care C_b este suma capacităţilor C_i, ale fiecărui semnal aplicat pe o intrare i din cele b intrări. De fapt, rezultă că g_b este suma eforturilor logice g_i pentru toate cele b semnale de intrare $g_b = \sum_{i=1}^{b} g_i$.

Din *Tabelul 1.13* rezultă că pentru o poartă cu cât este mai complexă funcţia logică realizată cu atât este mai mare efortul logic şi, mai mult, efortul logic al porţilor creşte cu numărul de intrări ale porţii, deci concluzia este că porţile mai complexe sau mai mari (ca număr de intrări) vor prezenta întârzieri mai mari. Această concluzie este un ghid în selectarea/alegerea unei structuri logice. Proiectarea care urmăreşte minimizarea numărului de niveluri logice va necesita mai multe intrări pentru fiecare poartă logică şi astfel efortul logic pe poartă devine mai mare. În schimb proiectarea care urmăreşte mai puţine intrări, şi astfel cu un efort logic mai mic pe nivel logic, va necesita mai multe niveluri logice. De fapt, acest aspect în proiectarea unui sistem este veşnicul compromis care trebuie făcut între adâncimea (întârzierea), $D(n)$, şi dimensiunea, $S(n)$, aspect ce va fi permanent în atenţie în capitolul 2 şi 3 al aceste lucrări.

Calculul întârzierii p, datorită capacităţilor interne parazite, nu este aşa de uşor de efectuat cum este pentru efortul logic. Principala contribuţie la această capacitate parazită o au capacităţile zonelor difuzate ale tranzistoarelor şi care sunt conectate la semnalul de ieşire. Pentru calculul unei astfel de capacităţi este luată în considerare atât suprafaţa zonei difuzate cât şi pereţii laterali ai zonei difuzate şi care depind de layout şi parametrii de proces. Considerând o capacitate de difuzie specifică pentru proces, C_d, atunci o aproximare grosieră pentru capacităţile zonelor difuzate de dren şi sursă poate fi exprimată prin produsul $W \cdot C_d$. Cu această aproximare se poate concepe un model pentru inversor în felul următor. Semnalul de ieşire din poarta inversor este conectat la două zone difuzate: una corespunde căii înspre masă prin canalul n pentru care se consideră $W = 1$ şi atunci această capacitate este C_d şi alta corespunde căii înspre V_{DD}, prin canalul p, pentru care capacitatea este $\gamma \cdot C_d$ deci suma lor este $(1 + \gamma)C_d$. Capacitatea de intrare în inversor este egală cu produsul dintre suprafaţa porţilor celor două canale $(1 + \gamma)$, pentru $W = 1$, şi capacitatea specifică de poartă C_{ox}, adică $(1 + \gamma)C_{ox}$. Întârzierea datorată capacităţilor interne parazite ale inversorului p_{inv} este egală cu raportul dintre capacităţile parazite şi capacitatea de intrare, $p_{inv} = \frac{(1+\gamma)C_d}{(1+\gamma)C_{ox}} = \frac{C_d}{C_{ox}}$. Se consideră pentru p_{inv} ca fiind o mărime adimensională egală cu unitatea $p_{inv} = 1$.

Întârzierea pentru o poartă se poate calcula din întârzierea inversorului p_{inv}, considerând că poarta are reguli de layout similare cu cele ale inversorului. Întârzierea pentru poartă, p, este mai mare decât a inversorului, p_{inv}, cu raportul dintre suma tuturor lăţimilor acelor zone difuzate W_d ale tranzistoarelor care sunt conectate la semnalul de ieşire şi lăţimea corespunzătoare a inversorului, cu condiţia că poarta logică este dimensionată să aibă acelaşi curent de ieşire ca şi inversorul:

$$p = \left(\frac{\sum W_d}{1 + \gamma} \right) p_{inv} \tag{1.91}$$

Aplicând această aproximare pentru poarta NAND cu n intrări care au în calea spre masă o singură zonă difuzată conectată la ieşire cu lăţimea nW_d şi în calea spre

tensiunea V_{DD} un număr de n zone difuzate de lăţimea γW_d, deci suprafaţa totală $n(1+\gamma)W_d$, rezultă $p = n \cdot p_{inv}$. La fel, se poate calcula pentru poarta NOR şi rezultă $p = n \cdot p_{inv}$. *Tabelul 1.14* prezintă, pentru câteva porţi uzuale, aceste calcule.

Tabelul 1.14 Estimarea întârzierilor p pentru unele porţi

Tipul de poartă	Formula	$p_{inv} = 1.0$		
		$n = 2$	$n = 3$	$n = 4$
NAND	$n \cdot p_{inv}$	2	3	4
NOR	$n \cdot p_{inv}$	2	3	4
Multiplexor	$2n \cdot p_{inv}$	4	6	8
XOR	$n \cdot 2^{n-1} \cdot p_{inv}$	4	12	

Estimarea componentei de întârziere p are serioase limitări când se consideră că această întârziere creşte liniar cu numărul de intrări ale porţii. Deoarece întârzierea datorată capacităţilor interne parazite este dependentă de layout cea mai bună cale pentru determinarea acesteia este prin simularea circuitului cu datele exacte de layout. Dar o determinare exactă pentru întârzierea p, datorită capacităţilor interne parazite, nu este necesară în etapa de dimensionare a tranzistoarelor deoarece această dimensionare se face în funcţie doar de efortul pe poarta $f = g \cdot h$ care este independent de p.

Valorile în unităţi de timp [ps] ale parametrilor τ şi p_{inv} pentru inversor şi, în extensie pentru fiecare poartă, pot fi determinate din reprezentarea grafică a întârzierii absolute a porţii inversor $d_{abs} = \tau(h + p_{inv})$, sau a unei porţi $d_{abs} = \tau(g \cdot h + p)$ în funcţie de efortul electric h, Figura 1.62. Această dreaptă poate fi trasată doar prin

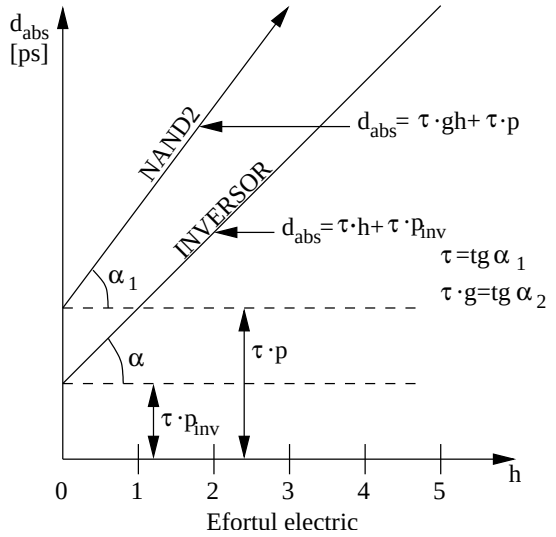


Figura 1.62 Utilizarea diagramei $d_{abs} = f(h)$ pentru calculul parametrilor τ şi p.

cunoaşterea a două puncte care pot fi determinate prin simulare, prin experiment sau din foaia de catalog a circuitului. Panta dreptei pentru inversor determină întârzierea unitară $\tau = tg\alpha_1$ iar tăietura pe ordonată este egală cu produsul $\tau \cdot p_{inv}$, deci rezultă p_{inv}. O metodă pentru determinarea întârzierii absolute d_{abs} pentru inversor este expusă în Exemplul 1.17. Prin măsurarea frecvenţei f a unui oscilator în inel compus din n inversoare, $d_{abs} = \tau(h+p)$ cu $h = 1$, $p_{inv} = 1$ rezultă $d_{abs} = 2\tau$ şi din egalitatea $2\tau = 1/2nf$ se obţine cu $n = 1$, $\tau = 1/4f$.

Exemplul 1.20 Să se estimeze întârzierea d pe un inversor care comandă patru inversoare identice, **inversor F**$O4$ (fan-out 4), Figura 1.63-a.

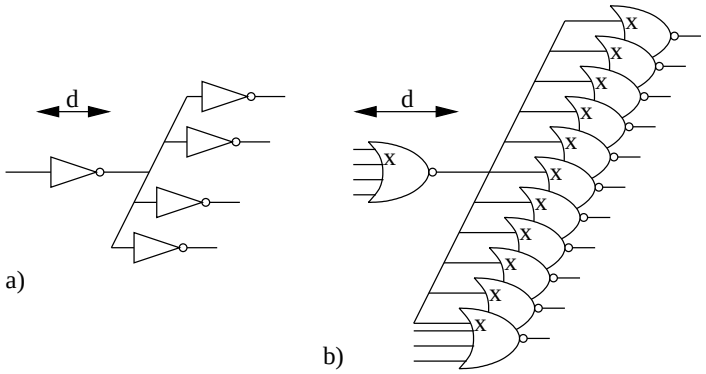


Figura 1.63 Calculul întârzierii pe poartă: a) Pentru inversor cu încărcare patru porţi inversor,FO4; b) Pentru NAND4 care comandă 10 porţi NAND4.

Soluţie. Deoarece toate inversoarele sunt identice $C_O = 4C_{in}$ deci $h = 4$. Pentru inversor efortul logic este unitar $g = 1$. Conform relaţiei 1.88 rezultă $d = gh + p = 1 \times 4 + p_{inv} = 4 + 1 = 5$. Adesea, întârzierea pe o poartă este exprimată în raport de întârzierea inversorului cu patru sarcini inversor, pentru că într-un proces este cunoscută întârzierea $FO4$, adică întârzierea de 5τ. Întârzierile pentru alte circuite vor fi multipli de 5τ (de FO4).

Exemplul 1.21 O poartă NOR4 (cu patru intrări) comandă alte 10 porţi NOR4 identice, Figura 1.63-b. Care este întârzierea d pe poarta de comandă?

Soluţie. Capacitatea C_{in} pe o intrare a porţii NOR4 este x iar capacitatea conectată pe ieşire $C_O = 10x$ deci efortul electric este $h = 10$. Efortul logic g pentru poarta cu patru intrări, obţinut din *Tabelul 1.13*, este $g = 9/3 = 3$, iar întârzierea p rezultă din *Tabelul 1.14*. Întârzierea pe poarta de comandă este $d = gh + p = 3 \times 10 + 4 \times 1 = 34$ unităţi de întârziere. Se observă că atunci când sarcina este mare întârzierea p datorată capacităţilor parazite interne este nesemnificativă în raport cu efortul pe poartă.

Exemplul 1.22 Să se structureze o poartă XOR şi să se calculeze efortul logic.

Soluţie. Exprimând $A \oplus B$ ca o funcţie negată, pentru implementare în CMOS,

$$A \oplus B = \overline{\overline{A \oplus B}} = \overline{\overline{A\overline{B} + \overline{A}B}} = \overline{AB + \overline{A}\,\overline{B}}$$

rezultă expresia logică $AB + \overline{A}\,\overline{B}$, pentru reţeaua nMOS, şi prin complementarea acesteia se obţine expresia logică $A\overline{B} + \overline{A}B$, pentru reţeaua pMOS, cu structurarea din Figura 1.64. Fiecare intrare este o pereche de semnale deoarece se generează atât variabila cât şi variabila negată.

Efortul total al porţii este $(8 + 8\gamma)/(1 + \gamma) = 8$ iar efortul logic pe o intrare (A, $\overline{A}$, B, $\overline{B}$) este de patru ori mai mic, adică 2. Evident că efortul logic pe o intrare pereche $A, \overline{A}$ sau $B, \overline{B}$ este de două ori mai mare, adică $2 \times 2 = 4$. De la structura XOR cu două variabile, prin extensie, se poate obţine o structură cu n variabile; în Figura 1.64-b, este prezentată o structură XOR pentru trei variabile $A \oplus B \oplus C$. O poartă XOR de n variabile va avea pentru reţeaua nMOS un număr de 2^{n-1} ramuri în paralel, fiecare ramură fiind compusă din n tranzistoare înseriate de lăţime n.Pentru reţeaua pMOS sunt, la fel, 2^{n-1} ramuri în paralel, fiecare ramură fiind compusă din n tranzistoare înseriate de lăţime $n\gamma$. Efortul logic total este $2^{n-1} \cdot n(n + n\gamma)/(1 + \gamma) = n^2 \cdot 2^{n-1}$, iar efortul logic pe o intrare va fi de $2n$ ori mai mic adică $n \cdot 2^{n-2}$, efortul logic pe o intrare pereche este de două ori mai mare $n \cdot 2^{n-1}$ (a se vedea *Tabelul 1.13*).

Cu această organizare simetrică a porţii XOR pentru $n \geq 3$ nu se obţine cel mai mic efort logic, de exemplu pentru $n = 3$ efortul logic este 36. Analizând semnalele pentru comanda tranzistoarelor se observă că unele dintre acestea pot fi comune pentru câte două ramuri, atât în reţeaua nMOS cât şi în reţeaua pMOS, astfel se obţine o organizare asimetrică prezentată în Figura 1.64-c.

La organizarea asimetrică calculând efortul logic total, pentru $n = 3$, rezultă valoarea de 24, valoare mai mică faţă de organizarea simetrică (32). Rezultă valoarea 6 pentru efortul logic al intrărilor pereche A şi C iar pentru intrarea pereche B valoarea 12 (ca şi la organizarea simetrică deoarece nici un tranzistor conectat la B sau $\overline{B}$ nu a fost eliminat).

Din organizarea porţii XOR cu două intrări rezultă imediat cea a porţii XNOR cu două intrări prin înlocuirea variabilei A cu $\overline{A}$ şi invers a variabilei $\overline{A}$ cu A.

1.5.6.2 Calculul întârzierii în reţelele de porţi logice

Într-o reţea de porţi logice metoda efortului logic permite evaluarea numărului optim de porţi pentru obţinerea întârzierii minime pe un anumit traseu precum şi repartizarea întârzierilor parţiale pe fiecare poartă din acel traseu. Noţiunile de efort logic şi efort electric, pentru un traseu de porţi, se obţin prin generalizarea acestor noţiuni definite pentru o singură poartă.

Efortul logic pentru un traseu, notat cu G, se obţine ca un produs al efortului logic g_i al tuturor porţilor de-a lungul acestui traseu:

$$G = \prod g_i \tag{1.92}$$

Efortul electric pentru un traseu, notat cu H, se obţine ca un raport între capacitatea totală care încarcă ieşirea ultimei porţi a traseului şi capacitatea de intrare de la prima poartă din traseu:

$$H = \frac{C_O}{C_{in}} \tag{1.93}$$

Dar în anumite puncte ale traseului considerat se pot ramnifica şi alte trasee (colaterale) deci în acele puncte curentul de ieşire al porţii respective se ramnifică între traseul considerat şi traseul/traseele colaterale; pentru aceste situaţii este necesar a se lua în considerare şi încărcările colaterale. Aceste încărcări colaterale sunt conţinute în noţiunea de efort de ramnificaţie la ieşirea porţii cu indexul i din traseu şi este

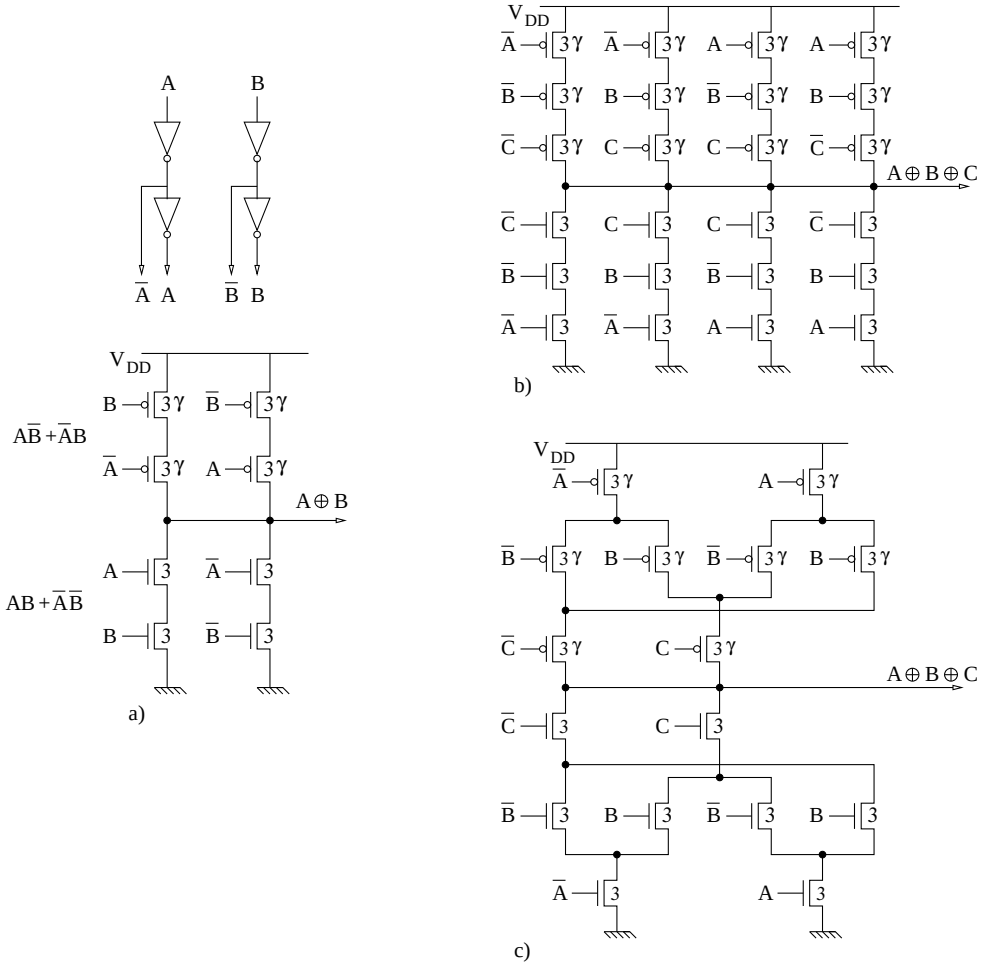


Figura 1.64 Structurarea porţii XOR în tehnologia CMOS: a) poarta XOR cu două intrări; b) structurare simetrică pentru poarta XOR cu trei intrări; c) o modalitate de structurare asimetrică pentru poarta XOR cu trei intrări.

notat cu b_i. **Efortul de ramnificaţie b_i** (în punctul de ramnificaţie) se defineşte ca raportul dintre suma capacităţii de intrare a porţii următoare din traseu, C_{it}, şi a capacităţilor de intrare de la porţile de pe traseul/traseele de ramnificaţie, C_{ir}, supra C_{it}.

$$b_i = \frac{C_{it} + C_{ir}}{C_{it}} \tag{1.94}$$

Într-un punct de neramnificaţie al traseului rezultă $b_i = 1$, deoarece $C_{ir} = 0$. Efortul de ramnificaţie pe întregul traseu, B, se obţine ca un produs al tuturor eforturilor de ramnificaţie b_i de-a lungul traseului:

$$B = \prod b_i \tag{1.95}$$

Similar cu relaţia 1.86 se poate defini un efort al traseului, notat cu F, ca produsul dintre efortul electric, H, efortul logic, G, şi efortul de ramnificaţie, B, de-a lungul traseului:

$$F = GBH \tag{1.96}$$

Se observă că F depinde numai de topologia (G) şi de încărcarea traseului (H) şi nu de dimensiunea tranzistoarelor din porţile de pe acest traseu. Mai mult, efortul traseului nu se schimbă dacă se introduc sau se scot inversoare deoarece acestea au efortul logic egal cu 1.

Pentru traseu, produsul dintre efortul electric H şi efortul de ramnificaţie B este egal cu produsul efortului electric h_i al tuturor porţilor de pe traseu:

$$BH = \frac{C_O}{C_{in}} \prod b_i = \prod h_i \tag{1.97}$$

această relaţie este importantă deoarece pentru un traseu se cunoaşte C_{in}, C_O şi eforturile de ramnificaţie b_i deci proiectantul va trebui să dimensioneze efortul electric pentru fiecare poartă h_i astfel încă să realizeze produsul BH.

Întârzierea pe traseu D este egală cu suma întârzierilor porţilor din traseu şi, la fel ca în relaţia 1.88, are două componente: una, întârzierea de efort a traseului D_F şi cealaltă întârzierea datorată capacităţilor interne parazite ale porţilor din traseu, P, deci se pot scrie relaţiile:

$$D = \sum d_i = D_F + P \tag{1.98}$$

$$D_F = \sum g_i h_i \tag{1.99}$$

$$P = \sum p_i \tag{1.100}$$

Se demonstrează că intârzierea pe traseu are valoarea minimă când fiecare poartă din cele N ale traseului are acelaşi efort, notat cu $\hat{f}$, a cărui valoare se obţine cu relaţia:

$$\hat{f} = g_i h_i = \sqrt[N]{F} \tag{1.101}$$

Din relaţiile 1.101 şi 1.98 se deduce principalul rezultat al metodei efortului logic, adică relaţia care exprimă întârzierea minimă,notată cu $\hat{D}$, ce se poate obţine pentru un traseu:

$$\hat{D} = N \cdot F^{\frac{1}{N}} + P \tag{1.102}$$

Pentru un traseu a cărui topologie este dată se poate calcula efortul F şi repartizând un efort egal pe fiecare poartă, din relaţia 1.101, rezultă cât trebuie să fie efortul electric $\hat{h}_i$ pe fiecare poartă din traseu

$$\hat{h}_i = \frac{F^{\frac{1}{N}}}{g_i} \tag{1.103}$$

iar, apoi, cu valoarea lui $\hat{h}_i$ cunoscută, se pot determina dimensiunile tranzistoarelor din fiecare poartă logică din traseu. De fapt, calculul începe cu dimensionarea tranzistoarelor din ultima poartă a traseului, pentru care se cunoaşte: încărcarea la ieşire C_O, efortul logic şi valoarea $\hat{f}$, deci rezultă capacitatea de intrare a ultimei porţi (care este capacitatea de ieşire a penultimei porţi atunci când efortul de ramnificaţie în acest punct este 1). Apoi, după dimensionarea tranzistoarelor din ultima poartă a traseului se dimensionează, succesiv, şi celelalte porţi în sensul invers de parcurgere a porţilor din traseu, adică de la penultima poartă până la poarta de intrare. Relaţia de dimensionare pentru poarta cu indicele i din traseu este:

$$C_{in_i} = \frac{g_i C_{O_i}}{\hat{f}} \tag{1.104}$$

din care rezultă capacitatea C_{in_i} de intrare în poarta i, care apoi se distribuie potrivit la toate tranzistoarele de pe acea intrare a porţii. C_{O_i} a rezultat deja de la dimensionarea (anterioară a) porţii următoare din traseu, $(i+1)$.

Exemplul 1.23 Un traseu $A - B$ ca în Figura 1.65-a este compus prin înserierea a trei porţi NAND2; capacitatea de intrare a primei porţi este C iar încărcarea ultimei porţi este tot C. Care este întârzierea minimă pe traseu şi care sunt dimensiunile tranzistoarelor acestor porţi?

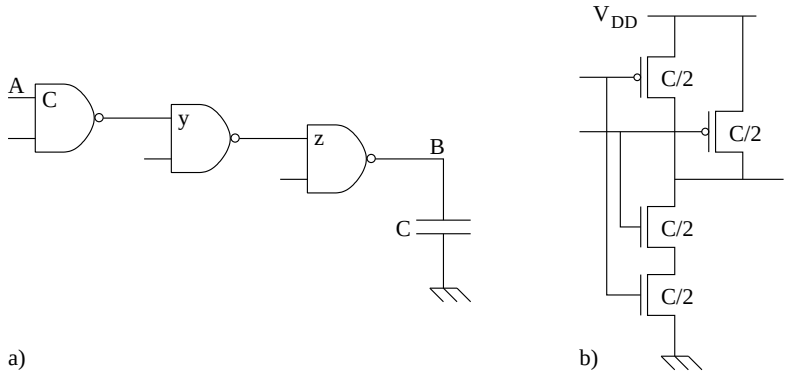


Figura 1.65 Explicativă pentru exemplul 1.23

Soluţie. Pentru a determina efortul traseului B trebuie calculate efortul electric H, de ramnificaţie B şi logic G, de-a lungul traseului. Efortul logic este produsul efortului logic al celor trei porţi NAND2, $G = g_0 g_1 g_2 = (4/3)(4/3)(4/3) = (4/3)^3 = 2,37$. Efortul de ramnificaţie B are valoarea 1 deoarece nu există nici o ramnificaţie pe traseu. Efortul electric al traseului este $H = C/C = 1$, deci $F = GBH = 2,37$. Când efortul pe fiecare

poartă are aceeaşi valoare egală cu $\hat{f} = \sqrt[3]{F} = 4/3$ întârzierea minimă care se poate obţine pe traseu se calculează cu relaţia 1.102, $\hat{D} = 3 \cdot \sqrt[3]{2,37} + 3(2p_{inv}) = 10$ unităţi de întârziere. Pornind dimensionarea tranzistoarelor de la ultima poartă care are capacitatea pe ieşire C şi aplicând relaţia 1.104 se obţine capacitatea de intrare pentru a treia poartă NAND2, $z = (c \cdot 4/3)/(4/3) = C$. Similar rezultă capacitatea de intrare pentru a doua poartă NAND2, $y = (z \cdot 4/3)/(4/3) = z = C$. Rezultă că toate porţile au aceleaşi dimensiuni pentru tranzistoare, adică porţile sunt identice, ceea ce nu este surprinzător deoarece fiecare poartă are aceeaşi sarcină şi acelaşi efort logic, iar pentru obţinerea timpului minim toate trebuie să aibă acelaşi efort. Făcând dimensionarea pentru $\mu_n/\mu_p = 2$ rezultă o poartă NAND2 cu dimensiunile pentru porţile tranzistoarelor specificate în Figura 1.65-b.

Exemplul 1.24 Pentru traseul $A - B$, Figura 1.65-a de la Exemplul 1.23, să se determine întârzierea minimă şi dimensiunile tranzistoarelor când sarcina la ieşire este $8C$.

Soluţie. $G = (4/3)^3$, $H = 8C/C = 8$, $B = 1$ deci $F = GBH = (4/3)^3 \cdot 8 = 18,96$ iar întârzierea minimă care se poate obţine este $\hat{D} = 3(18,96)^{\frac{1}{3}} + 3(2p_{inv}) = 14$ unităţi de întârziere. Se observă că deşi efortul electric este de 8 ori mai mare decât în Exemplul 1.23 întârzierea minimă a crescut numai cu 40 %. Pentru efort egal pe fiecare poartă $\hat{f} = \sqrt[3]{18,96} = 8/3$ se determină dimensiunile tranzistoarelor pornind cu ultima poartă $z = (8C \cdot 4/3)/(8/3) = 4C$, $y = (z \cdot 4/3)/(8/3) = 2C$, iar ca o verificare se calculează şi capacitatea de intrare pentru prima poartă $(y \cdot 4/3)/(8/3) = y/2 = C$. Fiecare poartă are capacitatea de intrare dublă faţa de capacitatea porţii anterioare, deci dimensiunile tranzistoarelor se dublează pe fiecare nivel în sensul de parcurgere a traseului de la A la B, tranzistoare cu lăţime de canal mai mare în etaje succesive au capabilitate mai mare de a comanda curenţi în sarcină.

Exemplul 1.25 Să se dimensioneze tranzistoarele porţilor de pe traseul $A - B$ din Figura 1.66-a pentru a obţine timpul minim de întârziere când efortul electric al traseului este $H = 4,5$.

Soluţie. Efortul de ramnificaţie la ieşirea primei porţi este $b_1 = (y + y)/y = 2$ iar la ieşirea celei de-a doua porţi este $b_2 = (2+2+2)/2 = 3$ deci $B = 2 \times 3$, $G = (4/3)^3$ iar $H = 4,5$, rezultă $F = GBH = 64$. Întârzierea minimă pe traseu se obţine $\hat{D} = 3 \times 64^{\frac{1}{3}} + 3(p_{inv}) = 18$ unităţi de întârziere. Se calculează un efort egal pe fiecare poartă $\hat{f} = \sqrt[3]{64} = 4$. Rezultă $z = (4,5 \times 4/3)/4 = 1,5$, $y = (b_2 z \times 4/3)/4 = z = 1,5C$ iar pentru prima poartă se poate face o verificare $(b_1 y \times 4/3)/4 = 2/3y = C$, deci se pot dimensiona tranzistoarele pe intrare ale porţilor NAND2.

Exemplul 1.26 Să se dimensioneze circuitul din Figura 1.66-b pentru a realiza o întârziere minimă. Se consideră că sarcina de ieşire C_O este capacitatea unei porţi logice cu lăţimea (de poartă) de $20\mu m$ iar capacitatea de intrare corespunde unei porţi de lăţime $10\mu m$.

Soluţie. Presupunând aceeaşi lungime pentru toate tranzistoarele rezultă că valoarea capacităţii lor este proporţională cu lăţimea porţilor lor deci se poate exprima valoarea capacităţii şi prin μm. Se calculează pentru traseu: efortul logic $G = 1 \times (5/3) \times (4/3) \times 1 = 20/9$, efortul electric $H = 20/10 = 2$, efortul de ramnificaţie $B = 1 \times 1 \times 1 = 1$ rezultă $F = GBH = 40/9$, deci efortul egal pe fiecare poartă, pentru întârziere minimă, este $\hat{f} =$

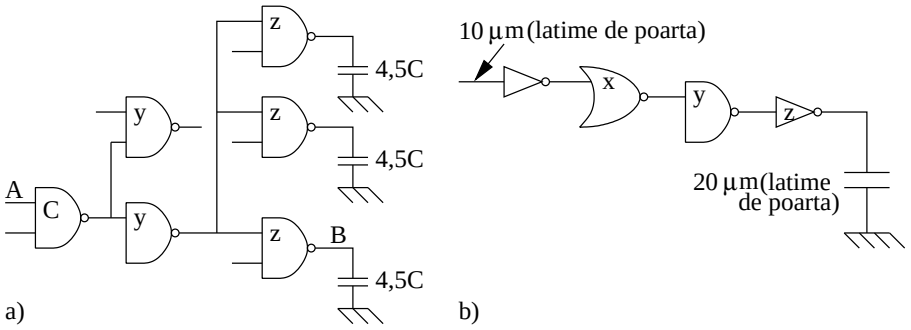


Figura 1.66 Explicativă pentru: a) Exemplul 1.25; b) Exemplul 1.26.

$\sqrt[4]{40/9} = 1,45$. Parcurgând traseul în sens invers, pornind cu ultima poartă, rezultă lăţimea totală a porţilor tranzistoarelor de intrare de pe nivelul respectiv: $z = (20 \times 1)/1,45 = 14\mu m$, $y = (14 \times 4/3)/1,45 = 13\mu m$ şi $x = (13 \times 5/3)/1,45 = 15\mu m$; apoi aceste valori de capacitate se distribuie potrivit pentru fiecare poartă de tranzistor din nivelul respectiv. Se observă, că rezultă lăţimi mai mari pentru porţile tranzistoarelor din inversoare decât pentru porţile tranzistoarelor din celelalte porţi logice ceea ce este normal deoarece inversoarele au o mai bună capabilitate de a comanda sarcini.

Se observă că în dimensionarea tranzistoarelor pentru obţinerea întârzierii minime pe traseu, relaţia 1.102, am neglijat întârzierea P deoarece aceasta este fixă, iar ajustarea dimensiunii porţii nu modifică întârzierea parazitică. De fapt, se poate neglija totdeauna întârzierea P când nu se doreşte o precizie ridicată în determinarea întârzierii pe traseu sau când se compară două trase care conţin porţi logice diferite ori conţin un număr diferit de porţi logice.

1.5.6.3 Alegerea numărului optim de niveluri pe un traseu

Când numărul N de niveluri în traseu este dat cu relaţia 1.101 se calculează valoarea efortului ,$\hat{f} = F^{\frac{1}{N}}$,egal pe fiecare nivel, care va determina întârzierea minimă pe traseu. Dar pentru un traseu, la care se cunoaşte efortul F dar nu şi numărul de niveluri, s-ar putea ca numărul dat N să nu fie egal cu cel optim, număr notat cu $\hat{N}$, care să genereze cea mai mică întârziere. Când numărul de etaje pentru un efort dat pentru traseu este cel optim $\hat{N}$, atunci cel mai bun efort egal pe etaj, notat cu ρ, se calculează cu relaţia $\rho = F^{\frac{1}{\hat{N}}}$ şi care este soluţia următoarei ecuaţii:

$$p_{inv} - \rho(1 - ln\rho) = 0 \tag{1.105}$$

Tabelul 1.15 prezintă numărul optim de niveluri $\hat{N}$ într-un traseu pentru obţinerea celei mai mici întârzieri pe traseu, când $p_{inv} = 1$.

Pentru un traseu cu N etaje se calculează efortul F, iar din inspecţia *Tabelului 1.15* rezultă că întârzierea cea mai mică necesită $\hat{N}$ etaje, deci este necesar a se adăuga sau a se scădea etaje din numărul de etaje N până se ajunge la $\hat{N}$ etaje în traseu. Informaţia tabelului este corectă numai când se măreşte sau se scade numărul de niveluri prin introducerea sau scoaterea de etaje inversor, deoarece se consideră că numai aceste etaje au întârzierea parazitică $p = 1$. Evident, pentru

Tabelul 1.15 Numărul optim de niveluri $\hat{N}$ în funcţie de efortul pe traseu F, pentru $p_{in} = 1$

Efortul de traseu F	Numărul optim de etaje $\hat{N}$	Întârzierea minimă D	Efortul pe etaj f
0		1	
	1		$0 \sim 5,8$
5,83		6,8	
	2		$2.4 \sim 4,7$
22,3		11,4	
	3		$2,8 \sim 4,4$
82,2		16,0	
	4		$3,0 \sim 4,2$
300		20,7	
	5		$3,1 \sim 4,1$
1090		25,3	
	6		$3,2 \sim 4,0$
3920		29,8	
	7		$3,3 \sim 3,9$
14200		34,4	
	8		$3,3 \sim 3,9$
51000		39,0	
	9		$3,3 \sim 3,9$
184000		43,6	
	10		$3,4 \sim 3,8$
661000		48,2	
	11		$3,4 \sim 3,8$
2380000		52,8	
	12		$3,4 \sim 3,8$
8560000		57,4	

a nu se schimba funcţia logică calculată pe traseu trebuie introduse sau scoase un număr par de inversoare. Unul sau două niveluri diferenţă faţă de valoarea optimă $\hat{N}$ modifică nesemnificativ întârzierea pe traseu, dacă acest traseu este destul de lung, numai la trasee scurte introducerea sau scoaterea a unui etaj sau două poate provoca abateri semnificative faţă de întârzierea cea mai mică.

Exemplul 1.27 Se consideră trei variante de trasee inversoare compuse din înserierea de respectiv 1,3 şi 5 inversoare, fiecare traseu comandă la ieşire o sarcină capacitivă egală cu de 25 de ori capacitatea de intrare. Care dintre aceste variante este optimă şi cât este întârzierea cea mai mică?

Soluţie. Pentru fiecare din trasee rezultă valori egale de $G = 1$, $B = 1$, şi $H = 25$. Întârzierea pe fiecare traseu este dată de ecuaţia 1.102, $\hat{D} = N \cdot (25)^{\frac{1}{N}} + N \cdot p_{inv}$, cu $N = 1$,3 şi 5. Pentru $N = 1$ rezultă $\hat{D} = 25$ unităţi de întârziere; pentru $N = 3$ rezultă $\hat{D} = 11,8$ unităţi de întârziere iar pentru $N = 5$ rezultă $\hat{D} = 14,5$ unităţi de întârziere. Varianta cu trei inversoare înseriate conţine numărul optim de inversoare în consecinţă realizează întârzierea cea mai mică $D = 11,8$ unităţi de întârziere, fiecare etaj suportă un efort egal cu $\sqrt[3]{25} = 2,9$, deci fiecare inversor va fi dimensional de $2,9$ ori mai mare decât predecesorul inversor. Se poate constata şi din *Tabelul 1.15* că pentru $F = 25$, $22,3 < 25 < 82,2$ numărul optim de etaje este $\hat{N} = 3$, adică exact cel calculat prin acest exemplu.

Exemplul 1.28 Un şir de inversoare în tehnologia $0,6\mu$ $(3,3V, \tau = 50ps)$ comandă un semnal conectat la un terminal de ieşire al circuitului integrat (pad). Capacitatea terminalului de ieşire este de $40pF$ care este echivalentă cu o capacitate de poartă de $20000\mu m$. Presupunând capacitatea de intrare în şirul de inversoare de $7,2\mu m$ să se dimensioneze şirul de inversoare cel mai rapid.

Soluţie. Se calculează $B = 1$, $G = 1$ şi $H = 20000/7,2 = 2777$. Din *Tabelul 1.15* rezultă că numărul optim de inversoare este $\hat{N} = 6$, $1090 < 2777 < 3920$. Efortul cel mai bun pe etaj este $\hat{f} = (2777)^{\frac{1}{6}} = 3,75$ deci fiecare inversor are capacitatea de intrare de $3,75$ ori mai mare decât a inversorului precedent, iar întârzierea este $\hat{D} = 6 \times 3,75 + 6 \times p_{inv} = 28,5$ unităţi de intârziere, adică $28,5 \times 50ps = 1,43ns$.

În acest exemplu s-a găsit că cel mai bun raport de multiplicare a dimensiunilor de la un nivel la următorul nivel este de $3,75$. Unele referinţe indică pentru acest raport de multiplicare valoarea de $2,71$, adică constanta e, care este corect numai când întârzierea parazitică p este nulă. Pe măsură ce p creşte raportul de multiplicare creşte peste valoarea lui e iar numărul de niveluri scade . Dacă efortul pe nivel variază în intervalul de 2 până 8 abaterea întârzierii realizată faţă de cea mai mică întârziere este 35%, iar dacă efortul pe nivel variază în intervalul de $2,4$ până la 6 abaterea este doar de 15%. Pentru $p_{inv} = 1$ cel mai bun efort pe nivel este de $3,59$. A intrat în uz ca cel mai bun efort pe nivel să fie considerat 4, pentru că este un număr "rotund" şi este uşor de calculat mental numărul de niveluri, această alegere de efort 4 pe nivel duce la o abatere de 1% faţă de întârzierea cea mai mică pentru $p_{inv} = 1$. Considerând efortul pe nivel 4 numărul $\hat{N}$ de niveluri rezultă prin relaţia $log_4 F$ iar întârzierea $FO4$, pe un nivel cu încărcarea 4, este cunoscută ca fiind egală 5τ (vezi Exemplul 1.20), deci întârzierea se calculează simplu cu relaţia $\hat{D} = 5log_4 F$. Cuadruplând efortul pe traseu atrage după sine o întârziere egală cu cea a unui inversor care comandă alte patru inversoare identice.

Un caz particular de structură de circuit, dar foarte des utilizat, este aşa numita "furcă" ce se compune din două ramuri de inversoare, pe o ramură numărul inversoarelor înseriate fiind impar iar pe cealaltă ramură numărul inversoarelor înseriate fiind par, astfel că la ieşiri se generează valoarea negată respectiv nenegată a semnalului comun aplicat la intrare, Figura 1.67. Pentru circuitul furcă se cunosc cele două capacităti de ieşire C_a, C_b, în general egale $C_a = C_b$, de la cele două ramuri, $C_O = C_a + C_b$. De asemenea, se cunoaşte capacitatea totală de intrare în furcă, C_{in}, care este sarcina suportată de semnal la intrarea în furcă, pe care va trebui să o repartizăm pe cele două intrări C_{in_a}, C_{in_b}, $C_{in} = C_{in_a} + C_{in_b}$, astfel încât întârzierile pe cele două ramuri să fie egale. Se poate defini un efort electric total pentru circuitul furcă $H = C_O/C_{in}$ şi, respectiv, câte un efort electric individual pentru fiecare ramură $H_a = C_a/C_{in_a}$, $H_b = C_b/C_{in_b}$, aceste eforturi electrice individuale, în general, nu sunt egale chiar dacă încărcările la ieşire sunt egale.

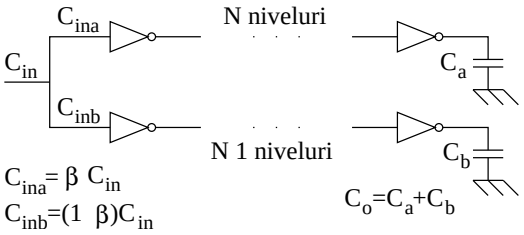


Figura 1.67 Structura generală a circuitului *furcă* :

Pentru proiectarea circuitului furcă se porneşte cu determinarea coeficientului de repartizare β a capacităţii de intrare, C_{in}, între cele două ramuri, $C_{in_a} = \beta \cdot C_{in_b}$, coeficient care rezultă din egalitatea întârzierilor, ecuaţia 1.102, pe cele două ramuri. Ramura care suportă un efort electric mai mare, uzual cea cu amplificatoare mai multe, are o capacitate de intrare mai mică $\beta < 0,5$, deci i se repartizează o parte mai mică din curentul semnalului de intrare decât celeilalte ramuri. Evident, că oricare ramură poate fi dimensionată să devină mai rapidă, prin reducerea efortului electric al său ceea ce implică mărirea dimensiunilor tranzistoarelor de la primul etaj, adică mărindu-i curentul repartizat în detrimentul celeilalte ramuri care va deveni mai lentă.

Un circuit furcă particular este referit prin numărul de inversoare conţinut în fiecare ramură, în general diferenţa este de un singur inversor, deci pot fi circuite: $2-1$, $3-2$, $4-3$, $5-4$, ... În funcţie de valoarea efortului electric total $H = C_O/C_{in}$ din *Tabelul 1.16* se obţine recomandarea ce structură de circuit furcă este indicat să se utilizeze (pentru $p_{inv} = 1$).

Exemplul 1.29 Să se proiecteze un circuit furcă $2-1$ cu capacitatea de intrare $C_{in} = 10$ iar capacitatea de ieşire, pe fiecare ramură, este $C_a = C_b = 100$. Care este întârzierea introdusă de circuit?

Soluţie. Coeficientul β de repartizare a capacităţii de intrare C_{in} între ramura cu două inversoare, $C_{in_a} = \beta \cdot C_{in}$ şi ramura cu un singur inversor $C_{in_b} = (1-\beta)C_{in}$, rezultă din egalizarea întârzierilor, ecuaţia 1.102, pe cele două ramuri.

$$2\left(\frac{100}{10\beta}\right)^{\frac{1}{2}} + 2p_{inv} = \left(\frac{100}{10(1-\beta)}\right) + p_{inv}$$

Prin rezolvare numerică rezultă $\beta = 0,258$, deci $C_{in_a} = 10\beta = 2,6$ şi $C_{in_b} = 10(1-\beta) = 7,4$. Al doilea inversor din ramura cu două inversoare are capacitatea de intrare, conform ecuaţiei 1.104, egală cu $C_{a2} = 2,6 \times (100/2,6)^{\frac{1}{2}} = 16,1$. Întârzierea în ramură cu un singur inversor, conform ecuaţiei 1.98, este $C_b/C_{in_b} + p_{inv} = 100/7,4 + 1 = 14,5$. Iar întârzierea în ramura cu două inversoare este $C_{a2}/C_{in_a} + C_a/C_{a2} + 2p_{inv} = 16,1/2,6 + 100/16,1 + 2 = 14,5$.

Tabelul 1.16 Circuitul furcă recomandat în funcţie de efortul electric total ($p_{inv} = 1$)

H		Circuitul furcă
De la	Până la	
	9,68	2-1
9,68	38,7	3-2
38,7	146	4-3
146	538	5-4
538	1970	6-5
1970	7150	7-6

Exemplul 1.30 Să se proiecteze un circuit furcă pentru comanda de validare/acces a 64 buffere tristate la o magistrală, Figura 1.68-b. Capacitatea de intrare la circuitul furcă este de 12 unităţi, iar un buffer tristate are dimensiuni de şase ori mai mari decât cele ale structurii de referinţă de circuit poartă tristate. Structura de referinţă de circuit

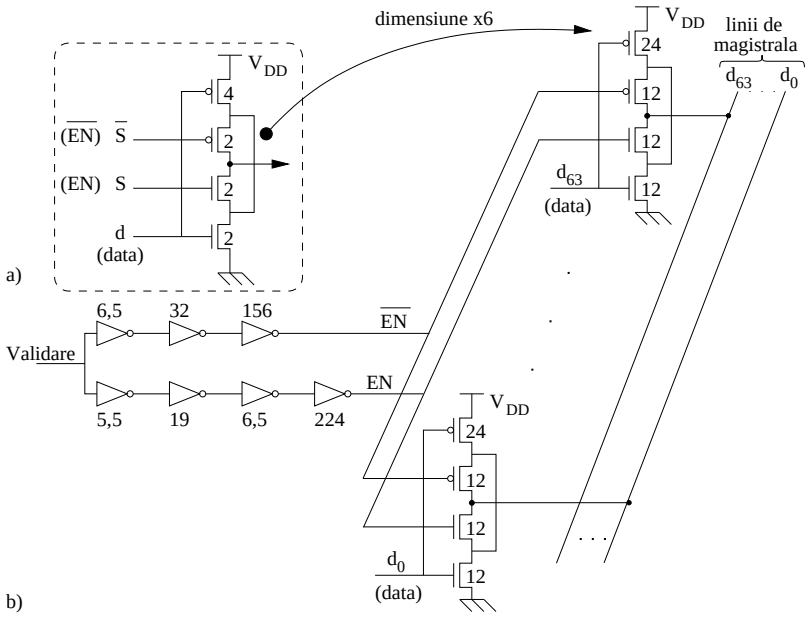


Figura 1.68 Explicativă pentru exemplul 1.30

poartă tristate este prezentată în Figura 1.68-a, acest circuit are aceeaşi funcţionare ca şi circuitul din Figura 1.52-c, care este format dintr-un inversor CMOS static având pe ieşire o poartă de transmisie CMOS (dar de data aceasta, tranzistoarele complementare ale porţii de transmisie sunt înseriate/incluse în coloana porţii CMOS statică şi comandate cu semnalele complementare s şi $\overline{s}$).

Soluţie. Sarcina pe ieşire pentru fiecare din semnalele de validare/ENable, EN, $\overline{EN}$ este $64 \times 6 \times 2 = 768$ unităţi de capacitate. Efortul electric pentru întregul circuit furcă este $H = (768 + 768)/12 = 128$, iar din *Tabelul 1.16* rezultă că structura de $4-3$ este cea optimă. Coeficientul de repartizare β a capacităţii de intrare pe cele două ramuri rezultă din relaţia

$$4\left(\frac{768}{12\beta}\right)^{\frac{1}{4}} + 4p_{inv} = 3\left(\frac{768}{12(1-\beta)}\right)^{\frac{1}{3}} + 3p_{inv}$$

Prin rezolvare numerică se obţine $\beta = 0,46$. Capacitatea de intrare pe ramura cu patru inversoare este $12 \times 0,46 = 5,5$ iar pe ramura cu trei inversoare este $12 - 5,5 = 6,5$, în consecinţa efortul electric pe ramură este respectiv egal cu $768/5,5 = 140$, $768/6,5 = 118$, la fel şi efortul pe nivel este $\sqrt[4]{140} = 3,44$, $\sqrt[3]{118} = 4,90$. Pornind cu dimensionarea, de la capacitatea de sarcină de ieşire spre capacitatea de intrare, rezultă capacităţile de intrare în fiecare nivel egale cu 224, 65,19 şi 5,5 pentru ramura cu 4 inversoare şi 156, 32 şi 6,5 pentru ramura cu 3 inversoare. Întârzierea pe fiecare ramură este $4(140)^{\frac{1}{4}} + 4p_{inv} = 4 \times 3,44 + 4 = 17,7$ şi $3 \times (118)^{\frac{1}{3}} + 3p_{inv} = 3 \times 4,90 + 3 = 17,7$. Se observă că efortul electric total al circuitului $H = 128$ $((118 + 140)/2 = 128)$ este balansat inegal pe cele două ramuri prin îmbunătăţirea întârzierii pe ramura mai lentă în detrimentul celei mai rapide astfel încât fiecare să ajungă la întârzierea de $17,7$.

Modul de analiză pentru circuitul furcă se poate extinde şi pentru circuite mai complexe cu mai multe ramuri, unde fiecare ramură conţine un număr diferit de niveluri, fiecare ramură realizează o funcţie logică diferită şi unde fiecare ramură comandă sarcină diferită. Totuşi această extindere la circuite mai complexe necesită unele artificii ceea ce arată unele deficienţe şi limitări ale metodei.

Metoda efortului logic este o procedură de proiectare pentru obţinerea celui mai mic timp de întârziere pentru un traseu dintr-o reţea. Această metodă combină într-un singur calcul atât capabilitatea de a comanda sarcini electrice mari cât şi realizarea unei funcţii logice. Expresiile de calcul în cadrul metodei sunt concentrate în *Tabelul 1.17* iar procedura de calcul parcurge următoarele etape:

1. Se calculează efortul $F = GBH$ pentru traseul din reţeaua de analizat. Efortul logic pe traseu, G, este produsul efortului logic al tuturor porţilor logice de pe traseu, efortul logic al porţilor este dat în *Tabelul 1.13*; efortul de ramnificaţie pe traseu, B, este produsul efortului de ramnificaţie al tuturor porţilor de pe traseu; efortul electric pe traseu, H, este raportul dintre capacitatea de sarcină totală a ultimului nivel şi a capacităţii de intrare la primul nivel.

2. Se estimează numărul optim de niveluri $\hat{N}$, pentru efortul F calculat, care produce cel mai mic timp de întârziere; pentru această estimare se utilizează fie *Tabelul 1.15*, fie relaţia $\hat{N} \approx log_4 F$.

3. Se estimează întârzierea minimă, $\hat{D} = \hat{N}F^{\frac{1}{\hat{N}}} + \sum p_i$, folosind pentru întârzierea datorită capacităţilor interne parazite datele din *Tabelul 1.14*. Procedura se

opreşte aici dacă se urmăreşte doar compararea întârzierilor diferitelor structuri. Pentru o proiectare se parcurg şi punctele următoare.

4. Se adaugă sau se elimină niveluri până când numărul de niveluri N atinge valoarea $\hat{N}$.

5. Se calculează efortul suportat pe fiecare nivel: $\hat{f} = F^{\frac{1}{N}}$.

6. Se dimensionează tranzistoarele din fiecare etaj, succesiv, pornind de la ultimul nivel folosind relaţia $C_{in} = \left(g_i/\hat{f}\right) C_O$ până la primul nivel; valoarea calculată C_{in} a unui etaj i devine capacitatea de ieşire a etajului $i-1$ (eventual modificată prin efortul de ramnificaţie); indicele etajului în traseu creşte începând cu etajul de intrare înspre etajul de ieşire.

Tabelul 1.17 Relaţiile utilizate de metoda efortului logic

Denumire	Expresia pe nivel	Expresia pe traseu
Efortul logic	g (Tabelul 1.13)	$G = \prod g_i$
Efortul electric	$h = \frac{C_O}{C_{in}}$	$H = \frac{C_{O_{traseu}}}{C_{in_{traseu}}}$
Efortul de ramnificaţie	-	$B = \prod b_i$
Efortul	$f = gh$	$F = GBH = \prod f_i$
Întârzierea de efort	f	$D_F = \sum f_i$ minimizat când $f = F^{\frac{1}{N}}$
Numărul de niveluri	1	N (Tabelul 1.15)
Întârzierea parazitică	p (Tabelul 1.14)	$P = \sum p_i$
Întârzierea totală	$d = f + p$	$D = D_F + P$

Metoda efortului logic poate indica rapid proiectantului care dintre structurile analizate trebuie a fi aleasă şi pentru cea aleasă se poate obţine o proiectare aproape optimă, iar apoi această structură poate apoi fi îmbunătăţită prin simulare cu un program de simulare. De asemenea, această metodă introduce noţiuni cantitative foarte necesare în cooperarea şi comunicarea celor care proiectează circuite rapide CMOS.

1.6 REJECŢIA ZGOMOTELOR

Prin semnal de zgomot se înţelege orice semnal electric nedorit care apare în sistem. Zgomotul este tolerat în sistem atât timp cât suprapus peste semnalul logic nu se ajunge la o amplitudine de semnal care să ducă la o funcţionare incorectă a sistemului. În sistemele digitale, spre deosebire de cele analogice, zgomotul nu se cumulează când se trece de la un nivel logic (poartă) la următorul nivel logic; această eliminare a zgomotului se datorează funcţionării "procustiene" a porţilor logice, adică o poartă dacă este comandată în limita valorilor de intrare permise va genera la ieşire numai semnale în limita valorilor garantate.

O primă cale prin care porţile pot tolera zgomotul, până la o anumită valoare, se obţine, intrinsec, prin fixarea de valori garantate şi de valori permise. Amplitudinea

semnalului de zgomot tolerat de către o poartă logică se poate exprima prin parameterii: marginea de zgomot în curent continuu în starea high, M_H, şi în starea low, M_L, relaţia 1.18; imunitatea la perturbaţii IP^+, IP^-, relaţia 1.19-a şi factorul de imunitate la perturbaţii FIP^+ şi FIP^-, relaţia 1.19-b.

Pentru tehnologia TTL valorile tipice pentru tensiunile de ieşire garantate şi cele de intrare permise sunt: $V_{OHmin} = 2,4V$, $V_{OLmax} = 0,4V$ şi $V_{IHmin} = 2V$, $V_{ILmax} = 0,8V$ deci rezultă:

$$M_H = 2,4V - 2V = 0,4V,\ M_L = 0,8V - 0,4V = 0,4V.$$

De fapt, zgomotul tolerat de poartă are valori mai mari decât marginile calculate, de $0,4V$, deoarece valorile tipice de ieşire sunt $V_{OH} = 3,4V$, $V_{OL} = 0,25V$. Considerând tensiunea de prag de comutare a porţii $V_T = 1,7V$ rezultă

$$IP^- = 3,4V - 1,7V = 1,7V,\ IP^+ = 1,7V - 0,25V = 1,45V$$

şi pentru $\Delta V = 5V$ se pot calcula factorii de imunitate la zgomot

$$FIP^-[\%] = \left(\frac{1,7}{5}\right) \times 100 = 34\%,\ FIP^+[\%] = \left(\frac{1,45}{5}\right) \times 100 = 29\%$$

deci, circuitul este mai bine protejat la zgomot în starea H decât în starea L.

Tehnologia CMOS, când comandă tot porţi CMOS, are următoarele valori tipice: $V_{OHmin} = V_{DD} - 0,1V$, $V_{OLmax} = V_{SS} + 0,1V$ şi $V_{IHmin} = 70\%$ din V_{DD}, $V_{ILmax} = 30\%$ din V_{DD} iar pentru $V_{DD} = 5V$ şi $V_T = 2,5V$ rezultă

$$M_H = 4,9V - 3,5V = 1,4V,\ M_L = 1,5V - 0,1V = 1,4V.$$

$$IP^- = 4,9V - 2,5V = 2,4V,\ IP^+ = 2,5V - 0,1V = 2,4V$$

$$FIP^-[\%] = \left(\frac{2,4}{5}\right) \times 100 = 48\%,\ FIP^+[\%] = \left(\frac{2,4}{5}\right) \times 100 = 48\%$$

În mediile puternic perturbative este recomandată utilizarea porţilor cu imunitate ridicată la zgomot, la care IP^+ şi IP^- au valori absolute destul de mari, pentru că tensiunile de alimentare sunt destul de ridicate, se poate ajunge până la $30V$.

O a doua cale de a tolera zgomotele de către porţile logice constă în dotarea porţilor cu circuit trigger Schmitt. Pentru porţile TTL circuitul trigger Schmitt se introduce după tranzistorul T1 şi va comanda tranzistorul inversor T2, Figura 1.22-c, iar pentru cele CMOS triggerul Schmitt va fi plasat înaintea unui inversor CMOS. Capacitatea de a tolera zgomot, de amplitudine destul de ridicată, rezultă din caracteristica statică de releu cu histerezis a circuitului trigger Schmitt, $V_O = f(V_I)$, Figura 1.69-a. Din caracteristica statică (cadranul I) se observă că pentru sensul crescător al tensiunii de intrare V_I numai când această tensiune atinge pragul superior V_{p^+}, ieşirea triggerului va bascula în starea 1; iar la descreşterea tensiunii de intrare bascularea ieşirii din 1 în 0 nu se produce la pragul superior V_{p^+} ci la pragul inferior V_{p^-} ($V_{p^-} < V_{p^+}$), valoarea histerezisului fiind $\Delta = V_{p^+} - V_{p^-}$.

La o poartă logică cu trigger Schmitt tensiunea de intrare, V_I, va fi încă interpretată ca 1 logic până când această tensiune (peste care, eventual, se suprapune semnal de zgomot cu fază opusă) scade sub valoarea V_{IH} până la valoarea V_{p^-}. Asemănător,

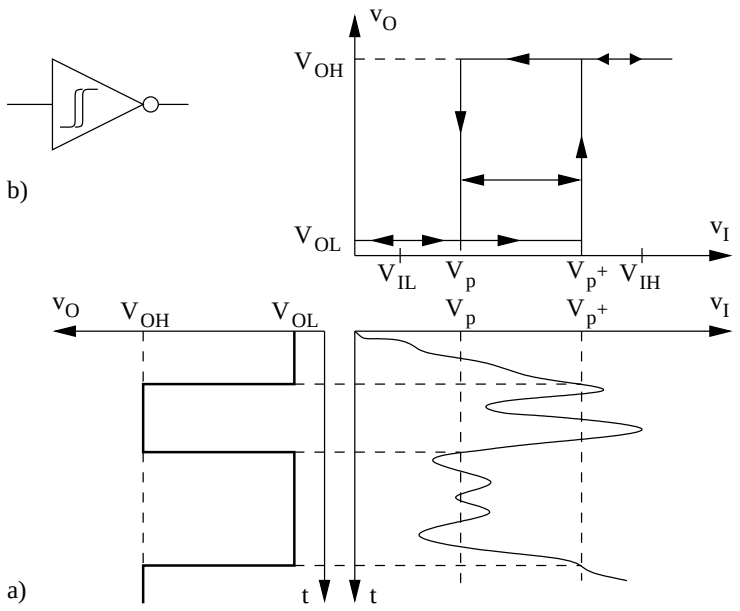
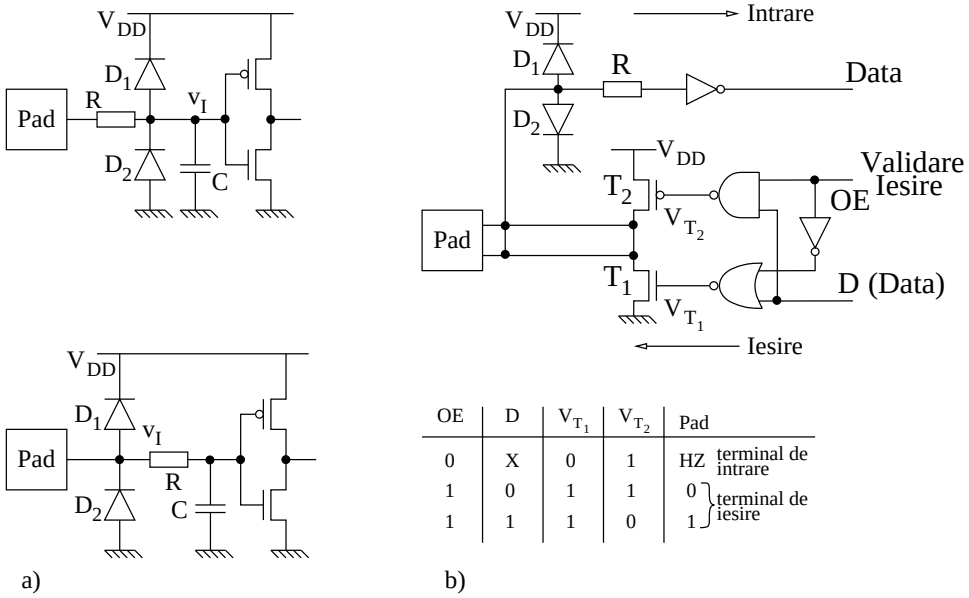


Figura 1.69 Triggerul Schmitt: a) caracteristica de releu cu histerezis (cadranul I), variaţia tensiunii de intrare (cadranul IV) şi variaţia corespunzătoare a tensiunii de ieşire (cadranul III); b) simbolul de reprezentare a unui buffer neinversor cu trigger Schmitt.

tensiunea de intrare (eventual cu semnal de zgomot suprapus) va fi încă interpretată ca 0 logic peste V_{IL} până când se ajunge la valoarea V_{p^+}. Intervalele tensiunilor de intrare în 1 şi 0 logic se suprapun pe intervalul de histerezis Δ. Marginile statice de zgomot pentru o poartă cu trigger Schmitt se calculează cu relaţiile: $M_H = V_{OHmin} - V_{p^-}$, $M_L = V_{p^+} - V_{OLmax}$, care, evident, sunt mai mari decât la o poartă obişnuită.

Porţile trigger Schmitt sunt recomandate în aplicaţiile, pentru care imunitatea la zgomot este o cerinţă principală, cum ar fi receptoarele de magistrală sau pentru recepţia unor semnale lent variabile (cazul semnalelor de intrare în sisteme, semnale obţinute de la diverse traductoare "care trebuie formate"). În Figura 1.69-a în cadranul IV este reprezentată variaţia în timp a unui semnal oarecare $v_I = f(t)$ aplicat la intrarea unui buffer neinversor trigger Schmitt iar în cadranul III este desenată variaţia corespunzătoare a tensiunii la ieşirea din buffer; pragurile de basculare fiind V_{p^-} şi V_{p^+}. (Sugerăm să desenaţi forma de variaţie a tensiunii de ieşire $v_O = f(t)$, obţinută pentru aceeaşi tensiune de intrare, dar când bufferul nu este un circuit trigger Schmitt şi apoi să comparaţi, tensiunea obţinută cu forma de variaţie a tensiunii de ieşire deja desenată în cadranul III).

Semnalele de zgomot nu trebuie considerate numai ca surse potenţiale de producere a unor funcţionări eronate ci şi ca posibile cauze de distrugere fizică a porţilor, mai ales când aceste semnale vin din exterior şi sunt aplicate pe intrări. Supratensiunile aplicate pe terminalele de intrare pot duce la străpungerea unor joncţiuni sau, pentru CMOS, la străpungerea stratului de oxid de sub poartă, ori la apariţia efec-



OE	D	V_{T_1}	V_{T_2}	Pad	
0	X	0	1	HZ	terminal de intrare
1	0	1	1	0	terminal de iesire
1	1	1	0	1	terminal de iesire

Figura 1.70 Terminale (paduri) la un circuit integrat: a) circuite de protecţie la un terminal de intrare; b) structură de circuit pentru utilizarea unui pad atât pentru semnale de ieşire cât şi semnale de intrare.

tului de zăvorâre ("latch-up"). Aceste supratensiuni pot apare la intrările porţilor datorită: reflexiilor pe liniile neadaptate, tensiunilor induse, alimentarea sistemului cu mai multe niveluri de tensiune, descărcărilor electrostatice.

Cea mai simplă, şi uzuală, cale de a **proteja o poartă** pe terminalele de intrare sau pe pad-uri (la sistemele integrate) constă în conectarea unor diode de descărcare înspre cele două bare de alimentare: V_{SS}(masă), V_{DD}/V_{CC}. Pentru porţile TTL diodele D4 şi D5 pe intrări, Figura 1.22-c, limitează supratensiunile negative la valoarea de $-0,7V$ (tensiunea de conducţie a diodei) sub potenţialul masei. Pentru terminalele de intrare în tehnologia CMOS este utilizată pentru protecţie structura de circuit din Figura 1.70-a care pentru tensiuni de intrare peste V_{DD} sau sub V_{SS}, prin diodele D1, D2, produce o limitare în interiorul intervalului $-0,7V < v_I < V_{DD} + 0,7V$. La această structură de protecţie prin C se consideră orice capacitate existentă pe intrare, iar rezistenţa R se realizează prin difuzie sau, mai indicat, din polisiliciu; evident, constanta de timp RC care apare pe intrare poate introduce o limitare a vitezei pentru circuitele de viteză ridicată.

Distrugerea unei porţi CMOS prin descărcare electrostatică poate apare nu numai în condiţii de utilizare a porţii ci şi în situaţia de manipulare/transport. O persoană mergând pe un covor, în condiţiile de 80% umiditate relativă, se încarcă electrostatic care, apoi, prin atingerea intrării unei porţi logice poate genera o tensiune de ordinul sutelor de volţi aplicată pe stratul de oxid de sub poartă (tensiunea de străpungere a stratului de oxid este între $40 \div 100V$). De exemplu, pentru o astfel de atingere, cu următoarele valori $C = 0,3pF$, $I = 100\mu A$, $\Delta t = 1\mu s$, se generează

pe stratul de oxid o tensiune egală cu $V = I \cdot \Delta t/C \approx 330V$; evident, poarta ar fi distrusă fără circuitul de protecţie cu diode de descărcare pe intrare. Ca măsuri de precauţie la porţile CMOS se recomandă:

1. manipularea/transportul se face numai în pungi, tuburi sau spumă care sunt conductivi;

2. în timpul asamblării sau depanării sistemelor, persoana executantă să fie conectată cu un fir spre pământ.

Efectul de zăvorâre se manifestă prin apariţia în interiorul plachetei de siliciu a unui traseu de scurtcircuitare între V_{DD} şi masă care, dacă nu este eliminat, poate duce la distrugerea cipului; acest efect este cel mai negativ efect al tehnologiei CMOS. Efectul de zăvorâre are ca explicaţie formarea unei structuri (parazite) de tiristor. Între zonele difuzate (sursă, dren, contacte metalice) şi substrat se formează structuri care, atunci când sunt polarizate corespunzător, au funcţionare de tranzistor *npn* sau *pnp*. Structurile de tranzistoare complementare parazite *npn* şi *pnp* împreună pot determina o funcţionare de tiristor care, în anumite condiţii de funcţionare anormală ale porţii, poate intra în conducţie deci are loc scurtcircuitarea barei V_{DD} spre masă. Tiristorul parazit odată amorsat nu mai poate fi comandat spre blocare – de unde şi denumirea de zăvorâre pentru acest efect; eliminarea unei astfel de căi de scurtcircuit, prin blocarea tiristorului, se poate face doar prin deconectarea tensiunii de alimentare. Circuitele logice CMOS sunt fabricate în structuri care previn efectul de zăvorâre [Weste ′00],[Kang ′97].

Intrarea în zăvorâre poate fi determinată de:

- aplicarea, la pornire, a tensiunii de alimentare V_{DD} cu un front de creştere cu pantă foarte mică (lent variabilă);

- variaţii mari ale semnalului de intrare peste V_{DD} sau V_{SS};

- intrările unui sistem/subsistem sunt comandate de ieşirile unui alt sistem/subsistem iar cele două sisteme au surse de alimentare diferite (semnalele de intrare la sistemul comandat ar putea fi aplicate înainte ca sistemul comandat să fie alimentat la V_{DD});

- curenţi de scurgere prin joncţiunile insulei de izolare, particule γ, radiaţii X sau cosmice.

Padurile circuitelor integrate sunt mari consumatoare de suprafaţă de siliciu apoi conexiunile de la paduri la pinii (terminalele) circuitului integrat ocupă spaţiu relativ mare; în plus comanda unui pad necesită un superbuffer, vezi Exemplul 1.28. Cu creşterea complexităţii circuitelor creşte şi numărul de pini (de ordinul sutelor) la circuitele integrate deşi există permanenta tendinţă de a ţine acest număr cât se poate mic. O variantă practică, în acest sens, este cea prin care un terminal este dedicat (multiplexat) atât pentru semnalele de intrare cât şi pentru semnale de ieşire, Figura 1.70-b. Din tabelul de adevăr ataşat acestei figuri rezultă că atunci când semnalul de validare ieşire OE (Output Enable) este activ data interioară D este transmisă la pad, deci padul constituie un terminal de ieşire. Pentru $OE = 0$, ambele tranzistoare T1 şi T2 sunt blocate, ieşirea înspre pad, din interiorul circuitului, este în starea de

înaltă impedanţă HZ, pe pad se poate aplica un semnal din exterior, deci este un terminal de intrare. Evident, aplicarea semnalului din exterior şi aplicarea semnalului din interior trebuie să fie operaţii disjuncte.

Terminalele circuitelor integrate **neutilizate** în sistemul respectiv nu trebuie lăsate flotante pentru că pot să capteze zgomote, mai ales la CMOS unde impedanţele de intrare sunt mai mari, iar circuitul poate manifesta o funcţionare hazardată. Problema terminalelor de intrare neutilizate poate fi soluţionată în trei modalităţi:

1. Se leagă împreună cu un alt terminal de intrare conectat la ieşirea unei porţi logice (evident că poarta respectivă este încărcată suplimentar pe ieşire);

2. Se leagă la tensiunea de alimentare V_{DD}/V_{CC} (1 logic) printr-o rezistenţă pentru porţile AND sau NAND;

3. Se leagă la potenţialul masei (0 logic) printr-o rezistenţă pentru poarţile OR, NOR.

Valoarea rezistenţei, de conectare, la masă sau la alimentare a intrărilor neutilizate, poate fi în intervalul $1 \div 10K\Omega$ pentru porţile CMOS, dar trebuie calculată exact pentru porţile TTL. La porţile TTL, care au curenţii de intrare mult mai mari decât la CMOS, este necesar ca valoarea căderilor de tensiune pe rezistenţa conectată la masă sau V_{CC} să situeze nivelul de tensiune produs pe intrare în plaja tensiunilor garantate $\leq V_{OLmax}$, $\geq V_{OHmin}$. De exemplu, pentru porţile LS-TTL care pe intrare în starea L au un curent de $0,4mA$ rezisteţa de conectare la masă R_{cm} a n intrări neutilizate se calculează cu relaţia $n \cdot 0,4mA \cdot R_{cm} \leq V_{OLmax}$; respectiv rezistenţa de conectare la V_{CC}, R_{ca}, când pe intrare în starea H se absoarbe un curent de $20\mu A$ se calculează cu relaţia $n \cdot 0,02mA \cdot R_{ca} < V_{CC} - V_{OHmin}$.

Majoritatea sistemelor integrate sunt realizate în tehnologia CMOS, dar în exterior unde, uneori, curenţii necesari au valori ridicate se utilizează şi circuite în tehnologie TTL, deci adesea apare o comandă de tip TTL pe o intrare a unui circuit CMOS. Valorile garantate pe ieşirile TTL sunt $V_{OLmax} = 0,4V$, $V_{OHmin} = 2V$, iar pentru CMOS valorile permise pe intrare sunt $V_{ILmax} = 0,8V$, $V_{IHmin} = 3,5V$; se observă că nu există compatibilitate pentru nivelul H, tensiunea generată de TTL ar trebui să fie ridicată cu cel puţin $1,5V$. Soluţia pentru compatibilizare constă în realizarea inversorului CMOS de pe intrare, după circuitele de protecţie, Figura 1.70-a, cu o tensiune de prag V_T situată la mijlocul intervalului între $0,8V$ şi $2V$ adică $V_T = 1,4V$. Cunoscând valoarea lui V_T se pot calcula dimensiunile porţilor W_p şi W_n astfel încât deplasarea de nivel de tensiune, de la $3,5V$ la $2V$, să se realizeze în jos, către $2V$ de către inversorul de intrare CMOS.

Se poate defini şi o mărime **margine dinamică de zgomot**, aceasta fiind amplitudinea zgomotului mai mare decât M_H, M_L, cu o durată mai mică decât timpul de propagare minim prin poartă, dar care nu provoacă comutarea ieşirii porţii. Implusurile de amplitudine mare, dar de durată mai mică decât timpul de propagare, nu sunt "simţite" la ieşire. Marginea dinamică de zgomot nu este garantată pentru o poartă de către fabricant. Este foarte greu să se garanteze valoarea minimă a timpului de propagare, τ_{pmin}, în catalog pentru o poartă este dată doar valoarea maximă a timpului de propagare, τ_{pmax}.

Pentru eliminarea sau atenuarea zgomotului într-un circuit este necesar a se cunoaşte natura şi locul sursei de zgomot. Primul pas necesar spre această cunoaştere

este o clasificare a zgomotului după locul unde se află sursa de zgomot; în acest sens trebuie să distingem dacă sursa de zgomot este internă sau externă circuitului şi, apoi, pentru această sursă să se identifice natura fenomenului producător de zgomot.

1.6.1 Rejecţia zgomotelor externe

Zgomotul extern are ca sursă: instalaţiile de electronică de putere, motoarele electrice, motoarele termice, comutaţiile în cablurile electrice de forţă, supratensiunile pe linia electrică, instalaţiile de înaltă frecvenţă de putere , transmisiunile emiţătoarelor RTV şi alte surse de radiaţii electromagnetice şi este introdus în sistem prin inducţie electromagnetică sau prin conducţie pe firele de alimentare de la reţea.

Zgomotul de conducţie. Acest zgomot poate intra în sistem prin firele de alimentare de la reţea, se poate elimina sau atenua prin: separare galvanică şi/sau filtrare (folosite, în general, împreună).

Separarea galvanică a sistemului de reţeaua de alimentare se face prin transformatorul redresorului de alimentare. Chiar dacă adaptarea nivelurilor de tensiune între reţeaua electrică şi sursa de alimentare (redresor stabilizat) nu necesită un transformator, totuşi se recomandă introducerea separării printr-un transformator cu raportul de transformare 1 : 1.

Prin filtrare se pot elimina atât frecvenţele joase cât şi cele ridicate generate de către sursa de alimentare. În acest sens se recomandă conectarea la intrarea pe placa de circuit imprimat, între barele de alimentare V_{CC}/V_{DD} şi masă, a unui filtru capacitiv. Cu cât capacitatea conectată pe intrare are valoare mai mare cu atât este mai bine dar, evident, o valoare mare a capacităţii poate fi limitată de preţ şi volumul ocupat. Practic, această capacitate conectată la intrarea pe placa de circuit imprimat a sistemului se realizează cu două condensatoare: unul electrolitic, de valori uzuale în gama $50 \div 100\mu F$, pentru filtrarea frecvenţelor joase şi medii şi unul ceramic, de capacitate în gama zecimi de μF, pentru filtrarea frecvenţelor înalte. De asemenea, pentru placa de circuit imprimat a sistemului o **legătură bună la pământ**, care să aibă o rezistenţă cât mai mică chiar şi pentru domeniul frecvenţelor radio, dar separată de împământarea la reţea, este foarte recomandată.

Zgomotul electromagnetic. Acest tip de zgomot pătrunde în sistem prin inducţie electromagnetică deci poate fi anulat sau atenuat prin ecranarea sistemului supus inducţiei cu ajutorul unui ecran din materiale feroase – cuşcă Faraday – şi legarea acestui ecran la priza de împământare şi printr-un filtru capacitiv şi la sursa de alimentare. Pătrunderi de câmp ce pot apărea în sistem prin zonele de întrerupere ale ecranului, fante pentru acces, orificii pentru cablurile cu exteriorul şi aceste pătrunderi pot duce în interior la o inducţie în sistem.

Dar cel mai pregnant efect al inducţiei electromagnetice constă în tensiuni de zgomot induse în conexiunile exterioare ecranării care leagă diferite părţi ale sistemului cu exteriorul. La transmisia unui semnal pe un singur fir şi reîntoarcerea prin traseul de masă, tensiunea obţinută la receptor se compune din tensiunea de la emiţător plus oricare semnal de zgomot indus pe linia de legătură v_{zl} sau pe linia de masă v_{zm}, Figura 1.71-a. Evident că, pentru o funcţionare corectă, marginea de zgomot a receptorului trebuie să fie mai mare decât amplitudinea maximă a zgomotului indus pe linie. Ideal, se recomandă ca legăturile exterioare să se facă prin cablu coaxial iar îmbrăcămintea (ecranarea) acestuia, la ambele capete, să fie legată la pământ şi să

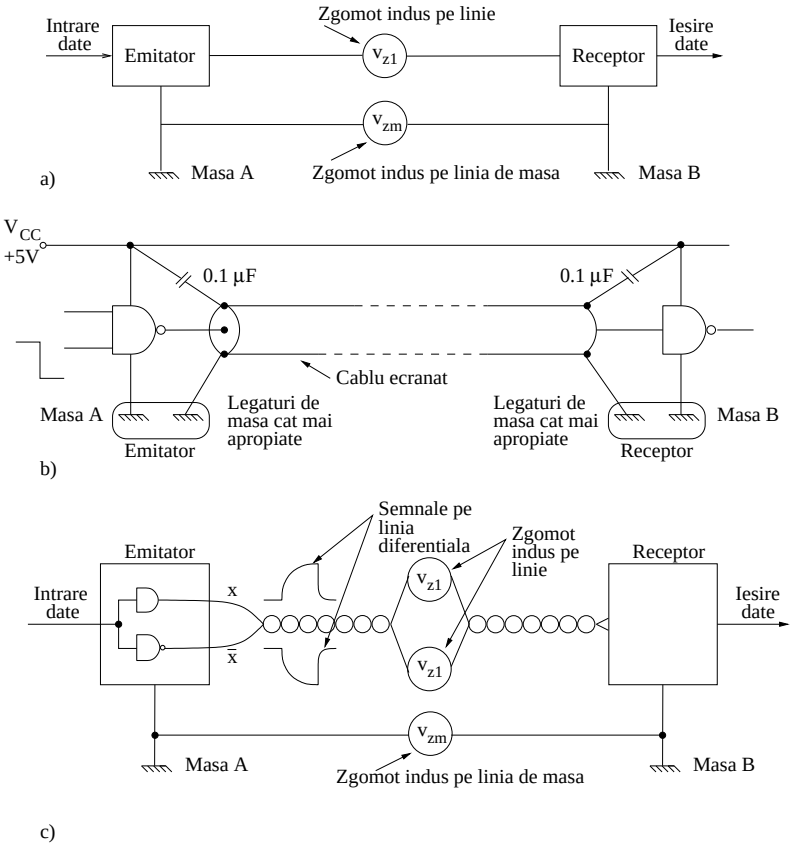


Figura 1.71 Zgomote externe: a) exemplificare pentru inducerea semnalelor de zgomot pe o linie de conexiune;b) eliminarea posibilităţii de inducere a zgomotului de inducţie prin utilizarea de cablu ecranat; c) eliminarea zgomotului suprapus peste semnal prin utilizarea modului de transmisie diferenţială.

se decupleze la sursă, Figura 1.71-b. Iar când legăturile exterioare sunt realizate prin cablu plat se recomandă ca între două fire, de transmisie pentru semnal, să fie utilizat un fir ca masă şi aceste fire de masă să fie legate la carcasă/ecran.

O soluţie foarte eficientă în eliminarea zgomotului indus constă în transmisia diferenţială a semnalului. Semnalul digital x ce trebuie transmis este convertit la emiţător şi în complementul său, $\overline{x}$, iar aceste două semnale complementare, x şi $\overline{x}$, sunt transmise pe două linii torsodate, Figura 1.71-c. Orice zgomot v_{zl} indus pe cele două linii sau pe traseul de masă este un semnal de mod comun, deci semnalele la receptor sunt: $x+v_{zl}$ şi $\overline{x}+v_{zl}$. Receptorul diferenţial va realiza la ieşire un semnal proporţional cu diferenţa celor două semnale aplicate la intrare, adică elimină semnalul de mod comun. Prin transmisia diferenţială se poate obţine la receptor un semnal "curat" chiar cu circuite care nu posedă imunitate ridicată la zgomot; atât emitătorul cât şi receptorul sunt alimentate la tensiunea standard de $+5V$.

1.6.2 Rejecţia zgomotelor interne

Zgomotul intern, după cum şi denumirea spune, este produs chiar de însuşi sistemul respectiv, iar în funcţie de natura fenomenului care îl generează pot fi identificate următoarele tipuri: zgomotul de masă, zgomotul datorită neadaptării liniilor, zgomotul indus prin cuplaj electromagnetic şi zgomotul datorită curenţilor de alimentare. În general, zgomotul intern poate cauza mai multe probleme decât cel extern ("răul este în noi!"). Este bine demonstrat că o proiectare şi o execuţie corectă şi îngrijită a sistemului sunt premise sigure pentru evitarea apariţiei zgomotului intern.

1.6.2.1 Zgomotul de masă.

Prin **masă electronică** într-un circuit se înţelege potenţialul de referinţă pentru toate tensiunile din circuit, fizic masa electronică este materializată printr-un conductor/traseu la care se conectează toate componentele circuitului; evident acest traseu trebuie să fie echipotenţial pentru toate componentele. Aplicarea unui semnal sau culegerea unui semnal de prelucrat se face pe un traseu compus din conductorul de semnal (firul cald) cu întoarcere prin conductorul de masă. Conductorul de masă fiind comun pentru toate circuitele de aplicare sau de culegere a semnalelor rezultă că acest conductor este parcurs de toţi curenţii de întoarcere ai semnalelor. Dacă rezistenţa conductorului de masă nu este nulă curenţii de întoarcere produc căderi de tensiune, iar diferitele puncte ale traseului de masă nu mai sunt echipotenţiale, deci componentele circuitului au potenţiale de masă diferite. Considerând punctul unde se conectează masa sursei de alimentare la masa circuitului/sistemului alimentat ca punct iniţial de masă, în cazul în care traseele de masă pornind din punctul iniţial nu au rezistenţă zero, potenţialul de masă al fiecărei componente va fi diferit faţă de potenţialul punctului iniţial de masă în funcţie de valorile curenţilor de întoarcere.

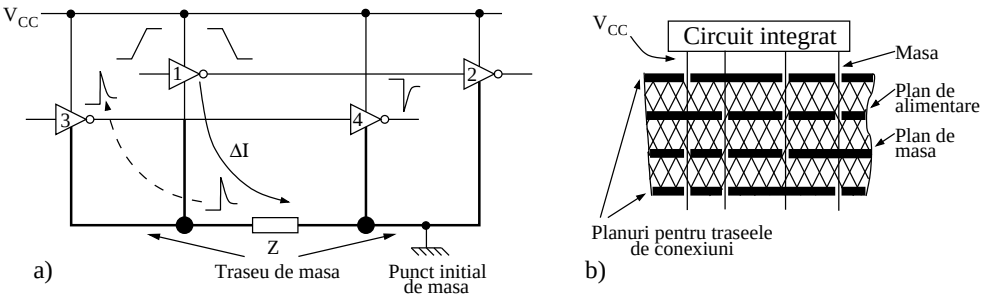


Figura 1.72 Zgomot de masă: a) exemplificare pentru o posibilă apariţie a semnalului de zgomot de masă datorită unei impedanţe, Z, pe traseul de masă; b) soluţie de eliminare a zgomotului de masă prin utilizarea unui plan separat de masă şi altul separat de alimentare (pe o placă multistrat).

De exemplu, pentru circuitul din Figura 1.72-a când ieşirea porţii1 (TTL) comută din 1 în 0 curentul de la sursă spre masă creşte cu valoarea ΔI. Deoarece, între punctul iniţial de masă şi punctul de conectare la masă al porţii 1, traseul de masă prezintă o impedanţă Z, se va produce un salt al căderii de tensiune ΔV pe această

impedanţă egal cu $\Delta V = \Delta I \cdot Z$. Punctul de masă al porţii 3 fiind foarte aproape de cel al porţii 1, saltul de tensiune produs ΔV poate comanda poarta 3 să genereze la ieşire un impuls fals care, apoi, este format (adus la nivelul 0) de către poarta 4, deci o funcţionare eronată a circuitului.

Remediu pentru zgomotul de masă apare prin realizarea unui traseu de masă echipotenţial pentru toate componentele circuitului ceea ce practic constă în:

1. Realizarea unui plan de masă în cazul utilizării unei plăci multistrat, Figura 1.72-b. La o placă multistrat poate fi utilizat în totalitate un strat numai pentru masa sistemului implementat, la fel, un alt strat numai pentru tensiunea de alimentare iar restul straturilor pentru conexiunile necesare (actual, există plăci cu până la 12 straturi metalizate);

2. La folosirea unui suport unistrat să se aloce traseului de masă o suprafaţă conductivă cât mai mare. Se recomandă ca traseul de masă, cu o lăţime cât mai mare, în funcţie de suprafaţă disponibilă pe placă, să înconjoare toate traseele de conexiuni, şi acest traseu (/suprafaţă) de masă să fie la ambele capete legat de masa sursei de alimentare.

1.6.2.2 Zgomotul datorită neadaptării liniilor.

În toate analizele de până acum s-a considerat că transferul semnalului pe traseul de conexiune între poarta care comandă şi o poartă comandată se face în timp zero, adică instantaneu, dar în realitate nu este aşa chiar dacă viteza de propagare v_p pe traseul de conexiune ar fi egală cu viteza luminii. Un traseu de conexiune, pe o placă de circuit imprimat, este caracterizat de mărimi electrice distribuite pe lungimea x a traseului. Astfel se definesc valorile, pe lungimea infinitezimală dx, pentru inductivitate, Ldx, şi pentru capacitate, Cdx, unde L şi C sunt valori pe unitatea de lungime. Viteza de propagare pe un astfel de traseu se poate exprima prin relaţia:

$$v_p = \frac{dx}{dt} = \frac{1}{\sqrt{LC}}[m/s] \tag{1.106}$$

sau în funcţie de viteza luminii, c:

$$v_p = \frac{1}{\sqrt{\mu_0 \varepsilon_r \varepsilon_0}} = \frac{c}{\sqrt{\varepsilon_r}}[m/s] \tag{1.107}$$

unde ε_0 şi μ_0 sunt respectiv permitivitatea electrică şi permeabilitatea magnetică a vidului iar ε_r este permitivitatea electrică relativă (pentru plăcile de sticlotextolit este în jur de $\varepsilon_r = 4,5$). Rezultă viteza de propagare v_p pentru circuitele imprimate în domeniul $15cm/ns \div 25cm/ns$ respectiv timpi de propagare pe metru în domeniul $7ns/m \div 4ns/m$. Pentru ca semnalul să parcurgă traseul de lungime l între două porţi de două ori, de la ieşirea porţii de comandă la intrarea porţii comandate şi reflectat înapoi la ieşirea porţii de comandă, este necesar timpul $2T = 2l/v_p$. Pentru cazurile când, Figura 1.15, fronturile de tranziţie ale semnalelor logice, τ_{HL}, τ_{LH} , sau timpii de propagare τ_p prin porţi, relaţia 1.20, sunt de acelaşi ordin sau mai mici decât $2T$ nu se mai poate utiliza analiza în curent continuu pentru procesul de tranziţie al semnalului. Se poate utiliza şi analiza în curent continuu dar numai când procesul

de transfer a ajuns în regim static, adică nu mai există variaţii, ceea ce practic ar corespunde parcurgerii lungimii l cam de cinci ori, $5T$.

Pentru analiza procesului tranzistoriu pe traseu este necesar să se utilizeze teoria liniilor de transmisie. Condiţia ca o linie de lungime l, care este parcursă cu viteza v_p, să fie considerată linie lungă, pentru un impuls cu durata τ, este exprimată prin relaţia $\tau \leq 2l/v_p = 2T$; adică durata semnalului este mai mică decât timpul necesar de parcurgere dus şi întors a traseului de conexiune (amintim că un semnal cu cât are variaţii mai rapide cu atât spectrul său de frecvenţă, B, este mai larg, ceea ce se poate exprima simplu cu relaţia $\boldsymbol{B \cdot T = 1}$, unde T este perioada semnalului respectiv). Cu cât viteza porţilor creşte cu atât lungimea de traseu care determină încadrarea în linie lungă se micşorează, de exemplu, pentru două porţi cu $\tau_{p_1} = 4ns$, $\tau_{p_2} = 1ns$ linia lungă se reduce respectiv la lungimile de trasă de $(4ns \times 15cm/ns)/2 \geq 30cm, (1ns \times 15cm/ns)/2 \geq 7,5cm$.

Un traseu de transmisie este caracterizat prin **impedanţă caracteristică**, Z_0. Impedanţă caracteristică este definită ca raportul dintre tensiunea tranzistorie, v, pe linie şi curentul tranzistoriu, i, generat în linie şi poate fi calculată prin relaţia:

$$Z_0 = \frac{v}{i} = \sqrt{\frac{L}{C}}\,[\Omega] \tag{1.108}$$

Valori uzuale pentru impedanţa caracteristică sunt: trasă de circuit pe placă de textolit pentru circuit imprimat $50 \div 150\Omega$, cablu coaxial 50Ω, cablu bifilar torsodat 120Ω, cablu plat $80 \div 120\Omega$.

Pe un traseu de transmisie ori de câte ori pentru impedanţa liniei există o discontinuitate apar reflexii ale semnalului, adică o parte din energia semnalului incident este îndreptată înapoi, Figura 1.73-a. Într-un punct al traseului când se trece de la impedanţa liniei Z_0 la impedanţa Z_r ($Z_0 \neq Z_r$)se defineşte un **coeficient de reflexie**, k_r, prin relaţia:

$$k_r = \frac{Z_r - Z_0}{Z_r + Z_0} \tag{1.109}$$

Dacă semnalul incident V ajunge în punctul de reflexie, de coeficient de reflexie k_r, atunci în urma reflexiei se propagă înapoi pe linie un semnal reflectat $v_r = k_r \cdot V$. Imediat după reflexie în punctul de reflexie semnalul rezultat (total), v_t, prin aplicarea principiului superpoziţiei, este egal cu semnalul incident V plus semnalul reflectat v_r:

$$v_t = V + v_r = (1 + k_r)V \tag{1.110}$$

În punctele de traseu, când tensiunea reflectată se propagă în sens invers, tensiunea rezultantă este egală cu tensiunea care exista în acel punct, în acel moment, plus tensiunea reflectată v_r. Pentru plaja de impedanţe $Z_r \in [0, \infty]$, care pot fi întâlnite într-un punct de reflexie, rezultă, din relaţia 1.109, pentru coeficientul de reflexie că are valori în intervalul $k_r = [-1, 1]$.

Pentru o linie de transmisie se vor analiza trei cazuri (limită) de reflexie după cum linia se termină pe impedanţele $Z_r = 0$, $Z_r = Z_0$ sau $Z_r = \infty$.

- Linia de transmisie **terminată în scurtcircuit,** $Z_r = 0$. Rezultă $k_r = -1$, $v_r = k_r V = -V$ iar $v_t = V - V = 0$. Imediat după reflexia semnalului incident V, în punctul de terminare pe impedanţă Z_r, tensiunea rezultată este zero şi în

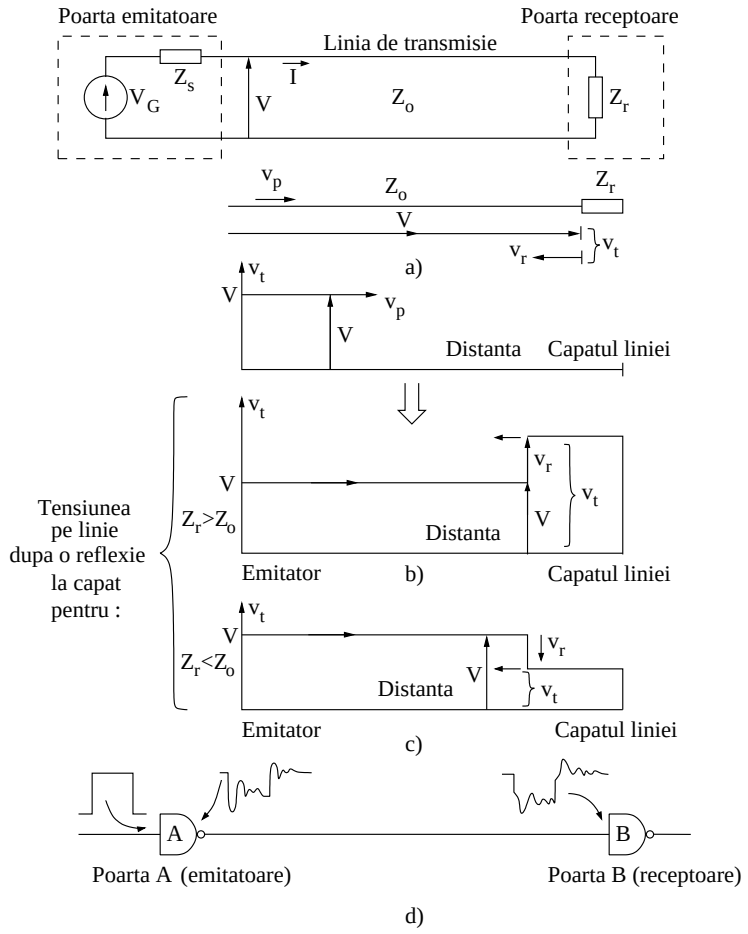


Figura 1.73 Reflexia semnalului pe liniile de conectare între porţi(a): b) reflexia pentru cazul când $Z_r > Z_0$; c) pentru cazul când $Z_r < Z_0$; d) exemplu de forme de semnale la transferul dintre două porţi.

continuare devine zero tensiunea pe linie pe măsură ce tensiunea reflectată v_r se propagă înapoi pe linie (semnalul se "stinge").

- Linia de transmisie **terminată pe impedanţă caracteristică ,** $Z_r = Z_0$ (linie adaptată). Rezultă $k_r = 0$, $v_r = 0 \cdot V = 0$ iar $v_t = V + 0$. Pentru o linie adaptată nu există componentă reflectată, după timpul T de propagare a semnalului pe linie până la punctul de impedanţă Z_r, procesul devine staţionar, iar tensiunile se pot calcula după legea lui Ohm.

- Linie de transmisie **terminată în gol,** $Z_r = \infty$. Rezultă $k_r = +1$, $v_r = k_r V = V$, $v_t = V + v_r = 2V$. Deoarece semnalul reflectat are amplitudinea egală cu V tensiunea pe linie devine egală cu $2V$, pe măsură ce tensiunea reflectată se propagă înapoi, începând cu punctul de reflexie (liniile care se termină pe intrările porţilor CMOS, care au impedanţa de intrare foarte mare $R > 10^{12}\Omega$,

pot fi considerate ca linii în gol).

Pentru două cazuri $Z_r > Z_0$ şi $Z_r < Z_0$, în Figura 1.73-b şi c se reprezintă simplificat modul cum se obţine tensiunea totală pe linie, v_t, prin sumarea tensiunii incidente, V (spre poarta receptoare) cu tensiunea reflectată, v_r, de către impedanţa de intrare, Z_r, în poarta receptoare.

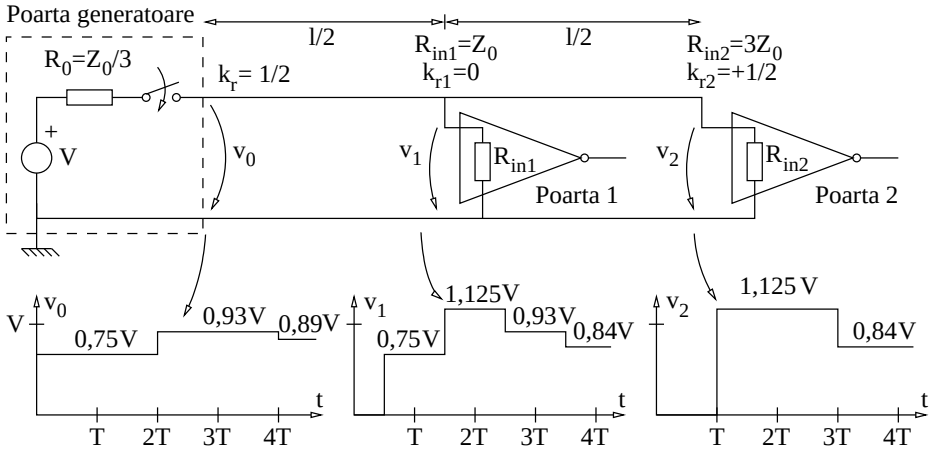


Figura 1.74 Variaţia tensiunii pe o linie de transmisie cu puncte de reflexie la ambele capete.

În Figura 1.74 este reprezentată schema echivalentă a unei porţi a cărei ieşire, prin intermediul unei linii de transmisie de impedanţă Z_0 şi lungime l, comandă în punctul terminus intrarea unei alte porţi. Poarta de comandă este echivalată cu un generator de tensiune V şi o rezistenţă de ieşire $R_O = Z_0/3$, iar intrarea porţii comandate din punctul terminus este echivalată numai prin rezistenţa de intrare $R_{in_2} = 3Z_0$. La mijlocul liniei de transmisiune, $l/2$, mai este conectată o poartă care are rezistenţa de intrare $R_{in_1} = Z_0$. Se va analiza variaţia tensiunii v_O la ieşirea porţii de comandă şi a tensiunilor v_1 şi v_2 la intrările porţilor comandate, când tensiunea generatorului are un salt de la zero la valoarea V.

La momentul $t = 0$ tensiunea generatorului se aplică pe divizorul rezistiv format din R_0 şi Z_0 (semnalul nu a început să se propage pe linie) deci la ieşirea porţii de comandă rezultă tensiunea egală cu $v_O = \left(\frac{V}{Z_0+Z_0/3}\right) Z_0 = 3/4V = 0,75V$. După timpul $(l/2)/v_p = T/2$ semnalul cu amplitudinea $3/4V$ ajunge la Poarta1 la a cărei intrare se va aplica tensiunea $v_1(T/2) = 3/4V$ deoarece nu există reflexie, intrarea fiind adaptată, $k_r = 0$. După încă un interval $T/2$ semnalul ajunge la intrarea Porţii2 de la capătul terminus unde suferă o reflexie cu un coeficient de reflexie $k_{r2} = (3Z_0 - Z_0)/(3Z_0 + Z_0) = 1/2$. Tensiunea reflectată este $v_r = 1/2 \times 3/4V = 3/8V$, iar tensiunea totală aplicată porţii rezultă $v_2(T) = 3/4V + 3/8V = 9/8V = 1,125V$. Unda de tensiune reflectată după intervalul de timp $T/2$ ajunge la Poarta1 la a cărei intrare se aplică tensiunea $v_1(3T/2) = 3/4V + 3/8V = 9/8V = 1,125V$, iar după încă un interval de $T/2$ atinge punctul de reflexie de la ieşirea porţii de comandă unde coeficientul de reflexie este $k_r = (Z_0/3 - Z_0)/(Z_0/3 + Z_0) = -1/2$. Tensiunea reflectată în acel punct este $v_r = (-1/2) \times 3/8V = -3/16V$; tensiunea rezultată

$v_0(2T)$ se compune din tensiunea existentă $3/4V$, din tensiunea incidentă $3/8V$ şi din tensiunea reflectată, deci $v_0(2T) = (3/4 + 3/8 - 3/16)V = 15/16V = 0,93V$. Urmează a doua parcurgere spre capătul terminus al traseului de către semnalul reflectat $v_r = -3/16V$ când se obţin la Poarta1: $v_1(5T/2) = (9/8 - 3/16)V = 15/16V = 0,93V$ şi la Poarta2: $v_2(3T) = (9/8 - 3/16 - 3/32)V = 27/32V = 0,84V$; în punctul terminus $9/8V$ este tensiunea existentă, $-3/16V$ este tensiunea incidentă iar $1/2(-3/16)V = -3/32V$ este tensiunea reflectată. Se începe a doua parcurgere înapoi înspre începutul traseului de către semnalul reflectat $v_r = -3/32V$; se obţin la Poarta1: $v_1(7T/2) = (15/16 - 3/32)V = 27/32V = 0,84V$, la poarta generatoare $v_0(4T) = (15/16 - 3/32 + 3/64)V = 57/64V = 0,89V$. În punctul de început, de la poarta generatoare, se produce acum un semnal reflectat egal cu $v_r = (-1/2) \cdot (-3/32)V = +3/64V$. La următoarea parcurgere înspre capătul terminus se obţin la Poarta1: $v_1(9T/2) = (27/32 + 3/64)V = 57/64V = 0,89V$ şi la Poarta2: $v_2(5T) = (27/32 + 3/64 + 3/128)V = 117/128V = 0,91V$ şi un semnal reflectat $v_r = 3/128V$. În final, se ajunge în regimul staţionar când tensiunea aplicată la capătul linei de transmisiuni $v_2(\infty) = (V/(Z_0/3 + Z_0 + 3Z_0)) \times (Z_0 + 3Z_0) = 0,92V$.

Se observă din diagramele din Figura 1.74 că tensiunile v_1, v_2, intrările porţilor comandate, pot avea valori mai mari decât V. Când la intrarea liniei de transmisiune se aplică un salt de la 1 logic la 0 logic pot apărea la intrarea porţilor comandate tensiuni (negative) sub nivelul masei. Protecţia împotriva acestor supratensiuni la intrarea porţilor se face prin diode de descărcare ca în Figura 1.22-c şi Figura 1.70-a. În plus, în intervalele scurte când diodele de pe intrare conduc realizează o impedanţă de intrare la masă de valoare foarte mică, deci un coeficient de reflexie care tinde spre -1, ceea ce duce la atenuarea regimului tranzistoriu.

Eliminarea reflexiilor pe liniile de transmisie se poate realiza în două modalităţi. Prima modalitate constă în conectarea la capătul terminus a unui **circuit terminator** (**Thévenin**) format din rezistenţele R_1 şi R_2 ca în Figura 1.75-a. (**Teorema Thévenin**: O reţea liniară şi activă, cu două borne de ieşire A şi B şi fără cuplaje inductive cu exteriorul poate fi substituită cu un generator ideal de tensiune V_{Thev} înseriat cu o rezistenţă R_{Thev}: V_{Thev} este egală cu valoarea tensiunii la bornele reţelei la mers în gol $V_{Thev} = V_{AB_0}$, iar $R_{Thev} = V_{Thev}/I_{AB_{sc}}$. $I_{AB_{sc}}$ este curentul generat de reţea când bornele A şi B sunt scurtcircuitate.) Schema echivalentă Thevenin este desenată în Figura 1.75-b. Rezultă pentru tensiunea echivalentă $V_{Thev} = \left(\frac{V_{CC}}{R_1+R_2}\right) R_2$, iar $I_{sc} = V_{CC}/R_1$ deci $R_{Thev} = V_{Thev}/I_{sc} = (R_1 \cdot R_2)/(R_1 + R_2)$ adică cele două ramuri în paralel. În alegerea valorilor de rezistentă ale terminatorului se ţine seama de următoarele [Wakerly ′00]:

- Valoarea rezisteţei R_{Thev} trebuie să fie cât mai aproape de Z_0;
- Valoarea tensiunii V_{Thev} trebuie aleasă încât să optimizeze curentul absorbit şi generat de poarta care comandă linia de transmisie. Pentru porţile simetrice pe ieşire, care au valori egale pentru curentul absorbit şi generat (cazul porţilor CMOS dar nu şi cele CMOS compatibile TTL), se recomandă $V_{Thev} = (V_{OL} + V_{OH})/2$. Pentru porţile asimetrice pe ieşire la care $I_{OL} > I_{OH}$ (porţile TTL şi CMOS compatibile TTL), se recomandă $V_{Thev} > (V_{OL} + V_{OH})/2$, prin aceasta se ajută poarta (când ieşirea sa este în H) printr-o generare suplimentară de curent de către terminator pe linia comandată de poartă, dar cu costul creşterii curentului pe care trebuie să-l absoarbă poarta de pe linie în starea L;

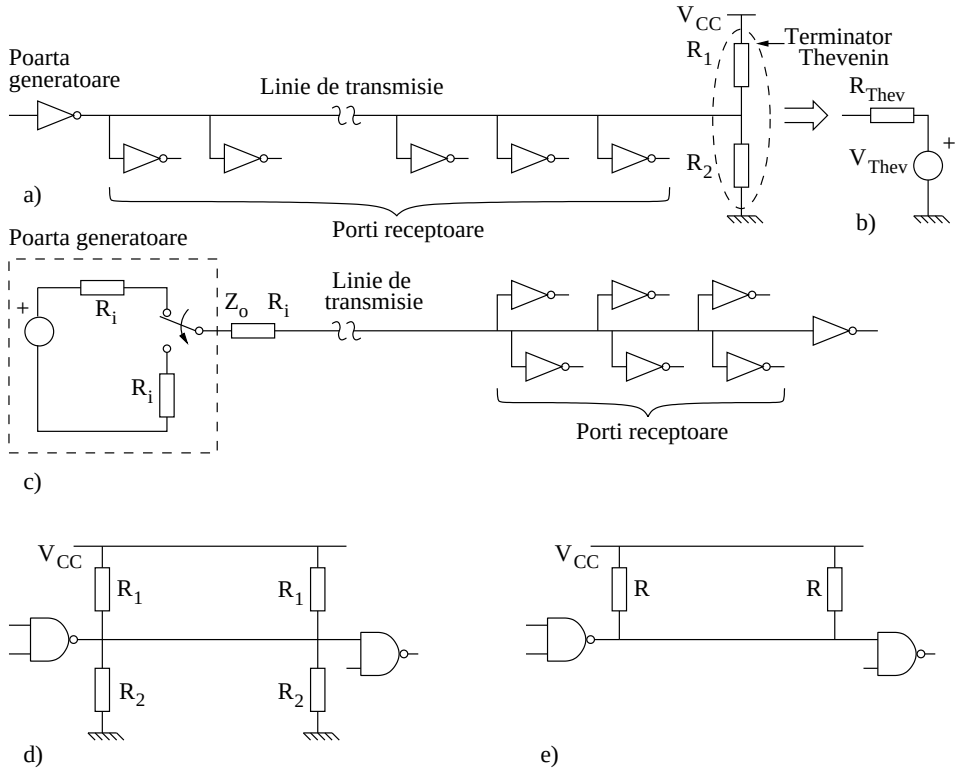


Figura 1.75 Modalităţi de adaptare a liniilor: a) adaptarea unei linii de magistrală prin conectarea la capăt a unui terminator Thevenin; b) schema echivalentă a terminatorului; c) adaptarea ieşirii unei porţi (la linia de magistrală) prin înserierea unei rezistenţe de valoare $Z_0 - R_i$; d,e) variante de adaptare a conexiunilor dintre două porţi.

- Când linia de transmisie este o linie de magistrală tip TSL va trebuie ca, atunci când magistrala nu este coamandată de către nici un emiţător, tensiunea V_{Thev} a terminatorului să fixeze un potenţial pe linie care să fie în intervalele de tensiuni de ieşire garantate (şi nu în intervalul de tensiuni interzise). Dacă tensiunea fixată pe linia de magistrală este aproape de tensiunea de pragul de comutaţie V_T (Definiţia 1.14) a porţilor receptoare, atunci pot să apară oscilaţii pe linie sau curenţii la intrările porţilor receptoare să crească cu mult peste valorile normale. (Mai recent, (porţile) buffere de magistrale au deja integrat pe ieşire un circuit activ de menţinere – bus holder, Figura 1.46-e – a nivelului pe linia de magistrală pentru intervalele când magistrala este în HZ.)

Valori uzuale pentru rezistenţele terminatorului sunt cele standard în plaja 150, 220, 270, 330, 390, 470Ω. O pereche, aproape standard, pentru aplicaţii TTL este $R_1 = 220\Omega$, $R_2 = 330\Omega$ pentru care se obţine $R_{Thev} = 132\Omega$ şi $V_{Thev} = 3V$, iar poarta care comandă linia trebuie să absoarbă în starea L un curent egal cu $(3V/132\Omega) =$

$22,7mA$ şi nu trebuie să genereze curent în starea H deoarece tensiunea H este menţinută de terminator. Dezavantajul principal al terminatoarelor Thévenin este consumul permanent de putere; uneori se utilizează ca terminator un circuit asimetric în care există numai R_1 iar $R_2 = \infty$.

A doua modalitate de eliminare a reflexiilor constă în înserierea unei rezistenţe, în linia de transmisie imediat în apropierea porţii care comandă linia, de valoare $R = Z_0 - R_i$, Figura 1.75-c. Pentru generatorul de tensiune al porţii, în schema echivalentă, apare o încărcare de $2Z_0$ (adică $R_i + Z_0 - R_i + Z_0 = 2Z_0$). Această modalitate este eficientă pentru porţile la care R_i rezistenţa de ieşire în stare H şi rezistenţa de ieşire în stare L sunt de valori apropiate (cazul porţilor CMOS), iar porţile comandate sunt toate grupate spre capătul terminus al liniei de transmisie. Pentru impedanţa caracteristică în intervalul $Z_0 = 50 \div 150\Omega$ se recomandă $R_i = 15 \div 40\Omega$.

Pentru porţile TTL(LS) rezistenţele de ieşire sunt: în jur de 30Ω în stare L şi în jur de 300Ω în stare H, iar cele de intrare în jur de 100Ω pentru $V_{in} \leq 1,5V$ şi $10K\Omega$ pentru $V_{in} > 1,5V$. Circuitele CMOS prezintă o impedanţă foarte mare pe intrare care pentru o linie de transmisiuni este echivalent cu o terminare în gol. Din dispersia acestor valori se poate constata că este foarte greu a se realiza o adaptare pentru toate conexiunile dintre două porţi.

La circuitele actuale (complexe) care funcţionează la frecvenţe ridicate (CPLD, FPGA, memorii, microprocesoare, vezi capitolul 4) există posibilitatea de a ajusta impedanţa de ieşire a unui driver care comandă o linie şi la fel impedanţa de intrare la un circuit receptor care recepţionează semnalul de la o linie. Această ajustare se face electronic, în funcţie de valoarea impedanţei caracteristice Z_0, se introduc sau se scot rezistenţe fixe încât rezistenţa echivalentă rezultată să coincidă cu Z_0. Fizic, introducerea sau scoaterea de rezistenţe se realizează prin înserierea cu fiecare rezistenţă fixă a unui tranzistor care se comandă respectiv în conducţie sau în blocare.

Se poate ca şi pentru conexiunile dintre porţi să se adopte modalităţile de adaptare utilizate pentru liniile de magistrală ca în Figura 1.75-d, 1.75-e, dar de cele mai multe ori se fac conexiuni între porţi fără a se face o astfel de adaptare. Pentru astfel de situaţii de neadaptare a conexiunilor dintre porţi se recomandă ca pe durata fronturilor de tranziţie ale semnalelor de comandă să se asigure ca linia de conexiune, de lungime l, să fie parcursă de cel puţin 5 ori. Se consideră că după cinci parcurgeri ($5T$) regimul tranzitoriu se stinge; şi în acest fel rezultă "crestat" variaţia de semnal doar în intervalul tranzitoriu, Figura 1.74, nu şi pe durata de regim static. Conform acestei recomandări, considerând $v_p = 20cm/ns$ pentru cinci parcurgeri pe durata frontului de cădere τ_{HL} (uzual $\tau_{HL} < \tau_{LH}$), rezultă lungimea maximă $l_{max} \leq (v_p \cdot \tau_{HL})/5$ a conexiunii neadaptate între două porţi, dar pentru care se asigură o funcţionare corectă. Aplicând această recomandare rezultă următoarele valori maxime pentru trasee: ECL, $\tau_{HL} \approx 2ns$, $l_{max} = 8cm$; TTL-S, $\tau_{HL} \approx 3 \div 4ns$, $l_{max} = 12 \div 16cm$; TTL, $\tau_{HL} = 5 \div 7ns$, $l_{max} = 20 \div 30cm$.

Se estimează că în viitor şi în interiorul unui sistem digital, similar ca în telecomunicaţii, între circuitele integrate chiar şi în interiorul circuitelor integrate, interconectarea va fi realizată prin trase pentru semnal optic. Avantajul fizic al semnalului optic faţă de cel electric este imunitatea la perturbaţii electromagnetice şi eliminarea constantelor de timp datorită încărcării capacitive. Estimările zic că, faţă de 2002, va fi curentă comunicaţia optică între cipuri în 5-10 ani, iar cea în interiorul cipului cam în jur de 15 ani.

1.6.2.3 Zgomotul datorat cuplajului electromagnetic (diafonia)

Diafonia (cross-talk), adică inducţia semnalului dintr-un traseu în altul vecin pe durata fronturilor, este o consecinţă a cuplajului inductiv şi capacitiv dintre trasee. Evident, cu cât frecvenţa semnalului printr-un traseu este mai ridicată cu atât tensiunea indusă în traseele vecine poate fi de valoare mai mare.

Definiţia 1.19 Nivelul de diafonie, D, este raportul dintre tensiunea (parazită) indusă într-un traseu (perturbat) şi tensiunea care o generează (perturbatoare).

$$D = \frac{V_{\text{perturbată}}}{V_{\text{perturbatoare}}}$$

◇

Se consideră, în Figura 1.76, două trasee unul compus din Poarta1 ce comandă, pe o linie cu impedanţa caracteristică Z_0, intrarea Porţii2 şi al doilea traseu compus din Poarta3 ce comandă Poarta4 pe o linie cu impedanţa caracteristică Z_0; între cele două trasee există o impedanţă de cuplaj Z_c. Cuplajul electromagnetic realizat prin impedanţa de cuplaj Z_c apare ca o rezultantă a cuplajelor formate prin capacităţile distribuite C_m şi cuplajului prin inductivităţile distribuite L. Pentru analiza cuplajului electromagnetic al celor două trasee se impune ca impedanţa de ieşire a porţilor de comandă Z_{out} să îndeplinească condiţia $Z_{out} \ll Z_0$ (poarta este un generator ideal de tensiune). Atunci, cu această condiţie, tensiunea indusă V_{I2} în linia 1-2, ce se aplică la Poarta2, de către tensiunea din linia 3-4, V_{O3}, generată la ieşirea porţii 3 va fi exprimată prin relaţia:

$$V_{I2} = \frac{V_{O3}}{1 + Z_c/Z_0} \rightarrow D = \frac{V_{I2}}{V_{O3}} = \frac{1}{1 + Z_c/Z_0} \tag{1.111}$$

Calitativ, relaţia 1.111, sugerează ce modalităţi pot duce la atenuarea diafoniei:

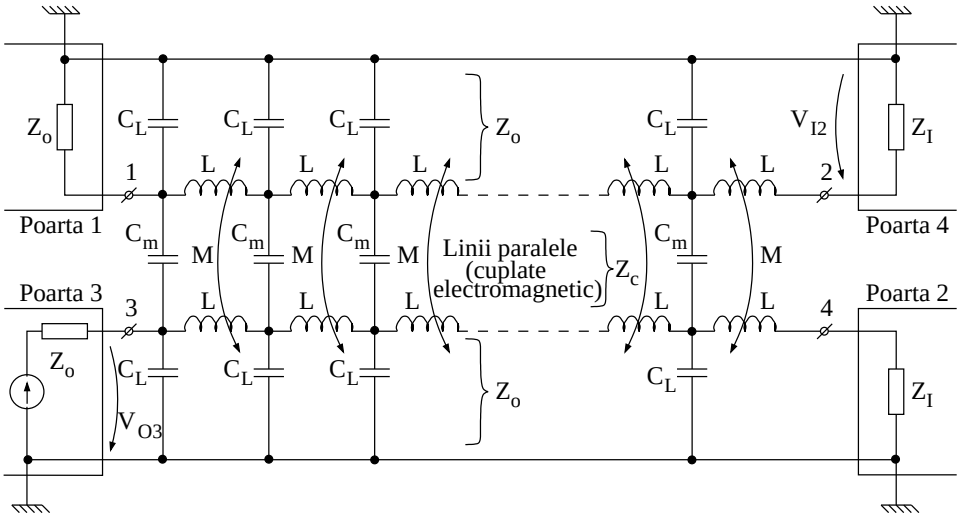


Figura 1.76 Explicativă pentru procesul de generare a diafoniei

- Mărirea impedanţei de cuplaj Z_c prin îndepărtarea traseelor (distanţa uzuală între traseele de circuit imprimat este de $0,2 \div 0,4mm$), sau micşorarea lungimii porţiunilor de paralelism între trasee sau ambele simultan;

- Micşorarea impedanţei caracterisitice Z_0 prin intercalarea unei trase de masă între trasele de circuit cuplate, sau utilizarea unui plan de masă;

- Utilizarea unor materiale care prezintă un coeficient de permeabilitate magnetică μ_r care se diminuează odată cu creşterea frecvenţei semnalului, deci se reduce cuplajul inductiv (polietilene utilizate pentru realizarea cablurilor plate);

- Micşorarea spectrului de frecvenţă al semnalelor prin reducerea pantei fronturilor de comutaţie. Aceasta se poate obţine prin conectarea la ieşirea porţilor a unor condensatoare de ordinul zeci $\div$ sute pF. Modalitatea aceasta este referită ca reducerea lui di/dt, care intervine în relaţia tensiunii induse $V = -L\, di/dt$. Creşterea frecvenţei semnalelor în sistemele digitale (la nivelul anului 2004 s-a ajuns la frecvenţa de ceas pentru microprocesoare în jur de 3GHz, perioada fiind de 333ps) crează dificultăţi în realizarea magistralelor; o magistrală de 64 de biţi are 64 de trasee care merg în paralel pe lungimi destul de mari. Pentru astfel de aplicaţii există circuite de comandă (drivere de magistrală) care transmit pe magistrală doar fronturile semnalelor digitale dar cu o pantă $di/dt \approx 1$; teoretic, un front al unui semnal digital (ideal) este $di/dt \to \pm\infty$.

Structura de driver de magistrală, din Figura 1.77-a, este de fapt un driver CMOS TSL, Figura 1.46-d, a cărei tensiune de ieşire la linia de magistrală, când este în stare HZ, este fixată la nivelul median $V_{DD}/2$ prin divizorul echilibrat format cu valori de impedanţă egale cu $2Z_0$. Un semnal digital $x(t)$ şi semnalul negat dar întârziat cu $3\tau_p$, $\overline{x}(t-3\tau_p)$, obţinut prin înserierea a trei inversoare, aplicate la intrarea unei porţi NAND va genera la ieşire un impuls negativ de lăţime $3\tau_p$, dar numai pe fronturile pozitive ale semnalului $x(t)$. Similar, aceleaşi semnale aplicate la intrarea unei porţi NOR va genera la ieşire un impuls pozitiv de lăţime $3\tau_p$ dar numai pe fronturile negative ale semnalului $x(t)$, Figura 1.77-b. Pe frontul pozitiv al semnalului de intrare $x(t)$ tranzistorul pMOS intră în conducţie iar tensiunea de ieşire v_O creşte liniar de la $V_{DD}/2$ la $V_{DD}/2+V_\Delta$ iar când tranzistorul se blochează scade liniar de la $V_{DD}/2+V_\Delta$ la $V_{DD}/2$. Similar, pe frontul negativ al semnalului $x(t)$ va conduce tranzistorul nMOS iar tensiunea de ieşire va avea variaţia liniară de la $V_{DD}/2$ la $V_{DD}/2-V_\Delta$, iar la blocarea tranzistorului, de la $V_{DD}/2-V_\Delta$ la $V_{DD}/2$. Semnalul $v_O(t)$ cu variaţii liniare cu panta în jur de unitate, aplicat pe o linie de magistrală, va induce în liniile vecine tensiuni de valoare mult mai mică decât un semnal cu fronturi foarte abrupte. La recepţie un circuit trebuie să sesizeze sensul frontului semnalului $x(t)$ şi să refacă amplitudinea şi durata acestui semnal, adică o deplasare de la $V_{DD}/2$ fie la V_{DD} fie la V_{SS}.

Exemplul 1.31 a) Dacă liniile de transmisiune sunt realizate din conductori de cupru cu $\phi = 1mm$, aşezaţi la o distanţă mai mică de $1mm$ unul faţă de celălalt şi la o distantă mai mare de $20mm$ faţă de orice conductor de masă impedanţele vor avea valorile $Z_0 = 200\Omega$, $Z_c = 80\Omega$ rezultă un nivel de diafonie $D_1 = 0,71$. Acest raport zgomot semnal este inacceptabil întrucât nici un circuit standard nu are o margine de zgomot mai mare decât o treime din valoarea saltului logic de tensiune.

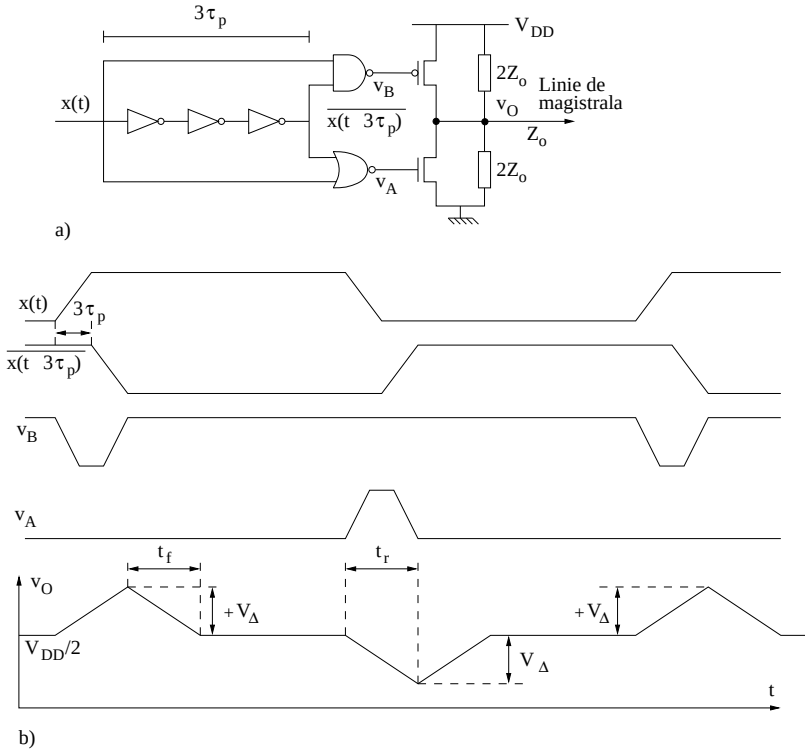


Figura 1.77 Driver de magistrală cu di/dt redus: a) structură circuit; b) explicativă prin diagrame de semnal pentru obţinerea pe linia de magistrală a unui semnal numai cu panta egală cu ± 1.

b) Dacă în exemplul a) se introduce un plan de masă la distanţa de $1mm$ de fiecare conductor se obţin valorile $Z_5 0\Omega$, $Z_c = 125\Omega$ iar nivelul de diafonie este $D_2 = 0,28$. Valoarea 0.28 pentru diafonie este destul de mare, se apropie prea mult de valoarea maximă de 30 % a marginii de zgomot în curent continuu de la porţile CMOS.
c) Dacă atât linia emiţătoare cât şi cea receptoare se realizează din cablul torsodat impedan-ţele au valorile $Z_0 = 80\Omega$, $Z_c = 400\Omega$ şi diafonia este $D_3 = 0,16$. Acest raport zgomot/semnal este satisfăcător pentru toate circuitele TTL şi TTLS.

1.6.2.4 Zgomotul datorită curenţilor de alimentare

Curentul absorbit de o poartă de la sursa de alimentare, în timp, nu are o valoare constantă. În oscilograma formei de variaţie a curentului de alimentare apar evident vârfuri cu amplitudinea destul de mare (spikes, glitches) în momentele de comutaţie $H - L$ şi $L - H$ ale porţii, Figura 1.78. Aceste vârfuri de curent din perioadele tranzitorii sunt cauze generatoare de zgomot în două modalităţi: 1- prin inducţie pot produce tensiuni parazite în circuitele vecine; 2- micşorează tensiunea de alimentare a porţii cu valoarea căderii de tensiune provocată pe impedanţa traseului de alimentare

(mai ales pe componenta inductivă a acestui traseu). La valoarea totală a curentului absorbit de o poartă contribuie trei componente una de regim staţionar şi două de regim tranzitoriu.

1. Curenţii absorbiţi în regim staţionar (de curent continuu). Valorile curenţilor printr-o poartă în starea H, I_{CCH}, şi în starea L, I_{CCL}, nu sunt egali, deci pe circuitul de alimentare, între cele două stări, poarta provoacă variaţiile de curent $\Delta I = I_{CCL} - I_{CCH}$ care prin căderile de tensiune de pe inductivităţile parazite L ale traseului şi ale pinilor circuitului micşorează tensiunea de alimentare V_{CC} cu valoarea ΔV:

$$\Delta V = L \cdot \frac{\Delta I}{\Delta T}\,[V] \tag{1.112}$$

Remediu pentru aceste variaţii de tensiune este un condensator de descărcare C_d conectat chiar pe terminalele V_{CC}/V_{DD} şi masă ale porţii, condensator care constituie un "rezervor" de energie şi care va livra pentru poartă necesarul de curent ΔI în momentele când tensiunea de alimentare scade. Impunând o anumită cădere de tensiune ΔV, când este necesar un curent suplimentar ΔI într-un interval de timp ΔT, valoarea minimă a condensatorului C_d se determină cu relaţia:

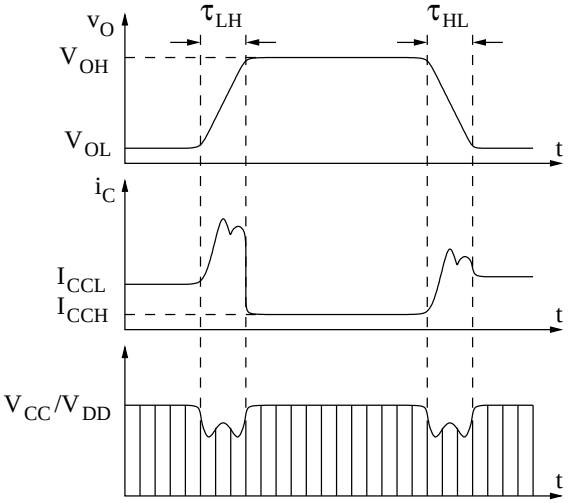


Figura 1.78 Vârfurile de curent de alimentare la tranziţiile $H-L$ şi $H-L$ produc variaţii în tensiunea de alimentare

$$C_d = \frac{\Delta I}{\Delta V/\Delta T}\,[F] \tag{1.113}$$

De exemplu, pentru circuitul TTL-LS 7400 (patru porţi NAND) consumul total al celor patru porţi când sunt alimentate la $V_{CC} = 5V$ este de $I_{CCL} = 2,4mA$ şi $I_{CCH} = 0,8mA$. Dacă presupunem că toate porţile sunt comandate simultan la aceeaşi frecvenţa de $1MHz$, pentru ca tensiunea de alimentare a circuitului să nu varieze cu mai mult de 5% ($\Delta V = 5\%V_{CC} = 0,25V$), este necesară o capacitate de descărcare $C_d = (2,4mA - 0,8mA)/(0,25V/0,5 \cdot 1\mu s) = 3,2nF$.

2. Curenţii de încărcare şi descărcare a capacităţilor în momentele de comutaţie. Aceşti curenţi determină peste 90% din puterea disipată pe o poartă CMOS, dar sunt destul de reduşi pentru tehnologia bipolară. Curentul de încărcare a sarcinii, la comutarea $L-H$, şi curentul de descărcare a sarcinii, la comutarea $H-L$ sunt egali numai când $W_p = 2W_n$ (pentru $\mu_n = 2\mu_p$). De exemplu, pentru circuitul 74HCT00 (patru porţi NAND) curentul de încărcare ΔI_{LH} şi curentul de descărcare ΔI_{HL} al circuitului, când porţile sunt comandate simultan pentru un salt logic de $3,6V$ cu o viteză de creştere de $0,25V/ns$ şi de $0,4V/ns$ pentru descreştere, se pot determina (cu o aproximaţie destul de bună pentru aplicaţiile practice) cu relaţia 1.112

$$\Delta I_{LH} = 4 \times 50pF \times 0,25 \cdot 10^9 V/s = 5mA$$

$$\Delta I_{HL} = 4 \times 50pF \times 0,4 \cdot 10^9 V/s = 8mA$$

iar timpii de creştere Δt_{LH} şi descreştere Δt_{HL} rezultă

$$\Delta t_{LH} = 3,6V/(0,25V/ns) = 14,4ns$$

$$\Delta t_{LH} = 3,6V/(0,4V/ns) = 9ns$$

3. Curenţii de scurtcircuit. Aceşti curenţi care apar pe traseele dintre V_{DD}/V_{CC} şi masă în momentele de comutaţie, atât în tehnologia CMOS cât şi bipolară, prin durata lor scurtă pot constitui surse de zgomot.

Variaţiile de curent datorită regimului tranzitoriu, punctul 2 şi 3, pot ajunge la valori mai mari decât $10mA/ns$ care pe o inductivite de $0,1\mu H$(valori tipice pentru traseele de circuit imprimat sunt $0,01 \div 0,02\mu H/cm$), conform relaţiei 1.112 vor produce căderi de tensiune $\Delta V = 0,1\mu H \times 10mA/ns = 1V$. Efectul acestor variaţii de curent este acelaşi ca şi cel produs de al variaţiilor curentului de alimentare cu deosebirea că de data aceasta spectrul de frecvenţe este mult mai înalt.

Atenuarea zgomotelor generate de curenţii de alimentare se poate realiza prin conectarea pe pinii de alimentare al fiecărui circuit integrat a unui condensator ceramic de ordinul $10 \div 100nF$ (pentru rejecţia frecvenţelor înalte) şi a unui condensator de decuplare cu tantal de $0,33\mu F$ (pentru frecvenţe joase) la câte un grup de 3–4 circuite integrate. De fapt, în aceeaşi manieră, dar de valori mai mari, se face decuplarea circuitului de alimentare la intrarea pe placa de circuit imprimat. Totuşi, numai decuplarea de la intrarea plăcii de circuit imprimat nu este suficientă deoarece inductivităţile parazite ale traseelor, de la intrare până la circuit, împiedică livrarea rapidă de curent spre circuite şi atunci această decuplare locală se repetă şi în apropierea circuitului.

Exemplul 1.32 Să se calculeze căderea de tensiune produsă pe o inductivitate $L = 2nH$ (de exemplu inductivitatea firului care conectează zona de pad la pinul circuitului) de către curentul de descărcare al sarcinii capacitive $C_L = 100pF$ de la ieşirea unui buffer CMOS conectat la pad. Se consideră sarcina încărcată la $V_{DD} = 5V$ iar $\tau_{\text{HL}} = 5ns$.

Soluţie. În Figura 1.79 este trasată variaţia reală a curentului de descărcare cu linie continuă, iar cu linie întreruptă se estimează o variaţie liniară. Pentru acest caz relaţia 1.113 se scrie sub forma

$$I_{d_{max}} \cdot \frac{t_d}{2} = C_L \cdot V_{DD}$$

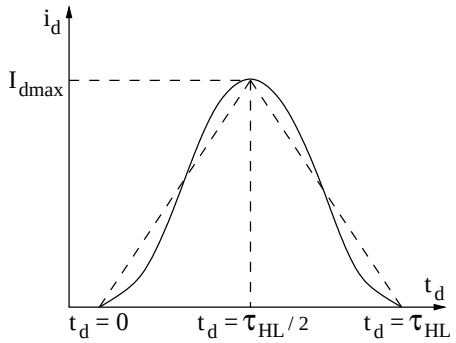


Figura 1.79 Explicativă pentru Exemplul 1.32

şi de asemenea, din estimarea de variaţie liniară pentru curentul i_d, se poate scrie relaţia de variaţie a curentului

$$\left(\frac{di_d}{dt_d}\right)_{max} \geq \frac{I_{d_{max}}}{\tau_{\rm HL}/2} = \frac{2I_{d_{max}}}{\tau_{\rm HL}}$$

rezultă relaţia

$$\left(\frac{di_d}{dt_d}\right)_{max} \geq \frac{4C_L V_{DD}}{\tau_{\rm HL}^2}$$

Introducând valorile numerice se obţine

$$\left(\frac{di_d}{dt_d}\right)_{max} \geq \frac{4 \times 100 \times 10^{-12} \times 5}{(5 \times 10^{-9})^2} = 80mA/ns$$

$$\Delta V = L\left(\frac{di_d}{dt_d}\right)_{max} \geq 160mV$$

Dacă poarta devine mai rapidă de două ori $\tau_{\rm HL} = 2,5ns$ căderea de tensiune se măreşte de patru ori $4 \times 160mV = 0,64V$.

PROBLEME

P1.1 Utilizând axiomele şi teoremele algebrei Booleene, să se demonstreze analitic următoarele identităţi şi apoi să se deducă tabelul de adevăr al expresiei respective:
a) $B + \overline{A}C = (A + B + C)(\overline{A} + B + C)(\overline{A} + B + \overline{C})$;
b) $\overline{A}D + \overline{C}D + A\overline{B} = \overline{A}\,\overline{C}D + A\overline{B}\,\overline{C} + A\overline{B}C + \overline{A}CD$;
c) $D(\overline{A} + +C + \overline{D})(A + B + \overline{C} + \overline{D}) = (D + A\overline{C} + \overline{A}C)(\overline{A}\,\overline{C} + BD + AC)$.

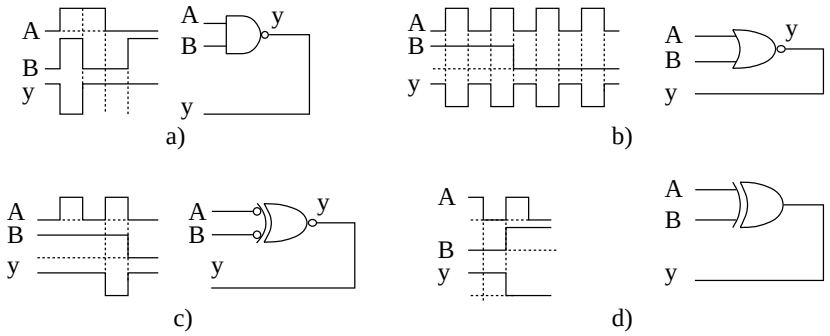
P1.2 Să se demonstreze următoarele identităţi şi apoi să se deducă tabelul de adevăr al expresiei respective:
a) $AB + (A + B)C = AB + (A \oplus B)C$;
b) $A \oplus B = B \oplus A = \overline{A} \oplus \overline{B}$
c) $A \oplus 1 = \overline{A}$ (inversorul comandat realizat cu o poartă XOR); $A \oplus 0 = A$; $A \oplus \overline{A} = 1$; $A \oplus A = 0$;
d) $\overline{A \oplus B} = AB + \overline{A}\,\overline{B}$;
e) $A \oplus B = \overline{A \oplus \overline{B}} = \overline{\overline{A} \oplus B}$;
f) $ABC + \overline{A}\,\overline{B} + \overline{AB\overline{C}}D = ABC + \overline{A}\,\overline{B} + D$;
h) $A\overline{B}C(BD + CDE) + A\overline{C} = A(\overline{C} + \overline{B}DE)$;
g) $\overline{A}BC + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}C + ABC = BC + A\overline{B} + \overline{B}\,\overline{C}$.

P1.3 Să dezvolte următoarele expresii (utilizând teoremele lui De Morgan):
a) $\overline{A\overline{B}(C + \overline{D})}$;
b) $\overline{AB(CD + EF)}$;
c) $\overline{(A + \overline{B} + C + \overline{D})} + \overline{ABC\overline{D}}$;

d) $\overline{\overline{(\overline{A}+B+C+D)}\ \overline{(A\overline{B}\ \overline{C}D)}}$;
e) $\overline{\overline{AB}(CD+\overline{E}F)(\overline{AB}+\overline{CD})}$
f) $\overline{\overline{(ABC)}\ \overline{(EFG)}+\overline{(HIJ)}\ \overline{(KLM)}}$;
g) $\overline{(A+\overline{B\overline{C}}+CD)+\overline{\overline{BC}}}$
h) $\overline{\overline{(A+B)}\ \overline{(C+D)}\ \overline{(E+F)}\ \overline{(G+H)}}$.

P1.4 Folosind porţile setului complet XOR, AND să se implementeze operatorii: NOT, AND, OR, NAND, NOR, XNOR. Se vor utiliza simbolurile ANSI/IEEE.

P1.5 Care porţi din Figura următoare (a, b, c, d) nu funcţionează corect? Oscilogramele semnalelor de pe intrări şi de pe ieşiri sunt prezentate in aceeaşi figură.



P1.6 Să se implementeze operatorul sumă modulo doi (XOR) pentru două variabile numai cu porţi $NAND$ sau numai cu porţi NOR cu două intrări.

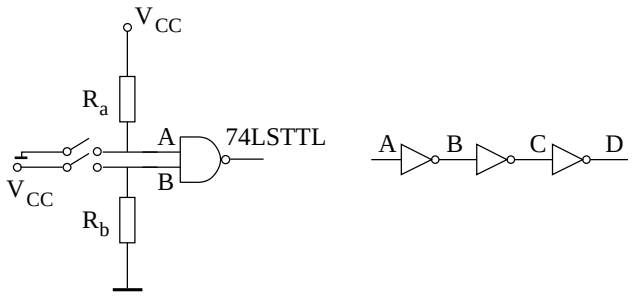
P1.7 Să se construiască tabelele de adevăr şi reprezentările simbolice pentru implementarea operatorului sumă logică de două variabile (OR2) considerând toate variantele de activare ale semnalelor de intrare şi de ieşire.

P1.8 Să se explice cum se procedează cu intrările neutilizate ale unei porţi logice astfel încât acestea să nu ducă la o funcţionare incorectă a porţii.

P1.9 Pentru porţile TTL, impedanţele de ieşire sunt în jur de 30 Ω în starea L şi în jur de 300Ω în starea H. Impedanţa caracteristică a traseelor de circuit pe placa de sticlotextolit are valori cuprinse între $Z_0 = 50 \div 150\Omega$. În scopul de a se evita reflexiile pe liniile de conectare între porţi,pentru realizarea condiţiei $Z_0 = Z_r$ (impedanţa pe care se realizează reflexia), se măreşte artificial impedanţa de ieşire a porţii care comandă linia prin inserarea, la ieşirea acesteia, a unei rezistenţe R. Să se determine valoarea maxima a rezistenţei R.

P1.10 Să se comande o diodă electroluminiscentă (LED) cu o poartă 74HC MOS şi 74LS TTL. Parametrii de catalog ai acestor porţi sunt daţi în tabelul de la P1.12. Punctul de funcţionare al LED-ului în starea de luminiscenţa are coordonatele $I_D = 8mA$, $U_D = 1,6V$.

P1.11 Să se calculeze valorile rezistenţelor R_a şi R_b ale circuitului din figura (a) astfel ca, atunci când ambele comutatoare sunt deschise, pe intrările A şi B să fie asigurate nivelele logice “0” şi “1”. Care este puterea disipată de aceste rezistenţe când comutatoarele sunt deschise şi închise? Valorile de catalog ale parametrilor porţii sunt date în tabelul de la P1.12.



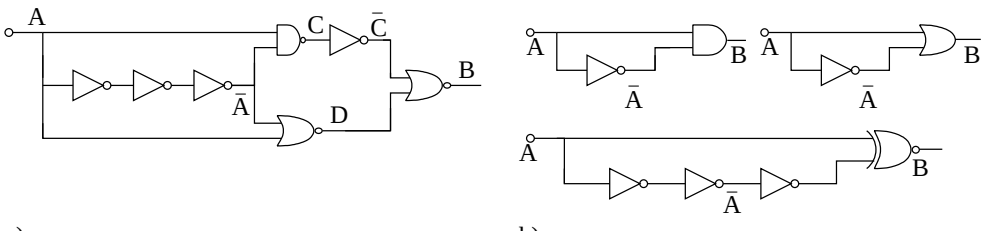
a) b)

P1.12 Să se determine dacă o poartă 74HCMOS poate comanda patru porţi 74LS TTL şi dacă o poartă 74LS TTL poate comanda patru porţi 74HC MOS. Valorile tipice de catalog ale parametrilor porţilor logice sunt date în tabelul următor:

Tip	$V_{IH(min)}$	$V_{IL(max)}$	$V_{OH(min)}$	$V_{OL(max)}$	$I_{IH(max)}$	$I_{IL(max)}$	$I_{OH(max)}$	$I_{OL(max)}$
74HCMOS	*3,5 V*	*1 V*	*4,9 V*	*0,1 V*	1 µA	–1 µA	40 µA	4 mA
74LSTTL	*2 V*	*0,8 V*	*2,7 V*	*0,4 V*	20 µA	–400 µA	–400 µA	8 mA

P1.13 Pentru porţile inversor ale circuitului b) (desenat la P1.11) timpii de propagare sunt: $\tau_{LH} = 6ns(min)/10ns(max)$, $\tau_{HL} = 4ns(min)/6ns(max)$, $\tau_r = \tau_f = 1ns$. Presupunând valorile minime pentru timpii de propagare,să se determine timpul total de propagare prin circuit pentru comanda intrării: $0 \to 1, 1 \to 0$. Considerând că fiecare inversor are timpul de propagare situat oriunde între valoarea minimă şi cea maximă, să se schiţeze formele de undă în punctele B,C şi D când semnalul pe intrarea A are tranziţiile: $0 \to 1, 1 \to 0$.

P1.14 Pentru circuitul din figura de mai jos (a) se consideră, pentru toate porţile, $\tau_{PHL} = \tau_{PLH} = \tau_p$. Să se determine expresiile logice pentru variabilele notate în figură. Discuţie. Ţinând cont de timpii de propagare, să se descrie prin forme de undă valorile variabilelor logice pe durata regimurilor tranzitorii. Discuţie.

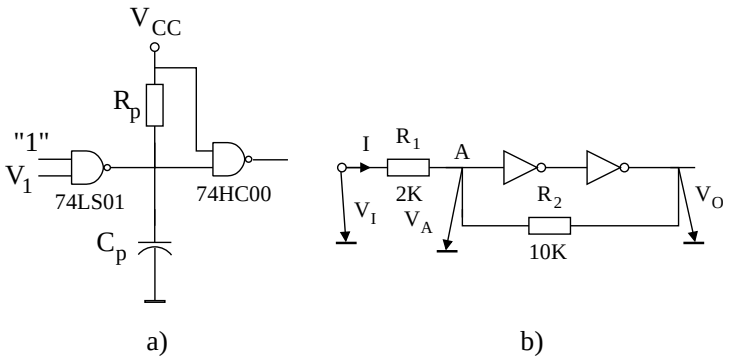


a) b)

P1.15 Pentru circuitele din figura b) (de la P1.14), să se determine formele de undă pentru variabila B când variabila A are o variaţie sub forma de semnal dreptunghiular de perioadă $T = 10\tau_p$ cu coeficientul de umplere 50%. Se consideră $\tau_p = \tau_{PHL} = \tau_{PLH}$, pentru toate porţile circuitelor. Să se interpreteze formele de undă pentru variabila

B, similar cu interpretarea de la problema 1.13.

P1.16 Pentru comanda porţii NAND 74HC00 de la ieşirea unei porţi NAND cu colectorul în gol, 74LS01, se utilizează interfaţarea din figura (a) de mai jos. Să se calculeze valoarea rezistenţei de pull-up R_p astfel încât să se obţină:
a) un consum minim de putere pe R_p;
b) un timp de tranziţie minim la intrarea porţii CMOS;



a) b)

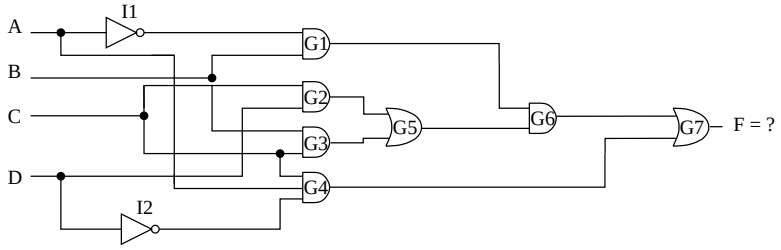
P1.17 Să se determine caracteristica de transfer $V_0 = f(V_1)$ pentru circuitul din figura b) (de la P1.16) de mai sus realizat cu porţi inversoare CMOS, având: $V_T = 2,5V$ (tensiune de prag de comutaţie), $V_{OHmin} = 4,9V, V_{OLmax} = 0,1V$. Să se deseneze formele de undă la ieşire când V_I are o variaţie triunghiulară cu $V_{Imax} = 4V$.

P1.18 Să se deseneze structura reţelei de porţi care realizează următoarea funcţie logică:

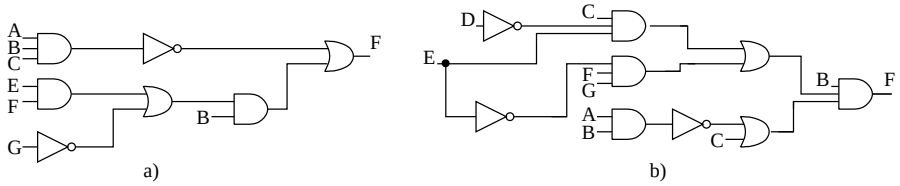
$$F = (ABC + D)\overline{EF} + GH(\overline{I + J} + K)$$

P1.19 Să se deseneze structura reţelelor care realizează următoarele funcţii logice:
a) $F = (\overline{AB} + C)[\overline{(D + E)}F + \overline{G}]$;
b) $F = (A\overline{B} + C)\overline{A}B + \overline{B}C$;
c) $F = \overline{A}B(\overline{C}\,\overline{D} + \overline{C}D) + AB(\overline{C}\,\overline{D} + \overline{C}D) + A\overline{B}\,\overline{C}\,D$;
d) $F = (\overline{A}\,\overline{B} + A\overline{B})(C\,D + C\overline{D})$;
e) $F = AB(C + DEF) + CE(A + B + F)$.

P1.20 Pentru reţeaua de comutaţie din figură să se deducă funcţia logică F pe care o realizează.

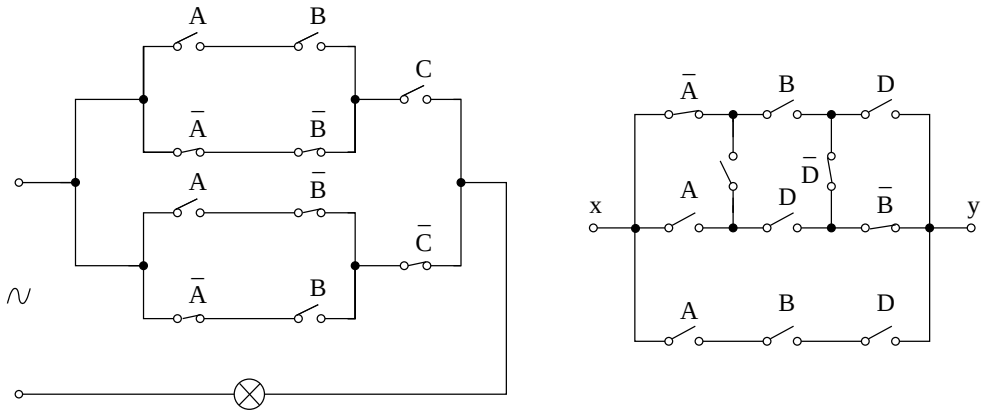


P1.21 Pentru reţelele de comutaţie din figurile următoare să se deducă funcţiile logice realizate.

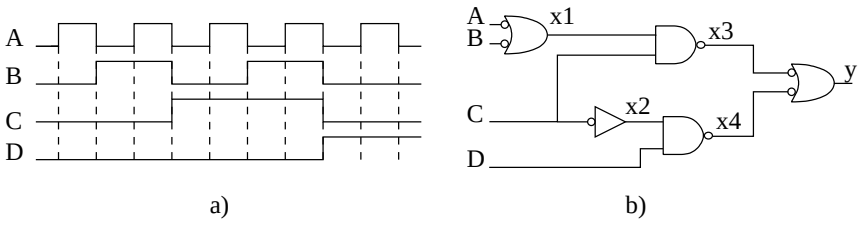


P1.22 Structura circuitului de comandă realizat cu contacte pentru aprinderea şi stingerea unui bec din oricare din cele trei puncte A,B şi C este reprezentată în figura (a)de la P1.23. Implementaţi acelaşi circuit de comandă cu porţi logice.

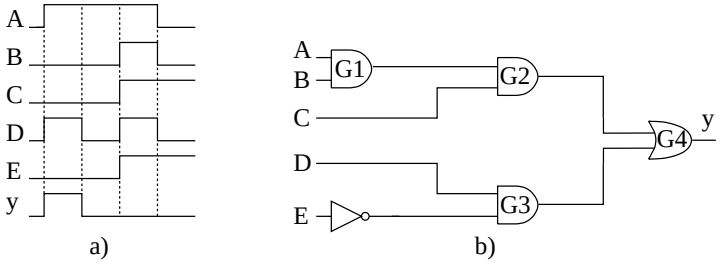
P1.23 Să se determine funcţia de comutaţie F pentru structura de reţea din figura de mai jos (b). Să se simplifice expresia funcţiei F şi să se realizeze o implementare cu porţi logice.



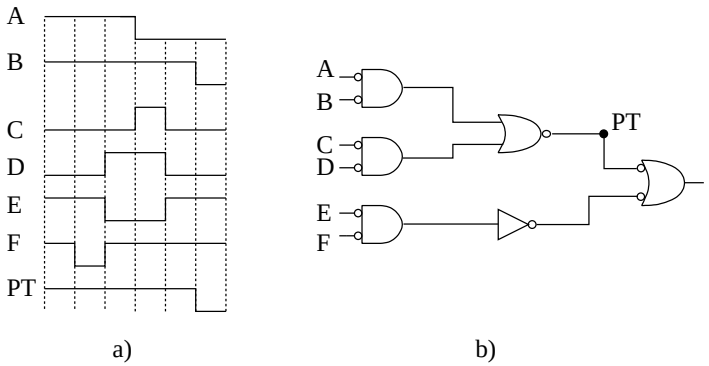
P1.24 Circuitului din figura (b) alăturată i se aplică semnalele de intrare A,B,C şi D având formele de undă în figura (a). Să se determine formele de undă în punctele x_1, x_2, x_3, x_4 şi y. Apoi, pentru acest circuit combinaţional să se deducă expresia logică a ieşirii y şi, cu ajutorul acesteia, să se deducă forma de undă a semnalului y pentru variaţia intrărilor.



P1.25 Pentru circuitul logic combinaţional din figura (b), la aplicarea formelor de undă la intrare, desenate alăturat în figura (a), se obţine forma de undă y la ieşire. Această ieşire y este incorectă datorită unei porţi defecte din structura circuitului. (O poartă defectă are ieşirea fie permanent în starea H fie permanent în starea L, indiferent de valoarea logică a intrărilor). Să se localizeze poarta defectă şi să se determine defectul acesteia (ieşire permanent H sau permanent L).



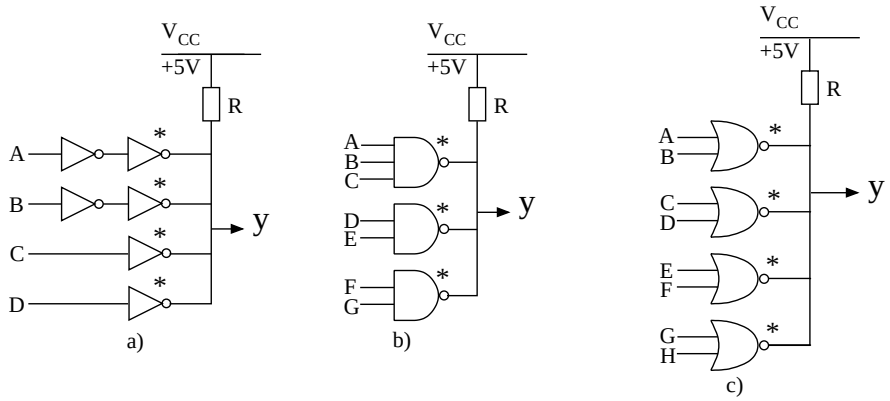
P1.26 Pentru circuitul din figura (b) sunt redate alăturat formele de undă pe intrări. Sunt accesibile pentru oscilografiere numai ieşirea şi punctul de test PT. Este corectă forma de undă PT? Dacă nu, care este defectul?



P1.27 Pentru circuitele din figurile următoare a),b) şi c) să se determine expresia logică realizată pe ieşirea y. Considerând că toate bufferele open colector pe ieşire au $I_{OLmax} = 40mA$, $V_{OLmax} = 0,25V$, $I_{OHmax} = 400\mu A$ să se determine valoarea rezistenţelor R dacă funcţia y reprezintă o încărcare de 10 intrări 74LS00 (vezi tabelul de la P1.12). Se admite pentru $\Delta V_{CC} = 0,1V_{CC}$ iar $M_H = M_L = 0,4V$.

P1.28 Folosind datele de catalog, date în tabelul de la problema P1.12, pentru poarta 74HC00 (NAND2) să se determine dacă poate comanda pe ieşire următoarele sarcini rezistive:

a) 120Ω conectată la V_{DD};
b) 270Ω conectată la V_{DD} şi 330Ω conectată la masă;
c) $1K\Omega$ conectată la masă;
d) 150Ω conectată la V_{DD} şi 150Ω conectată la masă;
e) 100Ω conectată la V_{DD};
f) 75Ω conectată la V_{DD} şi 150Ω conectată la masă;



g) 75Ω conectată la V_{DD} ;
h) 270Ω conectată la V_{DD} şi 150Ω conectată la masă.

P1.29 Pentru poarta 74HC00, utilizând datele din tabelul de la problema P1.12, să se estimeze rezistenţa de ieşire în stare H şi în stare L.

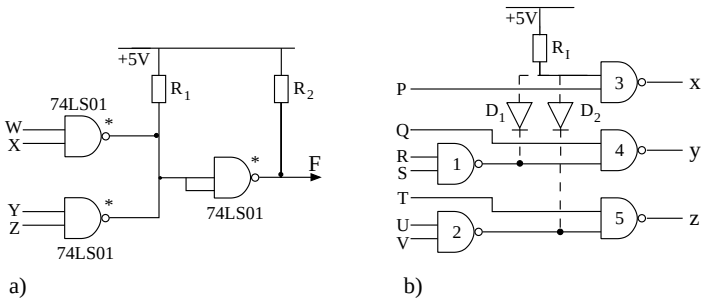
P1.30 Pentru decizia "pentru ieşirea unei porţi cu colectorul în gol sau cu drenul în gol se alege o rezistenţă de valoare mai mare sau mai mica" aduceţi argumente pro şi contra.

P1.31 Un buffer cu drenul în gol, $V_{OLmax} = 0,37V$, $I_{OLmax} = 12mA$, comandă un LED de semnalizare pentru care se fixează punctul de funcţionare $V_{LED} = 1,6V$, $I_{LED} = 10mA$. Să se determine valoarea rezistenţei înseriate cu LED-ul când $V_{DD} = 5V$.

P1.32 Care rezistor disipă mai multă putere cel care conectează o intrare neutilizată la V_{CC} a unei porţi NAND TTL-LS sau cel care conectează o intrare neutilizată la masă a unei porţi NOR TTL-LS? (Utilizaţi datele din tabelul de la problema P1.12)

P1.33 Ce se întâmplă dacă se încearcă să se comande, direct fără rezistenţă adiţională, un releu alimentat la +12V printr-o poartă normală TTL?

P1.34 Circuitul din figura (a) utilizează porţi 74LS01 cu colectorul în gol pentru care datele de catalog corespund cu cele din tabelul de la P1.12 de la seria LS cu diferenţa că I_{OHmax} este $100\mu A$.



a) Să se deducă analitic şi să se verifice prin metoda tabelului de adevăr expresia funcţiei F.
b) Considerând $M_H = 0,7$ care este valoarea maximă admisă pentru R_1?
c) Dacă semnalul F comandă două inversoare 74S04 (datele corespund seriei S din Tabelul 1.9) să se determine valoarea maximă şi minimă pentru R_2 când $M_H = 0,7V, M_L = 0V$.

P1.35 La o linie de magistrală sunt conectate n module.Fiecare modul este compus dintr-o poartă receptoare 74LS04 (parametrii acestui inversor sunt în Tabelul 1.9 la seria LS) şi un buffer, TSL 74LS125, emiţător pe magistrală. Bufferul TSL în starea HZ absoarbe sau generează un curent $\pm 20\mu A$ iar în starea normală cu ieşirea L absoarbe un curent $I_{OLmax} = 24mA$ şi cu ieşirea în H generează un curent $I_{OHmax} = 2,6mA$. Câte module se pot conecta la magistrală?

P1.36 Se descoperă o deficienţă la circuitul din figura (b) de la P1.34. Proiectantul remediază această deficienţă prin introducerea diodelor D1 şi D2 pe traseele desenate punctat. Descrieţi modificarea logică şi a marginii de zgomot, prin această introducere, asupra circuitului. Toate porţile sunt 74LS00 (NAND2).

P1.37 Să se deseneze structura circuitelor CMOS care realizează funcţiile $F_1 = A + BC$, $F_2 = A(B + C)$.

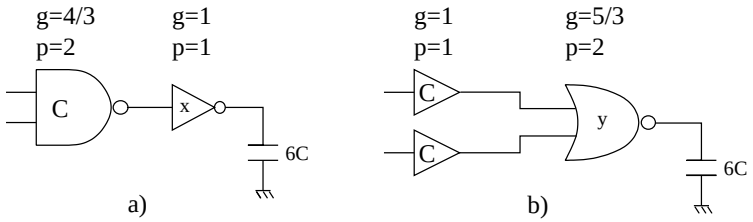
P1.38 Să se deseneze structura circuitului CMOS care implementează funcţia

$$F = \overline{(A_1 + A_2 + A_3)(B_1 + B_2)C}.$$

P1.39 Pentru funcţia logică $F = \overline{A + B + CD}$:
a) Să se deseneze structura de circuit CMOS;
b) Să se deducă drumul eulerian şi apoi să se deseneze layoutul simplificat.

P1.40 Pentru circuitul de coincidenţă:
a) Să se deseneze structura de circuit CMOS;
b) Să se deducă drumul eulerian şi apoi să se deseneze layoutul simplificat.

P1.41 Se consideră circuitele din figura (a)şi (b) care au un efort electric H=6.
a) Care este efortul total pentru fiecare circuit?
b) Care dintre ele este mai rapid?
c) Să se calculeze dimensiunile x şi y de poartă astfel încât pentru circuitul respectiv să se obţină întârzierea cea mai mică.



P1.42 Un traseu logic este proiectat pe trei etaje; pe fiecare dintre ele este repartizat un efort $F_1 = 10, F_2 = 9, F_3 = 7$.
a) Poate fi această proiectare optimizată? Dacă da în ce mod?
b) La prezenta proiectare ce îmbunătăţiri se pot aduce?

P1.43 Se consideră un traseu logic pe opt niveluri/etaje, pe fiecare nivel efortul electric este egal cu $h_i = 3$. Cea mai complexă poartă care poate fi într-un nivel, din

acest traseu, este NAND4. Să se estimeze care este intervalul de timp cel mai mic după care semnalul logic, aplicat la intrarea traseului, poate fi modificat.

P1.44 La structura de NAND8 din Figura 2.29-c adăugaţi, după ultimul inversor, încă două inversoare. Pentru cele trei structuri din Figura 2.29 şi cea obţinută în acest mod trasaţi, pe acelaşi grafic, dependenţa, întârzierea D funcţie de variaţia efortului electric H în intervalul $H = 12 \div 200$. Care este concluzia din analiza acestor dependenţe?

P1.45 Care dintre porţile CMOS, NANDn, NORn, pentru acelaşi efort electric, este mai rapidă? Argumentaţi afirmaţia.

P1.46 O linie de magistrală cu impedanţa caracteristică $Z_0 = 100\Omega$ este comandată de ieşirea unui buffer din L $(0, 2V)$ în H $(2, 7V)$ pe durata $\Delta t = 3ns$. Să se determine valoarea capacităţii de decuplare C_B conectată între linia de alimentare V_{CC} şi masă astfel încât acest salt să nu provoace o variaţie maximă a tensiunii de alimentare $\Delta V_{CC} > 0, 1V$.

P1.47 Un driver TSL, caracterizat prin $I_{OLmax} = 24mA, V_{OLmax} = 0, 4V$ comandă o linie de magistrală cu impedanţa caracteristica $Z_0 = 150\Omega$. Receptorii de la linia de magistrală sunt porţi de tip trigger Schmitt cu pragurile de comutaţie $V_{p^-} = 0, 9V$, $V_{p^+} = 1, 7V$ iar curenţii absorbiţi atât în starea L cât şi în starea H sunt neglijabili. Tensiunea V_{CC} poate varia în limitele $\pm 10\%$.

a) Să se dimensioneze rezistenţele pentru terminatorul Thevenin al liniei;

b) Să se calculeze marginile de zgomot garantate M_H şi M_L.

P1.48 Pentru o poartă logică TTL trigger Schmitt neinversor cu pragurile de basculare în intervalele $V_{p^-} = (0, 6 - 0, 9)V$, $V_{p^+} = (1, 7 - 2)V$ să se calculeze marginile de zgomot M_H şi M_L.

P1.49 Pentru capătul unei linii de magistrală, cu impedanţa caracteristică Z_0, să se dimensioneze un terminator Thevenin care să nu producă reflexii ale semnalului. Pentru perioadele când linia nu este comandată de nici un driver potenţialul liniei să fie stabilit de terminator fie la $V_{OH} = 3, 4V$, fie la $V_{OL} = 0, 25V$ $(V_{CC} = 5V)$.

Capitolul 2

CIRCUITE LOGICE COMBINAŢIONALE

Multitudinea funcţiilor logice de n variabile, 2^{2^n}, ar atrage dupa sine, dacă nu se face o selectare, realizarea a tot atâtea circuite. Selectarea funcţiilor candidat pentru implementare, sub formă de circuit, se face în funcţie de eficienţa şi frecvenţa în aplicaţii dar şi după realizabilitatea circuitului. De exemplu, după cum s-a vazut în capitolul anterior, chiar şi pentru două variabile, $n = 2$, din cele 16 funcţii posibile sunt implementate doar porţile uzuale (AND, OR, NAND, NOR şi XOR). În această abordare şi pentru funcţiile mai complexe, de natură logică sau aritmetică, de mai puţine sau de mai multe intrări, doar unele dintre ele au corespondentul fizic, sub formă de circuit; şi astfel de circuite sunt, de facto standard şi utilizate ca şi componente în implementarea altor funcţii/sisteme. Prezentarea, sinteza şi implementarea unora dintre acesta circuite constituie continutul acestui capitol de circuite numite combinaţionale. Dar de ce combinaţionale? Pentru că valoarea funcţiei, existentă la ieşire doar atâta timp cât există anumite valori pentru intrări, depinde exclusiv de combinaţia valorilor de intrare, altfel spus, de configuraţia valorilor de intrare.

2.1 CIRCUITUL LOGIC COMBINAŢIONAL

Un sistem este caracterizat prin natura semnalelor de intrare, a celor de ieşire, prin clasele de funcţii intrare-ieşire (transfer) şi prin natura prelucrărilor ce au loc in structura sa interna. Această caracterizare generală a unui sistem particularizată pentru un **C**ircuit **L**ogic **C**ombinational, **CLC**, poate fi exprimată formal prin tripletul:

$$CLC = (X, Y, F) \tag{2.1}$$

în care:

X – reprezintă mulţimea de configuraţii binare aplicate pe intrare sau mulţimea cuvintelor de intrare $X = \{X_0, X_1, ..., X_k, ..., X_{2^n-1}\}$. Fiecare cuvant de intrare X_k este un element al mulţimii $\{0, 1\}^n$ (vezi Definiţia 1.3) şi este de forma:

$X_k = x_{n-1}x_{n-2}...x_i...x_1x_0$, $i = 0, 1, ..., (n-1)$; x_i este valoarea binară a variabilei (semnalului) aplicată pe intrarea a i-a a circuitului combinaţional, Figura 2.1-a. Mulţimea cuvintelor de intrare X este complet definită dacă cuprinde toate configuraţiile binare formate cu cele n variabile de intrare, adică 2^n configuraţii; cardinalul mulţimii este $|X| = 2^n$. De exemplu, pentru un CLC cu patru intrări mulţimea X este complet definită dacă pe intrările x_3, x_2, x_1, x_0 nu este restricţionată aplicarea nici uneia din cele 16 combinaţii posibile de intrare, $\{0000, 0001, ..., 1110, 1111\}$.

Y – reprezintă mulţimea de configuraţii binare obţinute la ieşire sau mulţimea cuvintelor de ieşire $Y = \{Y_1, Y_2, ..., Y_l, ..., Y_q\}$. Y_l este un element al mulţimii $\{0,1\}^m$ şi este de forma $Y_l = y_{m-1}y_{m-2}...y_j...y_0$, $j = 0, 1, 2, ...(m-1)$; y_j este valoarea (semnalul) binară obţinută pe ieşirea j a circuitului combinaţional. Mulţimea cuvintelor de iesire Y este, in general, incomplet definită, deoarece la ieşire nu se genereaza toate cuvintele binare de m biţi, adică $q \leq 2^m$. De exemplu, pentru un CLC cu n intrări şi 3 ieşiri, mulţimea Y a cuvintelor de ieşire poate să nu fie complet definită deoarece cuvântul de ieşire, $y_2y_1y_0$, nu va lua toate cele 2^3 combinaţii posibile $\{000, 001, ..., 1111\}$ atunci când cuvântul de intrare va parcurge toate cele 2^n configuraţii de intrare.

F – este funcţia de transfer (intrare-ieşire) care, pentru un CLC cu n intrari si m iesiri (Definitiei 1.4), defineşte aplicaţia $\{0,1\}^n \rightarrow \{0,1\}^m$. Funcţia de transfer a unui CLC cu n intrări asociază fiecărei configuraţii binare de intrare $X = \{0,1\}^n$ un cuvânt Y din mulţimea configuraţiilor de ieşire $Y \subseteq \{0,1\}^m$, adică selectează perechi din submulţimea produsului cartezian $\{0,1\}^n \times \{0,1\}^m$. Conform celor enunţate, pentru CLC, o reprezentare sub forma de schemă bloc este cea din Figura 2.1.

Circuitul Logic Combinaţional cu o singură ieşire, $m = 1$, realizează aplicaţia $f : \{0,1\}^n \rightarrow \{0,1\}$, lungimea cuvântului generat la ieşire are lungimea de 1 bit (Figura 2.1-b). În general, vom studia circuie logice combinaţionale cu o singură ieşire, cele cu ieşiri multiple ($m \neq 1$) pot fi privite ca fiind compuse din m circuite cu o singură ieşire. Numarul de curcuite combinaţionale distincte, de n intrari şi cu o singura ieşire, este egal cu numărul tuturor funcţiilor logice de n variabile, adică 2^{2^n} (vezi secţiunea 1.1.3).

Definirea unui CLC cu n intrari si m ieşiri, ca un set de perechi intrare-ieşire din produsul cartezian $\{0,1\}^n \times \{0,1\}^m$, poate fi realizată printr-o funcţie de transfer care are o exprimare /definiţie simplă sau complexă. Atributul de complex sau de simplu pentru un circuit decurge din modalitatea complexă sau simplă prin care se descrie structurarea sau funcţionarea sa.

Exemplul 2.1 Pentru următoarele trei seturi de perechi intrare-ieşire să se exprime funcţiile de transfer

$$\begin{aligned} CLC_1 = &\ \{(00,0),(01,1),(10,1),(11,0)\} \\ CLC_2 = &\ \{(000,0),(001,1),(010,1),(011,0), \\ &\ (100,1),(101,0),(110,0),(111,1)\} \\ CLC_3 = &\ \{(00,010),(01,110),(10,010),(11,001)\} \end{aligned}$$

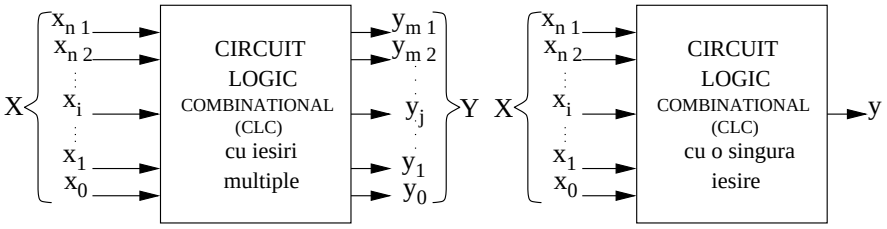
Soluţie. Primul circuit CLC_1, cu 2 intrări, $n = 2$, şi o ieşire, $m = 1$, realizează funcţia SAU EXCLUSIV de două variabile $y = x_1 \oplus x_0$.

Al doilea circuit CLC_2, cu trei intrări, $n = 3$, şi o ieşire, $m = 1$, realizează functia SAU EXCLUSIV de trei variabile $y = x_2 \oplus x_1 \oplus x_0$.

Al treilea circuit CLC_3, cu două intrări, $n = 2$, şi trei ieşiri, $m = 3$ (y_2, y_1, y_0), nu admite o exprimare simplă, printr-o funcţie logică elementară, ca cele două anterioare. Pentru aceasta putem să avem următoarea exprimare:

$$y_2 = 1 \; numai \; daca \; (x_1 = 0) \quad SI \quad (x_0 = 1)$$
$$y_1 = 1 \; numai \; daca \; (x_1 = 0) \; SAU \; (x_0 = 0)$$
$$y_0 = 1 \; numai \; daca \; (x_1 = 1) \quad SI \quad (x_0 = 1)$$

Din acest exemplu putem spune că CLC_1 este un circuit simplu deoarece definirea sa a necesitat o descriere simplă (funcţia sumă modulo 2, $x_1 \oplus x_0$). CLC_2 cu toate că are trei intrări poate fi exprimat, la fel, printr-o definiţie compactă: suma modulo 2 de trei variabile, care rezultă prin recurenţă din sumă modulo 2 de două variabile $x_2 \oplus (x_1 \oplus x_0)$. Se poate concluziona că un CLC, chiar cu un număr mare de intrări, rămâne tot un circuit simplu dacă poare fi exprimat, pornind de la un circuit simplu, extins prin recurenţă pentru un n mare ceea ce în cazul analizat ar fi $x_n \oplus (x_{n-1} \oplus (... \oplus (x_2 \oplus (x_1 \oplus x_0))...))$. Cel de-al treile circuit, CLC_3, la care definiţia nu a putut fi compactată nu mai este un circuit simplu ci un circuit complex.



Nr	Intrari					Iesiri					
	$x_{n\,1}$	$x_{n\,2}$	... x_i ...	x_0	x_1	y_0	y_1	... y_j ...	$y_{m\,1}$	...	$y_{2^n\,1}$
0	0	0	... 0 ...	0	0	d_{00}	d_{10}	... d_{j0} ...	$d_{(m\,1)0}$	...	
1	0	0	... 0 ...	0	1	d_{01}	d_{11}	... d_{j1} ...	$d_{(m\,1)1}$	...	
2	0	0	... 0 ...	1	0	d_{02}	d_{12}	... d_{j2} ...	$d_{(m\,1)2}$	...	
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		
$2^n\,2$	1	1	... 1 ...	1	0	$d_{02^n\,2}$	$d_{12^n\,2}$	... $d_{j2^n\,2}$ ...	$d_{(m\,1)(2^n\,2)}$	...	
$2^n\,1$	1	1	... 1 ...	1	1	$d_{02^n\,1}$	$d_{12^n\,1}$	... $d_{j2^n\,1}$ ...	$d_{(m\,1)(2^n\,1)}$	...	

m functii

c)

Figura 2.1 Simbol de reprezentare pentru CLC cu n intrări: a) cu ieşiri multiple; b) cu o singură ieşire; c) tabel de adevăr pentru un CLC cu ieşiri multiple (m).

Definiţia 2.1 Complexitatea unui circuit cu n intrări, notată cu $C(n)$,

este o mărime asociată dimensiunii definiţiei/descrierii acelui circuit. ⋄

Limitarea asimptotică a creşterii unei funcţii f, de către o alta funcţie g, poate fi exprimată prin notaţia O **(citită** *de ordinul***)** in felul următor: $f(n) \in O(g(n))$ [Greenlaw ′98].

Definiţia 2.2 Funcţia f este de ordinul lui g, notat $f(n) \in O(g(n))$, dacă şi numai dacă există o constantă $c > 0$ şi un $n_0 \in \mathbf{N}$ astfel incât $f(n) \leq c \cdot g(n)$ pentru toate valorile lui $n \geq n_0$. ⋄

De exemplu, pentru funcţia polinomială $f = 5 \cdot n^4 + 17 \cdot n - 10$, când n devine foarte mare, este limitată superior de $c \cdot n^4$ deci, se poate spune ca f este de ordinul $O(n^4)$, adică $(5 \cdot n^4 + 17 \cdot n - 10) \in O(n^4)$. Aplicând notaţia de limitare superioară pentru funcţia complexitate C(n), a unui CLC cu n intrări, se poate spune că circuitul este simplu când $C(n) \in O(1)$, adică dimensiunea definiţiei pentru n de valoare mare este limitată de o constantă $O(1)$. În schimb, nu se mai poate afirma la fel când $C(n) \in O(n)$, dacă n este de ordinul 10^6 când circuitul este greu, dacă nu imposibil, de exprimat, deci circuitul este complex. Evident, pentru implementări, se vor selecta circuite simple şi nu circuite complexe care implică efort si costuri foarte ridicate sau chiar depăşesc posibilitaţile tehnologice.

Observaţie importantă! În relaţia 2.1 de definire a circuitului logic combinaţional nu este implicat timpul; aceasta înseamna că la un CLC transferul configuraţiei de intrare X în obţinerea celei de ieşire Y se face instantaneu. Acest transfer poate fi doar teoretic instatntaneu, pe când la un sistem real timpul de transfer este egal cu timpul de propagare intrare-ieşire prin lanţul de porţi ce compun CLC. Neincluderea timpului nu are nici o consecinţa in regim static, formalismul algebrei Booleene poate determina corect semnalul de ieşire în funcţie de semnalele de intrare. Dar, uneori, în intervalele tranzitorii, când configuraţia de intrare se modifică, formalismul booleean aplicat pe lanţul porţilor de la intrare spre ieşire va calcula valori de ieşire care diferă de cele reale obţinute la ieşire; valorile de ieşire reale nu mai corespund regimului (static) de aplicare a axiomei de existenţă a complementului $x \cdot \bar{x} = 0$ şi $x + \bar{x} = 1$ ci ar corespunde situaţiei anormale de $x \cdot \bar{x} = 1$ şi $x + \bar{x} = 0$! Aceste situaţii, care pot apărea pe durata regimurilor tranzitorii, sunt referite cu termenul de hazard static şi vor fi studiate in secţiunea 2.3.1. Neincluderea variabilei timp poate fi considerată ca o lacună a formalismului booleean; exstă încercari de a elabora un formalism logic care să includă şi variabila timp.

Al doilea aspect care trebuie notat in relaţia 2.1 este faptul că nu există o reacţie, adică transferul este unidirecţional, de la intrare spre ieşire, mărimile de ieşire nu modifică in nici un fel intrarea în sistem. Clasa de circuite secvenţiale, capitolul 3, va include şi aceste două aspecte adică: timpul şi existenţa reacţiei.

2.2 REPREZENTAREA CIRCUITELOR LOGICE COMBINAŢIONALE

Reprezentarea/descrierea unui CLC este un instrument absolut necesar pentru procesele de: proiectare (sinteză), de testare şi de documentare. Modalitatea de descriere, prin complexitatea pe care o implică, trebuie aleasă adecvat pentru realizarea

eficientă a acestor procese. Se pot distinge următoarele modalitaţi de reprezentare: tabelul de adevăr, funcţia analitică, diagrama de decizie binară, exprimare prin limbaj natural.

2.2.1 Tabelul de adevăr

Noţiunea de tabel de adevăr a fost introdusă in secţiunea 1.14 prin Definiţia 1.12 iar exemple de tabele de adevăr au fost expuse in Figura 1.1 si 1.2 şi Tabelele 1.3 şi 1.6. În continuare se vor prezenta aspecte practice în realizarea şi utilizarea tabelelor de adevăr. În general, în procesul de sinteză al uni CLC se porneşte de la realizarea tabelului de adevăr pe baza cerinţelor de funcţionare impuse sistemului respectiv. Astfel, se consideră toate configuraţiile valorilor variabilelor de intrare (în număr de 2^n), care vor constitui liniile de intrare ale tabelului. Apoi, pentru fiecare configuraţie de intrare, $i \quad (i = 0, 1, ..., (2^{n-1}))$, se va înscrie în coloana de ieşire y_j a tabelului valoarea corespunzătoare a funcţiei, adică valoarea coeficientului funcţiei d_{ji}, Figura 2.1-c. Coficientul d_{ji} este egal cu valoarea funcţiei y_j, $j = 0, 1, ..., (m-1)$, pentru configuraţia a i-a a variabilelor de intrare. Tabelul de adevăr din figură corespunde unui CLC cu n intrări şi m ieşiri.

Mulţimea cuvintelor de ieşire Y s-a specificat anterior, în general, este o mulţime incompletă deoarece numărul de cuvinte distincte de m biţi obţinut la ieşire (un cuvânt fiind format din biţii de pe cele m coloane ale ieşirii care corespund la o configuraţie de intrare) este mai mic decât numărul total de cuvinte formate cu m biţi, adică 2^m. Fiecare coloană din tabelul de adevăr conţine un număr de 2^n valori binare, ceea ce înseamnă că ar putea fi un număr de 2^{2^n} coloane cu valori distincte, deci un număr de 2^{2^n} funcţii de n variabile.

Succesiunea configuraţiilor de valori binare aplicate pe intrarea unui CLC se consideră că sunt exprimate în cod binar natural, adică chiar numărarea în sistemul de numeraţie in baza doi. Fiecare bit din cuvântul de cod binar natural are ponderea 2^i unde i este poziţia bitului în cuvântul binar, ponderile cresc de la dreapta spre stânga, prima poziţie din dreapta are ponderea zero (2^0) ($6|_{10} = 110|_2$; $6 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$). Codul binar natural este un cod aritmetic (adică, poate fi utilizat in operaţiile aritmetice).

Definiţia 2.3 Distanţa Humming între două cuvinte de cod, de aceeaşi lungime, este dată de numarul biţilor diferiţi care apar între aceleaşi poziţii ale celor două cuvinte de cod. $\diamond$

Uneori succesiunea configuraţiilor de valori binare ale variabilelor de intrare sunt considerate in cod Gray. **Codul Gray** are proprietatea de adiacenţă, adică distanţa Humming (**distanţa de cod**) intre oricare două cuvinte consecutive este egală cu 1. La trecerea de la un cuvânt la următorul, fie in sens crescător, fie în sens descrescător, se modifică în cuvântul de cod doar un singur bit (000,001,011,010,110,111,101,100). Acest cod nu este aritmetic! Datorită proprietăţii de adiacenţă codificarea Gray este utilă în notarea la diagramele Veitch-Karnaugh şi în implementarea circuitelor care impun ca în funcţionare, la trecerea dintre două cuvinte (stări) consecutive, să se producă cu o singură comutare.

Conversia din cod binar natural în cod Gray se realizează în felul următor, Figura 2.2-a:

1. cel mai semnificativ bit al cuvântului de cod Gray este identic cu cel mai semnificativ bit din cuvântul de cod binar natural;

2. parcurgând cuvântul de cod binar natural, de la stânga la dreapta, prin sumarea fiecărei perechi de biţi adiacenţi se obţine bitul următor (în sensul de la stânga la dreapta) din cuvântul de cod Gray (bitul de transfer rezultat în urma sumării se neglijează).

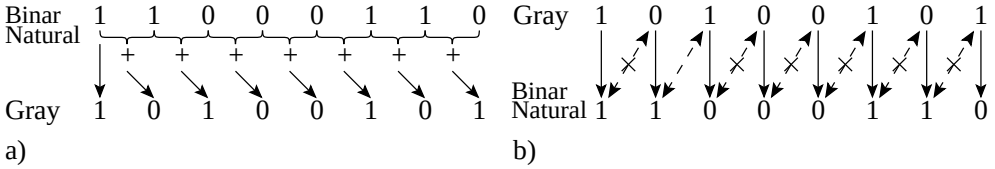


Figura 2.2 Modalitatea grafică de descriere a conversiei: a) din cod binar natural în cod Gray ; b) din cod Gray în cod binar natural.

Conversia din cod Gray in cod binar natural se realizează in felul următor, Figura 2.2-b:

1. cel mai semnficatv bit din cuvântul de cod binar natural este identic cu cel din cuvântul Gray;

2. parcurgând cuvântul in cod Gray, de la stânga la dreapta, se obţine bitul cuvântului binar natural de indice i (bitul cel mai puţin semnificativ având indicele $i = 0$) prin sumarea bitului $(i + 1)$ din cuvântul binar natural cu bitul de indice i din cuvântul Gray (bitul de transfer rezultat în urma sumării se neglijează).

Foarte utilizate sunt codificările zecimal binare, **BCD** (**B**inary **C**oded **D**ecimal), pentru cifrele zecimale (0,1,2...,8,9). Deoarece sunt 10 cifre zecimale de codificat cuvântul binar de cod trebuie sa aibă o lungime de patru biţi, deci din cele 16 cuvinte binare de patru biţi sunt alese doar 10,iar 6 dintre aceste nu sunt utilizate. Se pune întrebarea câte posibilitaţi distincte de codificare (coduri zecimal-binare), BCD, se pot realiza? Numărul grupelor diferite de câte 10 cuvinte de 4 biţi, din totalul de 16 cuvinte de 4 biţi, care se pot forma se calculează cu formula combinarilor $C_{16}{}^{10} = \frac{16!}{10!\cdot(16-10)!}$. Apoi, vor fi 10! modalitaţi prin care cele 10 cifre zecimale sunt asignate grupelor de câte 10 cuvinte de 4 biţi, deci în total numarul de coduri zecimal codificat binar este $C_{16}^{10} \cdot 10! = 29.059.430.400$.

Evident, din acest numar mare de coduri BCD, se utilizează foarte puţine şi fiecare dintre acestea are un nume. Varianta de cod prin care cifrelor zecimale (0,1,2,3,4,5,6,7,8,9) li se asignează cuvintele binare chiar în ordinea corespunzătoare de la numărarea in binar natural (0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001) este denumită uzual BCD (incorect! când de fapt toate variantele de codificare zecimal-binar sunt coduri BCD). Corect, eventual, ar trebui să fie denumirea acestui cod **NBCD** (**N**atural **B**inary **C**oded **D**ecimal), dar deoarece abreviaţia de BCD a intrat în exprimarea uzuală, cu regret o vom accepta în continuare. De

asemenea, chiar când se referă la primele zece cuvinte din binar natural se face uneori confuzia spunând coduri BCD (datorită identităţii cuvintelor).

Un alt cod, zecimal codificat binar, este cel denumit EXCESS3 care se obţine din cuvintele de cod BCD la care se adauga cifra 3 exprimată in binar natural, adică 0011; de exemplu, pentru cifra zecimală 2 codul binar obţint este $0010+0011=0101$, iar pentru cifra 5 codul este $0101+0011=1000$.

Avantajele utilizării reprezentarii în EXCESS3 sunt:

- într-o locaţie de memorie se poate face distincţia dacă in acea locaţie nu s-a înscris nimic (ceea ce corespunde stării utilizate 0000), sau s-a inscris cifra 0 (adica $0000+0011=0011$)
- complementul unei cifre zecimale faţa de 9 se obţine in EXCESS3 prin complementarea bit cu bit a cuvântului de cod ceea ce duce la o simplitate in implementare. De exemplu, complementul lui 7 ($0111+0011=1010$) faţă de 9 este 2 (0101) care se obţine prin complementarea lui 1010 (=7) faţă de, 1 adică 0101.

Cu aceste noţiuni sumare despre codurile zecimal-codificat-binar putem sa construim tabelele de adevăr pentru 2 circuite combinaţionale: un convertor de cod BCD - Gray şi un convertor de cod BCD - EXCESS3. Fiecare dintre aceste convertoare este un CLC cu 4 intrări A, B, C, D şi cu 4 ieşiri: E_3, E_2, E_1, E_0 pentru convertorul BCD - EXCESS3, Figura 2.3-b şi G_3, G_2, G_1, G_0 pentru convertorul BCD - Gray, Figura 2.3-c.

Tabelele de adevăr pentru aceste două converotaare, Figura 2.3-a, au pe intrări numai primele 10 combinaţii din NBCD, restul de 6 combinaţii (1010, 1011, 1100, 1101, 1110, 1111) nu se aplică niciodată deoarece nu aparţin codului BCD. In coloanele celor 6 funcţii (E_3, E_2, E_1, E_0 şi G_3, G_2, G_1, G_0) se înscriu valorile coeficienţilor obţinuţi conform regulilor de conversie ale celor două circuite codificatoare. Pentru cele 6 configuraţii de intrare excluse se consideră valorile coeficienţilor funcţiilor (ieşirile) ca fiind indiferente (notate prin —), adică se poate asigna fie valoarea 0 fie valoarea 1 deoarece oricum configuraţiile respective nu apar niciodată pe intrare.

Valori indiferente pentru coeficienţii unei funcţii pot fi trecute in tabelul de adevăr în două situaţii: fie când anumite configuraţii de valori ale variabilelor de intrare nu se aplică niciodată la intrarile CLC, fie când se aplică configuraţii pe intrări dar utilizarea ieşirilor corespunzătoare nu este semnificativă pentru ansamblul din care face parte circuitul.

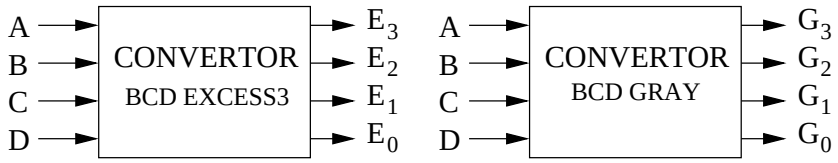
Când numărul de variabile de intrare ale unui CLC creşte peste 4, tabelul de adevăr se manipuleaza destul de greu din cauza numărului liniilor componente din tabel care creşte exponenţial cu numarul variabilelor. Dar, există o modalitate prin care se poate micşora numărul liniilor dintr-un tabel, astfel aducându-l la un instrument mai uşor de manipulat. Pentru circuitul CLC cu tabelul de adevăr din Figura 2.4-a, conform relaţiei 1.12, se poate scrie forma normală disjunctivă a funcţiei f (sinteză pe bază de 1)

$$f = \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + ABC\bar{D} \qquad (2.2)$$

Pentru combinaţia de intrare $A\bar{B}\bar{C}\bar{D}$ când funcţia nu este definită s-a considerat pentru coeficientul funcţiei valoarea 1, deci acest termen canonic a fost introdus in relaţia 2.2.

	A	B	C	D	E_3	E_2	E_1	E_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	1	1	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	0	1
2	0	0	1	0	0	1	0	1	0	0	1	1
3	0	0	1	1	0	1	0	1	0	0	1	0
4	0	1	0	0	0	1	1	1	0	1	1	0
5	0	1	0	1	1	0	0	0	0	1	1	1
6	0	1	1	0	1	0	0	1	0	1	0	1
7	0	1	1	1	1	0	1	0	0	1	0	0
8	1	0	0	0	1	0	1	1	1	1	0	0
9	1	0	0	1	1	1	0	0	1	1	0	1
linii nedefinite ale funcţiilor												
10	1	0	1	0	—	—	—	—	—	—	—	—
11	1	0	1	1	—	—	—	—	—	—	—	—
12	1	1	0	0	—	—	—	—	—	—	—	—
13	1	1	0	1	—	—	—	—	—	—	—	—
14	1	1	1	0	—	—	—	—	—	—	—	—
15	1	1	1	1	—	—	—	—	—	—	—	—

a)



b) c)

Figura 2.3 Conversia BCD—Gray şi BCD—EXCESS3: a) tabelele de adevăr; b,c) reprezentarea sub formă de schemă bloc a convertoarelor.

Coloana de intrare a variabilei D poate fi eliminată din tabelul de adevăr prin introducerea acestei variabile in expresiile coeficienţilor funcţiilor. Astfel se exprimă acesti coeficienţi prin intermediul variabilei D. Această modalitate este referită ca exprimare prin **coeficienţii cu variabile reziduu**. In acest caz, D este **variabila reziduu**. Valorile coeficienţilor cu variabila D reziduu se pot determina prin următorul raţionament efectuat pe coloanele D si f din tabelul de adevăr

- la configuraţia $ABC = 000$ pentru D, fie 1, fie 0, coeficienţii funcţiei au valoarea 0, funcţia este independentă de D, deci $f = 0$.

- la configuraţia $ABC = 001$ pentru D, fie 1, fie 0, coeficienţii funcţiei au valoarea 0, funcţia este independentă de D, deci $f = 0$.

- la configuraţia $ABC = 101$ pentru $D = 0$ şi $D = 1$, coeficienţii funcţiei au valoarile respectiv 0 si 1, deci $f = D$.

A	B	C	D	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	—
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

a)

$\Longrightarrow$

A	B	C	f
0	0	0	0
0	0	1	0
0	1	0	D
0	1	1	1
1	0	0	$\overline{D}$
1	0	1	$\overline{D}$
1	1	0	0
1	1	1	$\overline{D}$

b)

$\Longrightarrow$

A	B	f
0	0	0
0	1	C+D
1	0	$\overline{D}$
1	1	$C\overline{D}$

c)

Figura 2.4 Pentru tabelul de adevăr cu 16 configuraţii de intrare (a), prin introducerea variabilei reziduu D, se obţine un tabel cu 8 linii (b) şi prin introducerea variabilelor reziduu C,D se obţine un tabel cu 4 linii (c).

- la configuraţia $ABC = 011$ pentru D, fie 1, fie 0, coeficienţii funcţiei au valoarea 1, funcţia este independentă de D, deci $f = 1$.
- la configuraţia $ABC = 100$, pentru $D = 0$ şi $D = 1$, coeficienţii funcţiei au valorile respectiv 1 şi 0, deci $f = \bar{D}$ (s-a considerat valoarea 1 pentru semnul de indiferent).
- la configuraţia $ABC = 101$, pentru $D = 0$ şi $D = 1$, coeficienţii funcţiei au valorile respectiv 1 şi 0, deci $f = \bar{D}$.
- la configuraţia $ABC = 110$ pentru D fie 0, fie 1, coeficeinţiii funcţiei au valoarea 0, funcţia este independentă de D, deci f=0.
- la configuraţia $ABC = 111$, pentru $D = 0$ şi $D = 1$, coeficienţii funcţiei au valorile respectiv 1 şi 0, deci $f = \bar{D}$.

Tabelul de adevăr obţinut doar cu 3 variabile A, B, C, cu variabila D reziduu introdusă in coeficienţii funcţiei, este reprezentat în Figura 2.4-b. Expresia normal disjunctivă a funcţiei este:

$$f = (D) \cdot \bar{A}B\bar{C} + (1) \cdot \bar{A}BC + (\bar{D}) \cdot A\bar{B}\bar{C} + (\bar{D}) \cdot A\bar{B}C + (\bar{D}) \cdot ABC$$

care devine identică cu cea din relaţia 2.2 dacă al doilea termen produs $\bar{A}BC \cdot 1$ se expandează $\bar{A}BC(D+\bar{D}) = \bar{A}BCD + \bar{A}BC\bar{D}$. Coeficienţii reziduu (care conţin varaibila reziduu), pentru evidenţiere, sunt încadraţi în paranteză în exprimarea funcţiei.

Se poate continua procesul de reducere a numărului de linii de la 8 la 4 prin efectuarea unui raţionament similar, cu cel pentru reducerea de la 16 la 8 linii, asupra coloanei variabilei C şi a coloanei coeficienţilor funcţiei obţinându-se tabelul de adevăr din Figura 2.4-c. De data aceasta, coeficienţii funcţiei sunt exprimaţi în funcţie de 2 variabile reziduu C şi D.

Expresia disjunctivă a funcţiei este:

$$f = (C + D) \cdot \bar{A}B + (\bar{D}) \cdot AB \cdot +(C\bar{D}) \cdot AB$$

Dacă în aceasta expresie se introduc în termenii produs de 3 variabile şi variabila care lipseşte, prin relaţiile $1 = C + \overline{C}$, $1 = D + \overline{D}$ se obţine forma din relaţia 2.2.

Exemplul 2.2 Să se descrie sub formă de tabel de adevăr funcţionarea unui CLC care pentru oricare cuvânt de 4 biţi aplicat pe intrare generează la ieşire numărul binar care exprimă numărul de biţi 1 prezenţi în cuvântul de intrare (numărul de biţi 1 va fi exprimat in binar natural).

INTRARI				*IESIRI*		
x_3	x_2	x_1	x_0	y_2	y_1	y_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	0	0	1
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	0	1	1
1	1	0	0	0	1	0
1	1	0	1	0	1	1
1	1	1	0	0	1	1
1	1	1	1	1	0	0

Figura 2.5 Tabelul de adevăr pentru exemplul 2.2

Soluţie. Cuvintele aplicate pe intrare sunt toate cele 16 configuraţii de 4 biţi iar cuvintele de ieşire pot fi doar numerele $0, 1, 2, 3$ şi 4 exprimate în binar natural, deci circuitul trebuie să aibă 4 intrări şi 3 ieşiri. Tabelul de adevăr este dat in Figura 2.5.

2.2.2 Reprezentarea analitică

Reprezentarea analitică, sub forma unei funcţii logice (Definiţiile 1.8 şi 1.9) este forma de descriere cea mai potrivită pentru un CLC, mai ales dacă această formă se reduce la o expresie compactă atât pentru cazul când numărul de intrai n are valori mici

cât şi când are valori foarte mari. Expresia funcţiei ramâne compactă pentru un CLC chiar şi pentru n de valoare ridicată, dacă poate exista o exprimarea recurentă a funcţiei; ceea ce se reflectă în implementare printr-o structurare iterativă, adică o replicabilitate a unei structuri elementare (celulă).

Pentru CLC care nu realizează funcţii uzuale, cu exprimări cunoscute, expresia funcţiei logice se obţine din tabelul de adevăr întâi sub forma canonică normală disjunctivă/conjunctivă, relaţiile 1.10, 1.11, sau sub forma normală disjunctivă/conjunctivă, relaţile 1.12 şi 1.13 (sinteză pe bază de 1-uri respectiv pe bază de 0-uri) şi apoi prin reduceri succesive se poate obţine o formă redusă, care nu totdeauna este forma minimă.

Frecvent, forma canonică normală disjunctivă este transpusă intr-o exprimare recursivă ceea ce poate duce ca un CLC cu număr mare de intrări să poată fi structurat repetitiv cu un acelaşi tip de circuit dar care are un număr de intrări mult mai mic. Modalitatea de exprimare recursivă va fi prezentată in continuare.

Formele canonice normal disjunctive pentru funcţiile de una, două şi trei variabile sunt:

$$f_i{}^1 = d_{i_0}\bar{x}_0 + d_{i_1}x_0; \qquad i = 0,1,2,3.$$

$$\begin{aligned} f_i{}^2 &= d_{i_0}\bar{x}_1\bar{x}_0 + d_{i_1}\bar{x}_1x_0 + d_{i_2}x_1\bar{x}_0 + d_{i_3}x_1x_0 = \\ &= (d_{i_0}\bar{x}_0 + d_{i_1}x_0)\bar{x}_1 + (d_{i_2}\bar{x}_0 + d_{i_3}x_0)x_1 = \\ &= f_i{}^{1}\bar{x}_1 + f_i{}^{1*}x_1; \qquad i = 0,1...,15 \end{aligned} \tag{2.3}$$

$$\begin{aligned} f_i{}^3 = {} & d_{i_0}\bar{x}_2\bar{x}_1\bar{x}_0 + d_{i_1}\bar{x}_2\bar{x}_1x_0 + d_{i_2}\bar{x}_2x_1\bar{x}_0 + d_{i_3}\bar{x}_2x_1x_0 + \\ & + d_{i_4}x_2\bar{x}_1\bar{x}_0 + d_{i_5}x_2\bar{x}_1x_0 + d_{i_6}x_2x_1\bar{x}_0 + d_{i_7}x_2x_1x_0 = \\ = {} & (d_{i_0}\bar{x}_1\bar{x}_0 + d_{i_1}x_1x_0 + d_{i_2}x_1x_0 + d_{i_3}x_1x_0)\bar{x}_2 + (d_{i_4}\bar{x}_1\bar{x}_0 + \\ & + d_{i_5}x_1x_0 + d_{i_6}x_1\bar{x}_0 + d_{i_7}x_1x_0)x_2 = \\ = {} & f_i{}^2\bar{x}_2 + f_i{}^{2*}x_2; \qquad i = 0,1,2...,255. \end{aligned} \tag{2.4}$$

iar pentru n variabile se obţine:

$$f_i{}^n = f_i{}^{n-1}\bar{x}_{n-1} + f_i{}^{(n-1)*}x_{n-1}, \qquad unde\ i \in [0, 2^{2^n}) \tag{2.5}$$

Se observă că o funcţie de n variabile, $f_i{}^n$, se exprimă ca o funcţie de o singură variabilă x_{n-1} ai cărei coeficienţi sunt 2 funcţii reziduu $f_i{}^{n-1}$, $f_i{}^{(n-1)*}$ de celelalte $n-1$ variabile $(x_{n-2}, x_{n-3}, ..., x_2, x_1, x_0)$. La fel, funcţia de $n-1$ variabile $f_i{}^{n-1}$ se exprimă ca o funcţie de o singură variabilă x_{n-2} ai cărei coeficienţi sunt 2 funcţii reziduu de celelalte $n-2$ variabile ş.a.m.d. pâna la funcţia de o singură variabilă $f_i{}^1$, care este o sumă de două produse. Toate funcţiile, indiferent de numărul variabilelor, pot fi exprimate ca o funcţie de o singură variabilă, celelelte variabile sunt introduse în cei doi coeficienţi ca funcţii reziduu. Structura de circuit (modulul) care modelează fiecare funcţie $f_i{}^k$, $k = 1,2...,n$ este reprezentată în Figura 2.6-a. Pornind de la funcţia $f_i{}^n$, modelată cu acest modul, la care se introduce succesiv, pentru funcţiile reziduu, acelaşi tip de modul pâna la funcţia de o singură variabilă se obţine un circuit cu structură de arbore binar, numărul nivelurilor de module este egal cu n. Intrările în primul nivel, pe porţile AND, sunt perechi formate din variabilele $\overline{x_0}, x_0$ şi coeficienţii funcţiei d_{ij} care generează produsele: $d_{i_0}\overline{x_0}, d_{i_1}x_0, d_{i_2}\overline{x_0}, d_{i_3}x_0, ..., d_{i2^n-2}\overline{x_0}, d_{i2^n-1}x_0$. Coeficienţii d_{ij} $j = 0,1...,(2^n-1)$ sunt valorile din tabelul de adevăr al funcţiei $f_i{}^n$. Deoarece modulele din primul nivel calculează expresii banale de forma $d_{ij}\overline{x_0} +$

$d_{i(j+1)}x_0$, care pot avea numai valorile: $0, \overline{x_0}, x_0, 1$, aceste module pot fi eliminate şi înlocuite cu una din aceste valori, care nu trebuie calculate, deoarece există în sistem. Această analiză poate fi continuată şi la nivelurile următoare de module pâna la nivelul n, iar modulele care calculează expresii banale sunt înlocuite. În general, prin eliminările de module, se obţine un circuit de dimensiune mult mai mică decât arborele binar iniţial.

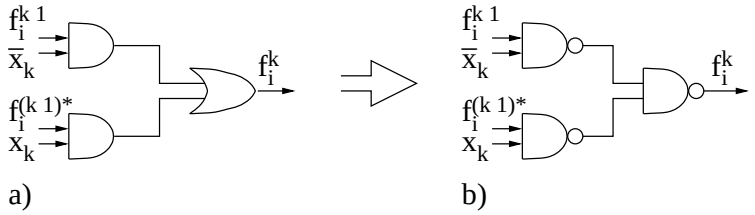


Figura 2.6 Modulul pentru implementarea recurentă a unei funcţii: a) cu structură AND-OR; b) structură NAND - NAND

O astfel de structură arborescentă este o "platformă" pentru implementarea oricărei funcţii de n variabile din cele 2^{2^n} funcţii posibile. Particularizarea "platformei" pentru implementarea unei anumite funcţii $f_i{}^n$ se realizează prin aplicarea pe fiecare poartă AND, din primul nivel, a câte unei valori a coeficienţilor d_{ij} în ordinea în care aceştia sunt în tabelul de adevăr al funcţiei respective.

Structurarea arborelui, pentru implementarea recurentă a funcţiei, sub formă de cascadă AND - OR - AND - OR - ... este adecvată pentru o conversie numai cu porţi NAND ($d_{i_0}\overline{x} + d_{i_1}x = \overline{\overline{d_{i_0}\overline{x}} \cdot \overline{d_{i_1}\overline{x}}}$), ca în Figura 2.6-b. Prin faptul că ceastă structură poate fi realizată cu acelaşi tip de poartă cu două intrări este recomandată pentru implementările pe arii de porţi logice (secţiunea 4.3).

Exemplul 2.3 Pentru celula sumator complet cu tabelul de adevar 1.6 să se modeleze funcţia sumă s_i pe un circuit cu structură de arbore.

Soluţie. Pe porţile AND din primul nivel, Figura 2.7, se aplică prima variabilă C_{i-1}, alternând negata $\overline{C_{i-1}}$ şi nenegata C_{i-1}, iar coeficienţii funcţiei s_i se aplica pe intrări în ordinea în care sunt în tabelul 1.6. Toate aceste poţi AND calculează valori banale, 0,$C_{i-1}, \overline{C_{i-1}}$ care există în sistem, deci porţile AND din primul nivel pot fi eliminate, la fel şi porţile OR pot fi eliminate, rezultă că primul nivel poate lipsi. Pe nivelul 2 de module se aplică ieşirile de la primul nivel şi alternativ variabila $\overline{B_i}$ şi B_i; pe acest nivel nu se mai pot elimina porţi. Pe nivelul 3 se aplică expresiile calculate pe nivelul 2 şi alternând variabilele A_i şi $\overline{A_i}$. Rezultă o modelare numai pe două niveluri de module. Analizând expresiile calculate după fiecare nivel rezultă pentru celula sumator complet expresia cunoscută a sumă $s_i = A_i \oplus B_i \oplus C_{i-1}$. Această structură arborescentă poate implementa oricare funcţie de 3 variabile, particularizarea pentru o anumită funcţie se face prin modificarea coeficienţilor pe intrările din primul nivel (care se citeşte din tabelul de adevăr al funcţiei).

De asemenea, forma canonică normală disjunctivă a funcţiei de n variabile poate fi transcrisă într-o reprezentare de funcţie numai de 2 variabile x_1, x_0, restul de $n-2$ variabile se introduc în coeficienţi sub forma unor funcţii reziduu. Astfel, pentru

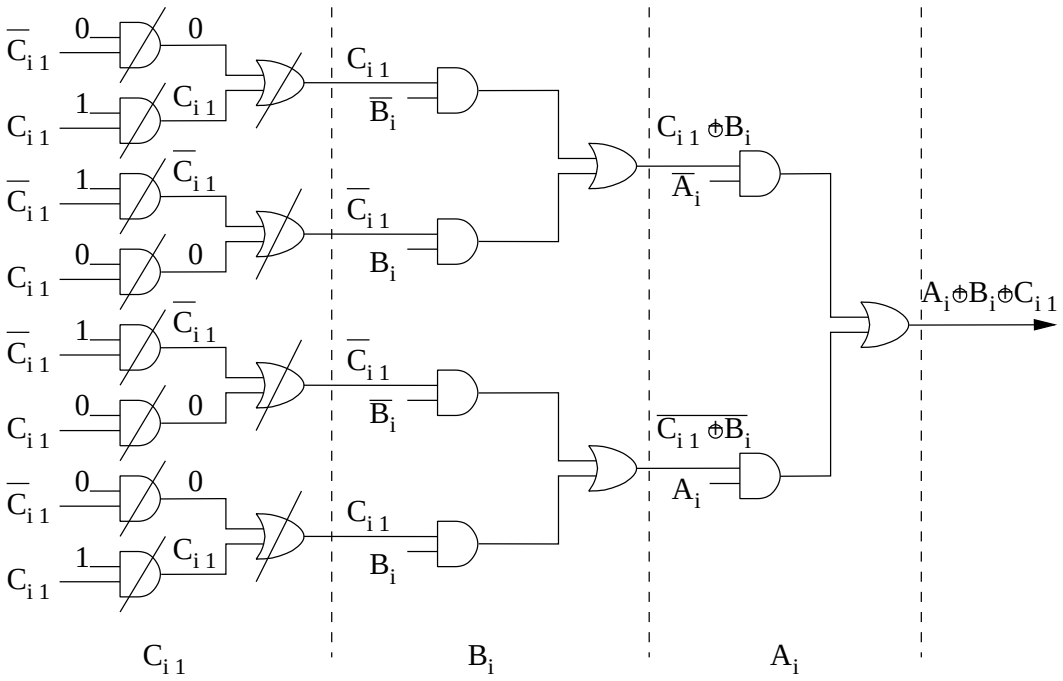


Figura 2.7 Implementarea funcţiei suma s_i, şi a unei celule sumator complet, pe o structură arborescentă.

funcţia ${f_i}^3$, relaţia 2.4, se obţine următoarea formă:

$$\begin{aligned} {f_i}^3 &= (d_{i_0}\bar{x}_2 + d_{i_4}x_2)\bar{x}_1\bar{x}_0 + (d_{i_1}\bar{x}_2 + d_{i_5}x_2)\bar{x}_1x_0 + \\ &+(d_{i_2}\bar{x}_2 + d_{i_6}x_2)x_1\bar{x}_0 + (d_{i_3}\bar{x}_2 + d_{i_7}x_2)x_1x_0 \end{aligned} \tag{2.6}$$

iar pentru o funcţie ${f_i}^n$ rezultă:

$$\begin{aligned} {f_i}^n &= {f_k}^{n-2}\cdot\bar{x}_1\bar{x}_0 + {f_l}^{n-2}\cdot\bar{x}_1x_0 + {f_p}^{n-2}\cdot x_1\bar{x}_0 + {f_q}^{n-2}\cdot x_1x_0 \\ &\text{iar } i \in [0, 2^{2^n}) \text{ şi } k,l,p,q \in [0, 2^{2^{n-2}}) \end{aligned}$$

Pentru un CLC cu 4 intrări A, B, C, D, având tabelul de adevăr din Figura 2.4-a, se poate obţine o exprimare ca o funcţie de numai 3 variabile A, B, C cu variabila D introdusă în coeficienţii funcţiei, Figura 2.4-b, sau ca o funcţie numai cu două variabile A, B, celelalte două variabile, C, D sunt introduse ca variabile reziduu în coeficienţii funcţiei, Figura 2.4-c.

În cazul general, oricare funcţie ${f_i}^n$ poate fi exprimată ca o funcţie numai de j variabile $x_{j-1}, x_{j-2}, ..., x_1, x_0$, restul variabilelor $x_{n-1}, x_{n-2}, ..., x_{n-1-j}$ (variabile reziduu) fiind incluse in coeficienţii funcţiei.

$$\begin{aligned} {f_i}^n &= {f_k}^{n-j}\cdot x_{j-1}x_{j-2}...x_1x_0 + ... + {f_q}^{n-j}\cdot\bar{x}_{j-1}\bar{x}_{j-2}...\bar{x}_1\bar{x}_0 \\ &\text{iar } i \in [0, 2^{2^n}),\ k, ..., q \in [0, 2^{2^{(n-j)}}) \end{aligned} \tag{2.7}$$

În concluzie, o funcţie ${f_i}^n$ poate fi exprimată ca o funcţie de $n, n-1, n-2, ..., 3, 2, 1$ variabile dacă respectiv un număr de $0, 1, 2, ..., n-3, n-2, n-1$ variabile sunt introduse ca variabile reziduu în expresiile coeficienţilor.

Exprimarea funcţiilor cu variabile reziduu poate determina implementări mai simple. Intuitiv, dar nu totdeauna, o funcţie cu mai puţine variabile necesită o implementare mai puţin costisitoare. Acceptând această afirmaţie, se aduce funcţia la un număr cât mai mic de variabile şi implementarea va fi mai ieftină dacă şi coeficienţii de variabile reziduu se reduc la valori banale sau se calculează cu circuite simple.

2.2.3 Diagrama Veitch - Karnaugh

Diagrama Veitch - Karnaugh (**V-K**) este o altă modalitate de a descrie un CLC. Diagrama Veitch - Karnaugh, ca modalitate grafică de descriere, nu este prea uzuală dar, în schimb, utilzarea diagramei V-K pentru minimizarea unui CLC, pornind de la tabelul de adevăr sau de la funcţia logică, este un instrument de lucru uzual în proiectare sau analiză. Acest instrument grafic pentru un CLC cu n intrări şi o singură ieşire, este o diagramă dreptunghiulară sau pătrată care conţine 2^n căsuţe elementare pe care sunt mapate, fie cele 2^n valori ale coeficienţilor funcţiei din tabelul de adevăr, fie cei 2^n temeni canonici P_i sau S_i ai funcţiei (relaţiile 1.10 şi 1.11). Modul cum se scrie "coordonata" unei căsuţe elementare determină cele 2 variante: diagrama Veitch şi diagrama Karnaugh.

Într-o nuanţa de tratare didactică, fiecare din cele 2 diagrame sunt prezentate separat, atât pentru exprimarea funcţiei sub forma canonică normală disjunctivă (termenii produs P), cât şi pentru forma canonică normală conjunctivă (termeni suma S), pentru $n = 2, 3, 4, 6$, Figura 2.7.

În varianta Veitch, coordonata unei căsuţe elementare este exprimată prin variabilele funcţiei şi aceste variabile sunt notate pe marginile diagramei, iar in acea căsuţa se introduce valoarea lui P_i sau S_i (indicele i este numărul zecimal a cărui reprezentare în binar natural rezultă din codul format de variabilele coordonate ale casuţei). În varianta Karnaugh, coordonata unei căsuţe elementare este exprimată prin valorile variabilelor (conform codificării mintermilor sau maxtermilor, 1.1.4) şi aceste valori ale variabilelor sunt notate pe marginile diagramei; în acea căsuţa se introduce coeficientul d_i din tabelul de adevăr corespunzător pentru configuraţia valorilor de intrare (care constituie coordonatele căsuţei). Ambele variante reprezintă acelaşi CLC, la varianta Veitch se ajunge mai uşor pornind de la forma canonică a funcţiei iar la varianta Karnaugh se ajunge mai uşor de la tabelul de adevăr. De fapt, în practică, notaţiile de la varianta Veitch şi de la varianta Karnaugh se mixează rezultând diagrama referită ca Veitch - Karnaugh, la care este redundanţa de informaţie, adică sunt prezente atăt variabilele cât şi valorile variabilelor.

În figurile 2.8-a şi 2.8-b pentru $n = 2, n = 3$ sunt reprezentate separat diagramele Veitch şi Karnaugh şi pentru fiecare dintre ele s-a figurat atât forma canonică normală disjunctivă (P), cât şi forma canonică normală conjunctivă (S). Dar, in Figura 2.8-c, pentru $n = 4$, la cele 2 diagrame notarea (variabilele şi valorile) este intermixată atât pentru forma canonică normală disjunctivă cât şi pentru forma canonică normală conjunctivă (adică, pe marginile diagramelor sunt notate atât variabilele, cât şi valorile acestor variabile). Diagramele pentru $n = 5$ se obţin prin alăturarea de două diagrame de $n = 4$ iar pentru $n = 6$ din alăturarea a patru diagrame de $n = 4$ sau două diagrame de $n = 5$. Dar pentru a păstra adiacenţa la alaturare a două diagrame de $n = 4$ trebuie păstrată numărarea in cod *Gray* adică, $000, 001, 011, 010, 110, 111, 101, 100$.

În oricare din diagrame, pentru cele 2^n căsuţe elementare, fiecare variabilă intră

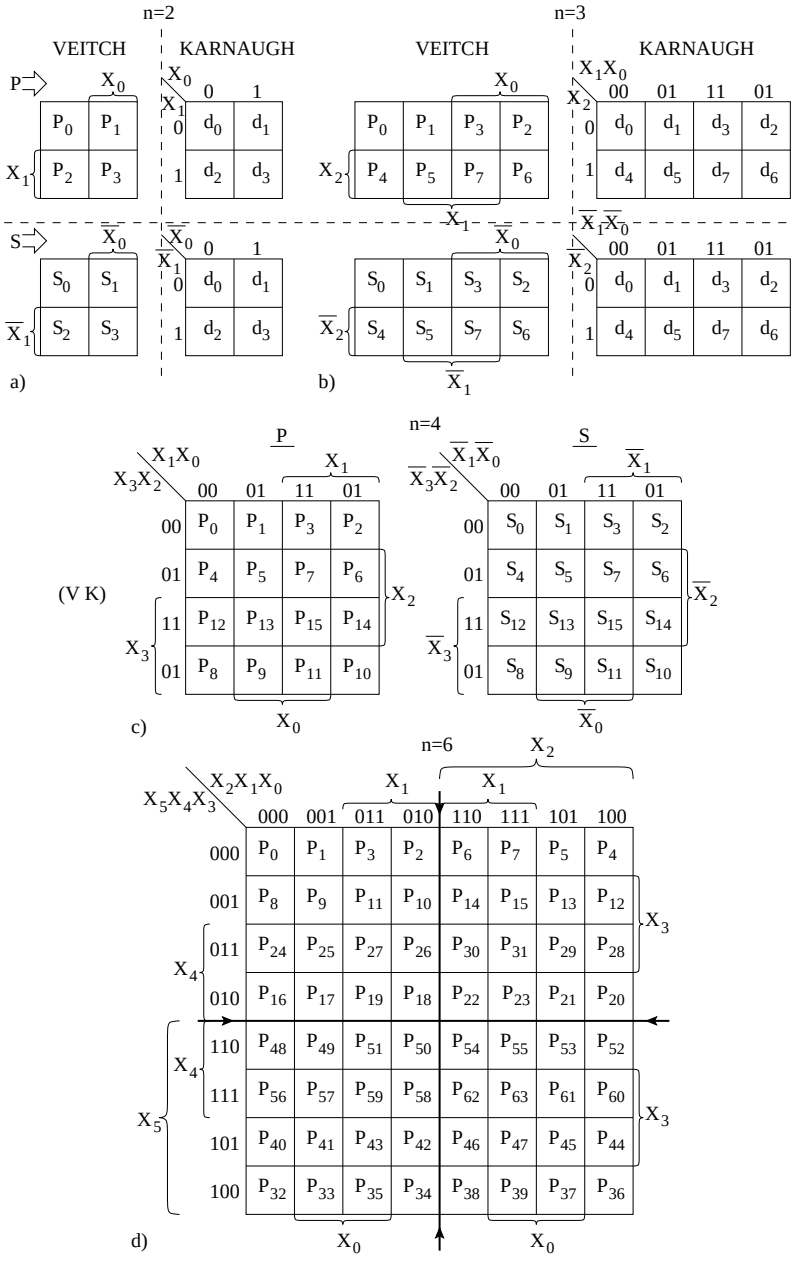


Figura 2.8 Diagramele Veitch şi Karnaugh: a,b) forma P şi forma S pentru $n = 2$ şi $n = 3$; c) diagrama V-K forma S şi P pentru $n = 4$; d) diagrama V-K forma P pentru $n = 6$.

negată în coordonatele a $2^n/2$ căsuţe şi nenegată în restul de $2^n/2$ căsuţe. Trecerea de la o diagramă pentru forma P la o diagramă pentru forma S se realizează prin substituirea în coordonate a variabilei nenegată cu variabila negată, $x \rightarrow \overline{x}$, iar a celor negate cu nenegate $\overline{x} \rightarrow x$ şi în interiorul căsuţei se substituie termenii $P_i \rightarrow S_i$. Trecerea de la forma S la cea P este, la fel, o negare a variabilelor coordonate ale casuţelor iar în căsuţe se substituie termenii $S_i \rightarrow P_i$.

În fiecare diagramă V-K două căsuţe vecine sunt adiacente, adică distanţa Humming este egală cu 1. Adiacenţa rezultă in urma faptului că notarea, pe marginea diagramelor, a coordonatelor căsuţelor se face prin numărare în cod Gray. Adiacenţă există între căsuţele elementare de pe fiecare două linii alăturate sau de pe fiecare două coloane alăturate, adiacenţa se extinde şi intre cele două linii extreme sau între cele 2 colane extreme (de exemplu, in Figura 2.8 - c între coloanele 00 cu 01 sau între liniile 00 cu 01). Nu sunt adiacente două căsuţe care sunt situate în dinagonală una faţă de pe alta.

Datorită adiacenţei, când se trece de la scrierea coordonatei unei căsuţe elementare la scrierea coordonatei a două căsuţe elementare, adică luate împreună (2^1), din coordonata rezultată va lipsi variabila care îşi modifică valoarea între cele două căsuţe vecine (deoarece $x + \bar{x} = 1$ şi $x \cdot \bar{x} = 0$). De exemplu, in Figura 2.8-c pentru forma P, când se scrie coordonata pentru căsuţele adiacente P_5 şi P_7 luate împreună

$$\bar{x}_3 x_2 \bar{x}_1 x_0 + \bar{x}_3 x_2 x_1 x_0 = \bar{x}_3 x_2 x_0 (\bar{x}_1 + x_1) = \bar{x}_3 x_2 x_0$$

coordonata rezultată are o variabilă mai puţin, deoarece a dispărut variabila x_1 care are valoarea diferită în cele două coduri 0101 şi 0111 (x_1 schimbă valoarea la trecerea dintre P_5 şi P_7).

În aceeaşi figură, pentru forma S când se scrie coordonata pentru căsuţele adicente S_5 şi S_7 luate împreună

$$(x_3 + \bar{x}_2 + x_1 + \bar{x}_0)(x_3 + \bar{x}_2 + \bar{x}_1 + \bar{x}_0) = (x_3 + \bar{x}_2 + \bar{x}_0)$$

dispare variabila x_1.

Dacă se consideră o suprafaţă care grupează patru căsuţe adiacente (2^2) în coordonata comună rezultantă se vor elimina două variabile; pentru un grup de opt căsuţe adiacente (2^3) se vor elimina trei variabile, iar pentru un grup de 2^n căsuţe adiacente se vor elimina n variabile. Această modalitate grafică, de indentificare de suprafeţe ce grupează căsuţe adiacente, în fond, este o grupare de termeni canonici, dar în varianta grafică operaţia de reducere a termenilor canonici este mai simplă şi mai puţin supusă erorii decât atunci când se lucrează cu formele analitice normale conjunctive, FNC, sau normale disjunctive, FND, ale funcţiilor.

În diagrama Veitch - Karnaugh se introduc în fiecare căsuţa elementară valorile coeficienţilor din tabelul de adevăr al funcţiei. Când se face sinteza funcţiilor pe baza FND se consideră toţi mintermii care au valoarea 1 iar pentru sinteza pe baza de FNC se consideră toţi maxtermii care au valoarea 0. Rezultă că in diagrama V-K de tip P sau S trebuie luate respectiv toate căsuţele elementare care au valoarea unu sau toate cele care au valoarea zero şi aceasta se face identificând suprafeţe care cuprind căsuţele adiacente grupate in numar de puteri ale lui doi. Evident, se caută a se forma suprafeţe care să cuprindă un număr, de puteri ale lui doi, de căsuţe adiacente cât mai mare. Vom referi coordonata unei suprafeţe de căsuţe adiacente

grupate cu termenul de **Implicant Prim, IP**. În această operaţie de identificare de suprafeţe maxime poate apare o suprafaţa ce acoperă o grupare de căsuţe adiacente, dar fiecare din aceste căsuţe elementare ale grupării mai este acoperită cel puţin de încă o altă suprafaţa; o astfel de suprafaţa corespunde unui **IP neesenţial**. Dar poate există o suprafaţa care acoperă o grupare de căsuţe adiacente dar dintre aceste căsuţe elementare există cel puţin o căsuţa care nu mai este acoperită şi de către o altă suprafaţă; o astfel de suprafaţa corespunde unui **IP** esenţial. Pentru exprimarea funcţiei în forma redusă trebuie luate în considerare suprafeţe maxime astfel încât să fie acoperite toate căsuţele elementare cu valoarea 1 pentru forma redusă disjunctivă (sumă de produse), respectiv să fie acoperite toate căsuţele elementare cu valoarea 0 pentru forma redusă conjunctivă (produs de sume).

Definiţia 2.4 Forma redusă a unei sume de produse este minimă atunci când nu există o altă formă redusă a funcţiei cu mai puţini termeni produs sau oricare altă formă cu acelaşi număr de termeni produs are cel puţin acelaşi număr de variabile. ◇

După cum s-a arătat în secţiunea 1.5 forma redusă ca sumă de produse se implementează simplu pe o structură AND-OR, iar forma redusă ca produs de sume pe o structură OR-AND.

Rezultă, implicit, pentru implementare, că forma minimă va duce la un număr minim de porţi pe primul nivel (de AND) şi de intrări pe porţile din nivelul doi (OR) şi, evident, in cadrul acesta la cel mai mic număr de intrări (variabile) pe primul nivel.

La fel se poate da o definiţie pentru forma minimă a unui produs de sume.

Pentru exprimarea funcţiei în formă redusă se selectează:

1. obligatoriu toţi IP esenţiali;

2. un număr cât mai mic de IP neesenţiali, dar care să acopere toate căsuţele elementare înscrise cu unu ce nu au fost acoperite de către IP esenţiali

---

Exemplul 2.4 Pentru funcţia F dată sub forma FND,

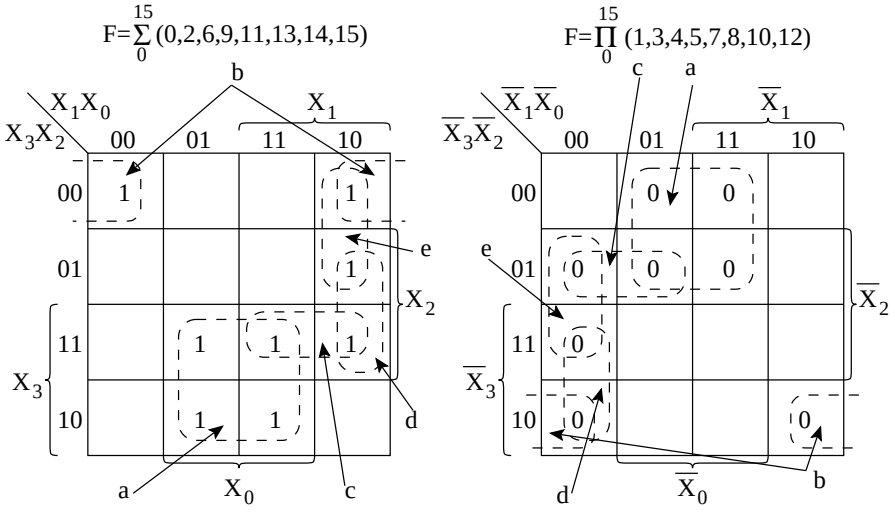
$$F = \sum_{0}^{15}(0, 2, 6, 9, 11, 13, 14, 15)$$

sau sub forma FNC

$$F = \prod_{0}^{15}(1, 3, 4, 5, 7, 8, 10, 12)$$

să se realizeze minimizarea cu diagrama V-K.

Soluţie. Din expresia funcţiei dată sub forma listelor anterioare se completează cu unu diagrama V-K pentru sinteza ca sumă de produse, Figura 2.9-a şi se completează cu 0 diagrama V-K pentru sinteza ca produs de sume, Figura 2.9-b. Se consideră suprafeţe de căsuţe adiacente, grupate câte două sau patru şi rezultă pentru forma sumă de produse doi IP esenţiali ($a = x_3x_0,\ b = \bar{x}_3\bar{x}_2\bar{x}_0$) şi trei IP neesenţiali ($c = x_3x_2x_1,\ d = x_2x_1\bar{x}_0,\ e = \bar{x}_3x_1\bar{x}_0$), iar pentru forma produse de sume doi IP esenţiali ($a = (x_3 + \bar{x}_0),\ b = (\bar{x}_3 + x_2 + x_0)$) şi trei IP neesenţiali ($c = (x_3 + \bar{x}_2 + x_1),\ d = (\bar{x}_3 + x_1 + x_0),\ e = (\bar{x}_2 + x_1 + x_0)$). Acoperirea completă pentru prima sumă de produse trebuie sa conţina cei doi IP esenţiali a şi b iar dintre cei neesenţiali un număr cât mai mic, dar care să acopere restul de căsuţe în care este



a)
IP esentiali:
$a=X_3X_0$; $b=\overline{X}_3\overline{X}_2\overline{X}_0$
IP neesentiali:
$c=X_3X_2X_1$
$d=X_2X_1\overline{X}_0$
$e=\overline{X}_3X_1\overline{X}_0$

b)
IP esentiali:
$a=(X_3+\overline{X}_0)$; $b=(\overline{X}_3+X_2+X_0)$
IP neesentiali:
$c=(X_3+\overline{X}_2+X_1)$
$d=(\overline{X}_3+X_1+X_0)$
$e=(\overline{X}_2+X_1+X_0)$

Figura 2.9 Exemplu de minimizare: a) pe bază de 1 pentru funcţia $F = \sum_0^{15}(0, 2, 6, 9, 11, 13, 14, 15)$; b) pe bază de 0 pentru funcţia $F = \prod_0^{15}(1, 3, 4, 5, 7, 8, 10, 12)$.

înscrisă valoarea 1. Se obţin formulele de acoperire $a + b + d$ sau $a + b + c + e$. Evident, prima formă este forma minimă

$$f = x_3x_0 + \bar{x}_3\bar{x}_2\bar{x}_0 + x_2x_1\bar{x}_0$$

În acelaşi mod se obţine forma redusă pentru produs de sume:

$$f = (x_3 + \bar{x}_0)(\bar{x}_3 + x_2 + x_0)(\bar{x}_2 + x_1 + x_0)$$

Forma minimă a unei funcţii f ca produs de sume se obţine prin acoperirea tuturor căsuţelor elementare din diagrama V-K care au valorile 0. Dar se poate obţine aceeaşi formă minimă şi printr-o sinteză de suma de produse a funcţiei complementare $\bar{f}$. Dacă funcţia f are valori 1 în anumite căsuţe elementare evident că funcţia negată $\bar{f}$ are valori 1 in restul de căsuţe elementare ale diagramei V-K, adică tocmai în căsuţele în care funcţia f are valori 0, şi care pot fi utilizate pentru sinteza ca produs de sume. Deci, obţinând forma redusă ca sumă de produse a funcţiei complementare $\bar{f}$ şi negând această expresie rezultă forma redusă ca produs de sume a funcţiei f. În acest mod se obţine forma minimă a produsului de sume a lui f numai când forma redusă a lui $\bar{f}$ a fost cea minimă (şi aceasta depinde de cum s-au selectat IP neesenţiali). În Exemplul **2.4** , Figura 2.9-b, o formă redusă ca sumă de produse a lui $\bar{f}$ poate fi:

$$\bar{f} = \bar{x}_3x_0 + x_3\bar{x}_2\bar{x}_0 + x_2\bar{x}_1\bar{x}_0$$

iar prin negare şi aplicare teoremei De Morgan se obţine chiar forma minimă a produsului de sume:

$$f = (x_3 + \bar{x}_0)(\bar{x}_3 + x_2 + x_0)(\bar{x}_2 + x_1 + x_0)$$

obţinută prin sinteza pe bază de zero-uri.

Calculul formei reduse pentru $\bar{f}$, ca sumă de produse, este utilă pentru că unele dispozitive programabile (PLA) prezintă pe ieşire posibilitatea de a nega, printr-o poartă XOR, funcţia calculată, deci generarea formei produs de sume. Dacă se poate implementa uşor atât forma produs de sume, cât şi forma sumă de produse se pune intrebarea care dintre cele două forme reduse obţinute (nu totdeauna minime!) se alege? se va alege forma care are numărul minim de termeni.

2.2.3.1 Minimizarea funcţiilor incomplet definite

Exista situaţii în care funcţionarea unui CLC prin valoarea logică generată pe ieşire nu modifică cu nimic comportamentul util al circuitului. Valoarea logică pe ieşire se poate modifica fie în 1 fie în 0 dar aceste valori sunt indiferente (*do not care*) pentru utilitatea circuitului, de aceea ieşirea respectivă se notează in tabelul de adevăr sau în diagrama V-K cu simbolul "-". Astfel de situaţii apar când anumite configuraţii de intrare, restricţionate prin condiţiile de funcţionare, nu se aplică niciodată pe intrări, sau când pentru anumite configuraţii de intrare funcţionarea sistemului nu consideră valorile logice de pe ieşiri. Simbolurilor indiferente din diagramele V- K, prin includerea lor în grupuri de căsuţe elementare adiacente, pot aduce la expresii mult mai simple pentru IP esenţiali sau neesenţiali. Dupa caz, acestor simboluri li se

pot atribui valoarea 1, pentru sinteza ca sumă de produse, respectiv 0, pentru sinteza ca produs de sume, astfel încât suprafeţele care acoperă căsuţe adiacente sa devină cât mai mari.

Exemplul 2.5 Pentru convertorul BCD - EXCESS3, din Figura 2.3 să se deducă formulele minime pentru ieşirile E_3, E_2, E_1, E_0.

Soluţie. Din tabelul de adevăr din Figura 2.3 se obţin sub formă de listă valorile ieşirilor:

$$E_3 = \sum_{0}^{15}(5,6,7,8,9) + d(10,11,12,13,14,15)$$

$$E_2 = \sum_{0}^{15}(1,2,3,4,9) + d(10,11,12,13,14,15)$$

$$E_1 = \sum_{0}^{15}(0,3,4,7,8) + d(10,11,12,13,14,15)$$

$$E_0 = \sum_{0}^{15}(0,2,4,6,8) + d(10,11,12,13,14,15)$$

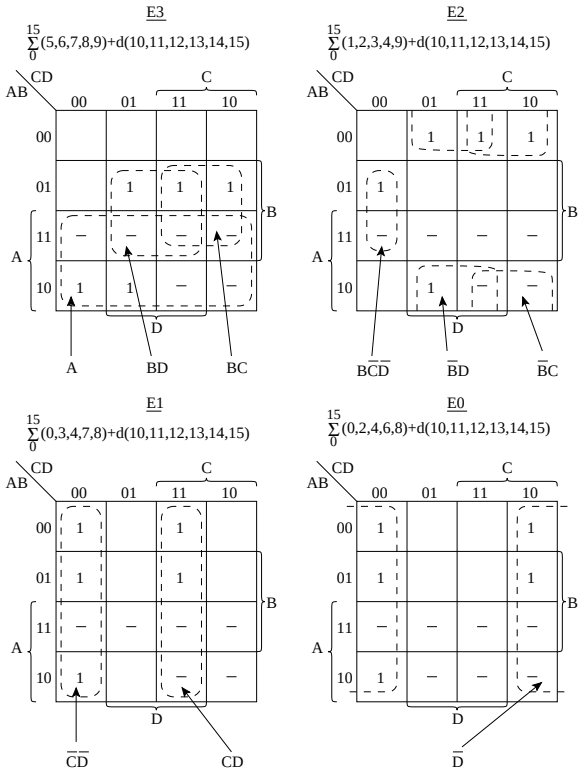


Figura 2.10 Minimizarea funţiilor incomplet definite. Exemplificare pentru sinteza convertorului BCD-EXCESS3, Exemplul 2.5.

Pe intrările circuitului se aplică numai codurile BCD, combinaţiile binare pentru numerele 10,11,12,13,14,15 nu se aplică niciodată, deci ieşirile corespunzătoare se notează cu indiferent în diagramele V-K din Figura 2.10. Pentru sinteză ca sume de produse, pentru ieşirile E_3, E_2, E_1, E_0, căsuţelor notate cu "-", ce intră în grupuri de adiacenţă selectate, li se atribuie valoarea logică 1. Se obţin astfel următoarele forme minime:

$$E_3 = A + BD + BC$$
$$E_2 = B\bar{C}\bar{D} + \bar{B}D + \bar{B}C$$
$$E_1 = \bar{C}\bar{D} + CD$$
$$E_0 = \bar{D}$$

2.2.3.2 Minimizare pe diagrame V-K cu variabile reziduu

Tabelul de adevăr, când numărul de variablie este mare, poate fi redus la o formă cu mai puţine linii dacă unele din variabilele funcţiei sunt introduse în coeficienţii funcţiei ca variabile reziduu, această modalitate de reducere a fost exemplificată in Figura 2.4. Similar, şi diagrama V-K poate fi redusă de la 2^n căsuţe la 2^{n-1} căsuţe când se introduce una din variabile în coeficienţii din interiorul căsuţelor, respectiv la 2^{n-2} căsuţe când se introduc două variabile reziduu şi aşa mai departe. Pentru o funcţie de două variabile, f_i^2, sau de trei variabile, f_i^3, formulele analitice cu o variabilă reziduu introdusă în coeficienţii funcţiei sunt cele date de relaţiile 2.3 respectiv 2.4.

Se observă că expresia unui coeficient reziduu, în funcţie de variabila reziduu x, este de forma $d_i\bar{x} + d_jx$. Ca structură diagrama V-K, în raport cu o variabilă care se introduce ca variabilă reziduu x, cuprinde 2^{n-1} căsuţe care au în coordonată variabila $\bar{x}$ şi 2^{n-1} căsuţe care au în coordonată variabila x. Căsuţele elementare care cuprind în coordonată viitoarea variabilă reziduu x se spară în două părţi egale dacă se trasează o linie prin mijlocul zonei corespunzătoarea variabilei x din diagrama V-K. În ambele părţi ale liniei mediane se grupează câte două căsuţe elementare, una care are în coordonată $\bar{x}$ şi coeficientul d_i şi una care are în coordonată x şi coeficientul d_j. Se efectuează calculul $d_i\bar{x}+d_jx$ care va fi noul coeficient al căsuţei elementare în diagrama V-K rezultată, iar coordonata acestei căsuţe rezultate este tocmai coordonata comună a celor două căsuţe elementare din care a provenit. Acest nou coeficient poate avea doar una din valorile $0, 1, x, \bar{x}$.

Această modalitate de reducere este prezentată în Figura 2.11-a pentru o funcţie de trei variabile când se elimină variabila x_1, iar in Figura 2.11-b pentru o funcţie de patru variabile când se elimină variabila x_0. Pentru fiecare din aceste cazuri se haşurează zona variabilei reziduu, respectiv se duce linia mediană prin această zonă şi se grupează o căsuţa din zona haşurată (x) cu una din zona nehaşurată ($\bar{x}$). De exemplu, pentru $n = 3$, prin gruparea căsuţei haşurate cu coeficientul d_3 cu cea nehaşurată cu coeficientul d_1 se obţine coeficientul $d_3\overline{x}+d_1x$ care se va înscrie în noua căsuţa de coordonate $\overline{x_2}x_0$ (coordonata comună a celor două căsuţe luate împreună). Iar în diagrama V-K de patru variabile (x_3, x_2, x_1, x_0) produsul $d_6\bar{x}_0 + d_7x_0$ care va fi în diagrama de trei variabile (x_3, x_2, x_1) coeficientul căsuţei de cordonate $\bar{x}_3x_2x_1$.

Diagrama cu variabile reziduu poate fi o formă mai compactă şi, uneori, cu astfel de diagrame se obţine mai uşor forma minimă. Deducerea formei reduse dintr-o diagramă care cuprinde coeficienţi cu variabile reziduu se face în următorii paşi:

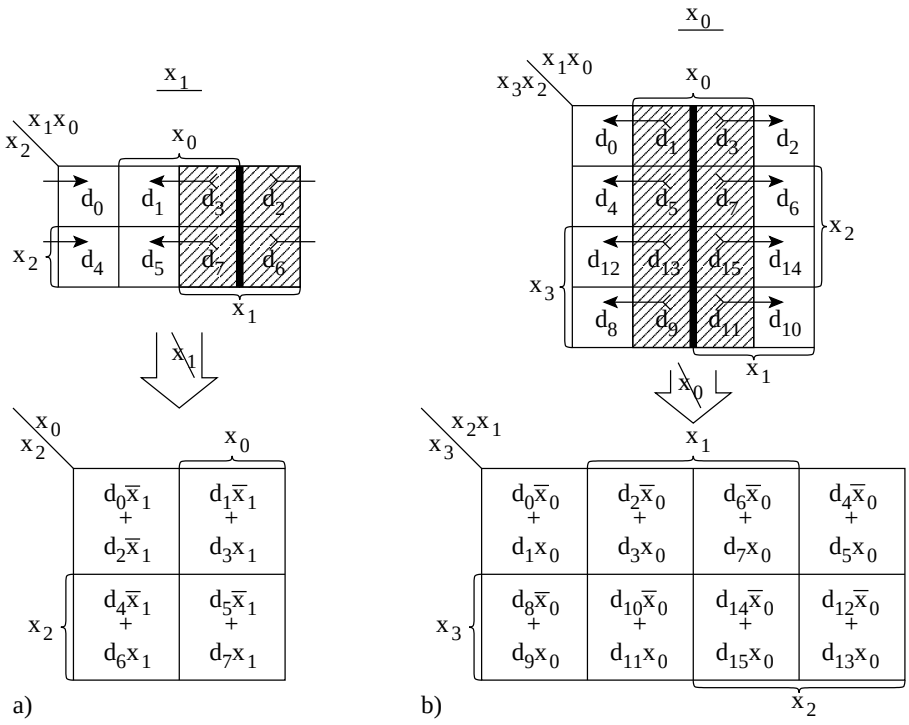


Figura 2.11 Exemplificare de introducere a unei variabile reziduu: a) pentru o diagramă V-K de trei variabile; b) pentru o diagramă V-K de patru variabile.

Pasul 1. In toate căsuţele elementare în care coeficientul conţine variabile reziduu se substituie acesta cu zero şi apoi se face extragerea funcţiei după regula normală prin gruparea suprafeţelor de 1-uri.

Pasul 2. In toate căsuţele în care coeficientul este 1 se substituie acesta cu indiferent şi apoi se face extragerea după gruparea suprafeţelor care cuprind coeficienţii care au aceeaşi expresie de variabile reziduu.

Pasul 3. Forma redusă a funcţiei se obţine prin sumarea logică a expresiei obţinută la Pasul 1 cu cea obţinută la Pasul 2.

Dar nu totdeauna forma redusă obţinută la la Pasul 3 este cea minimă. Pentru a obţine forma minmă se recomandă:

1. Folosind axiomele şi teoremele algebrei logice, Tabelul 1.2, să se transforme un coeficient compus din termeni produs, care conţine variabile reziduu, încât să se obţină un număr cât mai mic de termeni produs diferiţi.

2. Dacă într-o căsuţă există un coeficient compus dintr-o sumă de termeni produs, care conţin variabile reziduu, atunci această căsuţă se include, conform pasului 2 enunţat anterior, in fiecare din suprafeţele care acoperă unul dintre termenii produs din această sumă.

Exemplul 2.6 Pentru ieşirea E_3 a convertorului BCD - EXCESS3, cu tabelul de adevăr dat în Figura 2.3, să se deducă expresia minimă utillizând diagramele V-K pentru 4,3 şi 2 variabile.

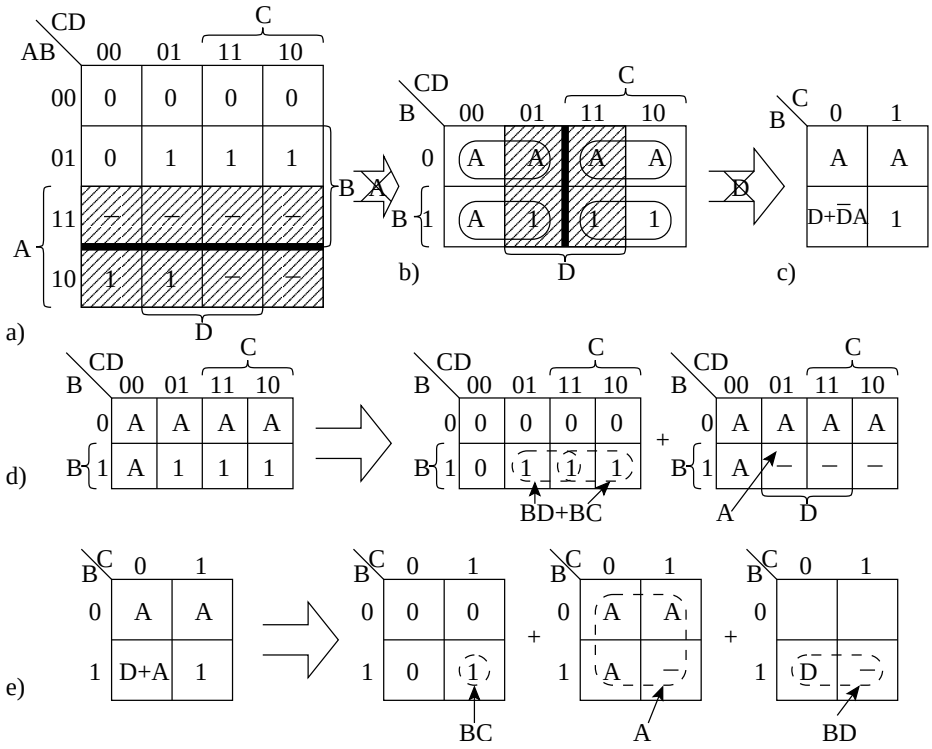


Figura 2.12 Pentru ieşirea E_3 a unui convertor BCD-EXCESS3 s-a dedus expresia minimă: a) utilizând diagrama V-K de patru variabile; b,d) diagrama V-K de trei variabile (B,C,D); c,e) diagrama V-K de două variablie (B,C).

Soluţie. Din prezentarea funcţiei E_3 într-o diagramă V-K de patru variabile, Figura 2.12-a, şi prin considerarea valorii 1 în căsuţele cu indiferent, se obţine forma minimă $E_3 = A + BD + BC$ (expresie obţinută şi în Exemplul **2.5**).

Introducând variabila A ca variabilă reziduu se obţine diagrama de trei variabile (B,C,D) din Figura 2.12-d. Aplicând, Figura 2.12-b, cei trei paşi pentru obţinerea formei minime, dintr-o diagramă cu variabilă reziduu, se obţine expresia minimă pentru E_3, identică cu cea anterioară.

Introducând şi variabila D ca variabilă reziduu se obţine diagrama V-K de două variabile (B,C) din Figura 2.12-c. Aplicând cei trei paşi pentru minimizare, dintr-o diagramă cu coeficienţi reziduu, se obţine forma redusă $E_3 = BC + A\overline{B} + BD + AB\overline{D}$, care diferă de forma minimă obţinută cu cele două diagrame anterioare. Pentru a obţine forma minimă trebuie făcute asupra coeficienţilor din diagrama din Figura 2.12-c următoarele transformări: Expresia coeficientului $D + \bar{D}A$, aplicând teorema de absorbţie inversă, devine $D + A$ ceea ce face ca în loc de 3 termeni produs ($A, D, \overline{D}A$), incluşi în coeficienţii reziduu din diagramă, să fie doar doi: *A si D*, Figura 2.12-e. Apoi, aplicând paşii pentru extragerea formei reduse,

primul pentru valorile de 1 iar al doilea pas separat pentru fiecare din cei doi termeni reziduu A *si* D, se obţine $E_3 = BC + A + BD$ care este forma minimă.

Uneori, prin extragerea formei reduse, dintr-o diagramă V-K care conţine variabile reziduu, nu se ajunge la forma minimă din cauza folosirii redundante a unor căsuţe cu valoarea 1. Poate exista situaţia în care o căsuţă elementară cu valoarea 1 în Pasul 1, care va avea valoarea indiferentă în Pasul 2, să fie inclusă în suprafaţa unui termen produs care conţine o variabilă reziduu (x) şi de asemenea să fie inclusă şi în suprafaţa unui alt termen produs care conţine aceeaşi variabilă reziduu doar negată ($\overline{x}$). Un astfel de 1 se numeşte *dublu acoperit*. Forma minimă se obţine doar când căsuţa cu un 1 dublu acoperit se consideră 0 în Pasul 1 de extragere (când se consideră suprafeţele de 1-uri).

Exemplul 2.7 Pentru CLC cu tabelul de adevăr din Figura 2.4 să se realizeze diagramele V-K de 4,3,2 variabile şi să se extragă expresia minmă.

Soluţie. Diagrama V-K pentru patru variabile (A, B, C, D) este reprezentată în Figura 2.13-a; s-a considerat valoarea coeficientului funcţiei egală cu 1 pentru configuraţia de intrare indiferentă $A\overline{B}\,\overline{C}\,\overline{D}$. Se extrage din această diagramă expresia minimă $A\overline{B}\,\overline{D}+\overline{A}BD+BC\overline{D}$.

Introducând variabila D ca variabilă reziduu se obţine diagrama V-K de trei variabile (A, B, C), Figura 2.13-b. Din această diagramă rezultă următoarea formă redusă: $\bar{A}BC + A\bar{B}\bar{D} + BC\bar{D} + \bar{A}BD$ care diferă de forma minimă extrasă din diagrama de patru variabile. În această formă redusă apare în plus termenul $\bar{A}BC$ care corespunde tocmai suprafeţei de 1. La o inspecţie mai atentă, Figura 2.13-d, se observă că 1 din diagramă este un 1 dublu acoperit. În Pasul 2, în căsuţa respectivă se consideră indiferent, aceasta este acoperită atât de suprafaţa lui D cât şi de suprafaţa lui $\bar{D}$, deci termenul $\bar{A}BC$ trebuie eliminat din forma redusă obţinându-se astfel forma mimimă.

Diagrama V-K de două variabile (A, B), Figura 2.13-c, se obţine din diagrama de trei variabile cu coeficienţi reziduu prin introducerea şi a variabilei C ca variabilă reziduu. Forma redusă obţinută din această diagramă este (Pasul 1 nu se aplică deoarece nu există suprafeţe de 1) $A\bar{B}\bar{D}+\bar{A}BC+\bar{A}BD+ABC\bar{D}$ care diferă de expresia minimă. Prin aplicarea teoremei de absorbţie inversă termenul $C + D$ devine $C\bar{D} + D$, deci în loc de patru termeni produs $C, D, \bar{D}, C\bar{D}$ în coeficienţii reziduu vor fi doar trei: $D, \bar{D}, C\bar{D}$. Aplicând doar Pasul 2 de extragere, deoarece nu există căsuţe cu coeficient 1, Figura 2.13 - e, se obţine forma minimă $A\bar{B}\bar{D} + \bar{A}BD + BC\bar{D}$.

2.2.3.3 Minimizarea prin diagrame V-K a circuitelor cu ieşiri multiple

S-a arătat că un CLC cu m ieşiri, Figura 2.1, poate fi considerat ca find compus din m circuite cu o singură ieşire. Păstrând acest mod de abordare, şi pentru minimizarea circuitului cu m ieşiri, procesul de minimizare se reduce la extragerea separată a fiecăreia din cele m ieşiri pe câte o diagramă V-K; şi, s-ar putea ca rezultatul să fie cel optim, dar numai când cele m funcţii de ieşire nu au în componenţa lor termeni produs comuni. Pentru a identifica eventualii termeni produs comuni, la unele din ieşirile circuitului, se impune ca procesul de reducere al funcţiei FNC să se facă corelat.

Pentru identificarea termenilor produs comuni în cele m funcţii de ieşire $f_0, f_1, ...$ f_{m-1} se procedează în felul următor:

1. Se realizează diagramama V-K a funcţiei produs logic între toate cele m funcţii

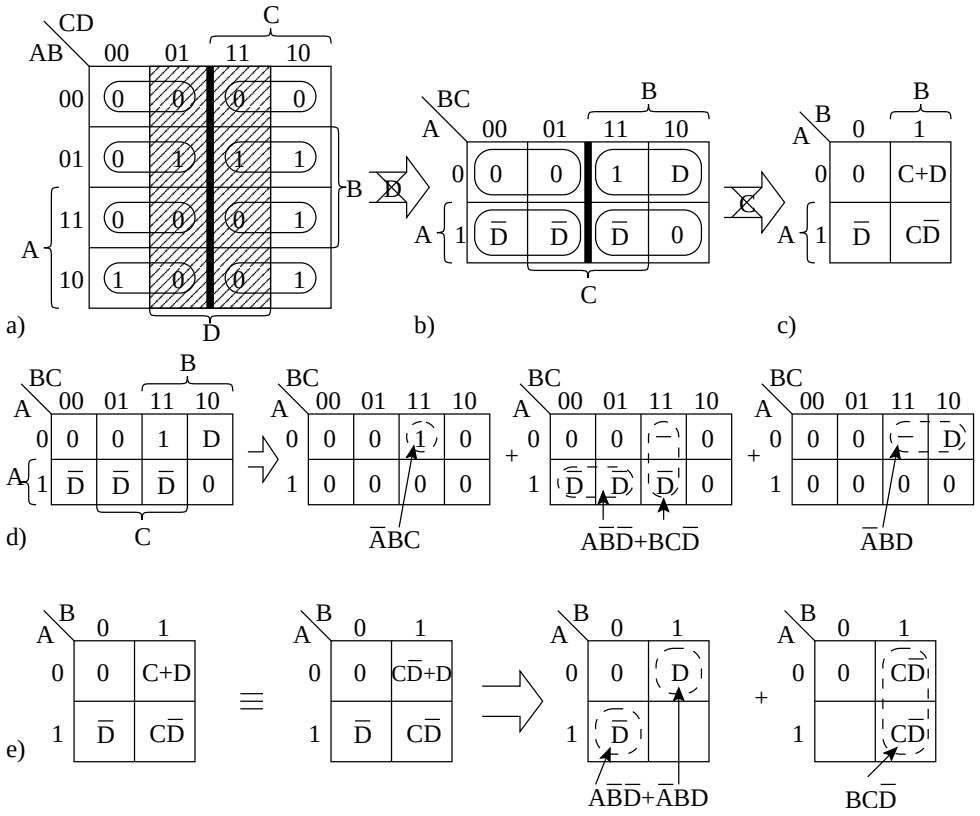


Figura 2.13 Exemplu de minimizare a unei funcţii de patru variabile (*A*,*B*,*C*,*D*): a) pe o diagramă V-K de patru variabile; b,d) pe o diagramă V-K de trei variabile (*A*,*B*,*C*); c,e) pe o diagramă V-K de două variabile (*A*,*B*);

iniţiale $f_0 \cap f_1 \cap f_2 \cap ... \cap f_{m-1}$ (practic se obţine printr-o intersecţie între toate diagramele V-K ale celor m funcţii). Se realizează diagramele V-K ale funcţiilor produs între câte $m-1$ din cele m funcţii, apoi se realizează diagramele V-K ale funcţiilor produs între câte $m-2$ din cele m funcţii ş.a.m.d. pâna la realizarea diagramelor V-K ale funcţiilor produs între câte 2 funcţii. Numărul total de astfel de funcţii (diagrame V- K) produs este egal cu $2^m * (m-1)$ la care se adaugă încă m diagrame ale funcţiilor iniţiale (pentru $m = 8$ rezultă 255 de diagrame!). Evident, această metodă neautomatizată este practicabilă doar la circuite cu cel mult 4 ieşiri.

2. În diagrama V-K produs logic de m funcţii se identifică dacă există implicanţii primi comuni pentru toate funcţiile iniţiale. Aceşti implicanţi primi identificaţi sunt eliminaţi din toate diagramele (V-K) produs de $m-1$ funcţii şi apoi din toate diagramele (V-K) produs pâna la diagramele produs de două funcţii. Pentru aceşti implicanţi primi identificaţi se figurează suprafeţele de acoperire în toate cele m diagrame V-K ale funcţiilor iniţiale. Se reia acelaşi proces de identificare a implicanţilor primi comuni pe diagramele V-K produs logic de $m-1$

funcţii pâna la figurarea suprafeţelor respective de acoperire în diagramele V-K ale funcţiilor iniţiale. Procesul de identificare de implicanţi primi comuni se contimuă pâna la diagramele V-K produs logic de două funcţii.

3. Din cele m diagrame V-K ale funcţiilor iniţiale se extrag formele reduse ale funcţiilor, după procedeul normal, încercând a se selecta cât mai multe dintre suprafeţele care figurează împlicanţi primi comuni (identificaţi conform procedurii de la punctul 2).

Circuitul cu ieşiri multiple, rezultat printr-o minimizare corelată, în general, utilizează un număr de termeni produs diferiţi mai mic decât numărul total de termeni produs diferiţi obţinut printr-o minimizare separată (necorelată) pentru fiecare funcţie în parte, deci un număr mai mic de porţi pentru implementare. Totuşi, alegerea finală pentru implementare nu este impusă numai de acest rezultat ci trebuie luate în considerare şi: tipul de poartă logică, factorii de încărcare la intrare şi ieşire şi disponibilitatea semnalelor in sistem.

Metoda minimizării corelată este indicată pentru sinteza circuitelor care se implementează cu porţi logice discrete, pe circuitele arii de porţi logice şi pe circuite logice programabile unde economisirea doar şi a unei porţi, la o replicare mare, determină o economisire substanţială.

Exemplul 2.8 Pentru următoarele trei funcţii

$$F_0(A,B,C,D) = \sum_0^{15}(2,4,6,7,9,11,12,15)$$

$$F_1(A,B,C,D) = \sum_0^{15}(4,6,7,10,14,15)$$

$$F_2(A,B,C,D) = \sum_0^{15}(3,7,8,10,12,14,15)$$

să se realizeze o minimizare corelată.

Soluţie. Extragerea necorelată a fiecărei funcţii este realizată pe diagramele V-K din Figura 2.14-a şi se obţin următoarele expresii minime:

$$F_0 = A\bar{B}D(1) + BCD(2) + B\bar{C}\bar{D}(3) + \bar{A}C\bar{D}(4)$$
$$F_1 = BC(5) + AC\bar{D}(6) + \bar{A}B\bar{D}(7)$$
$$F_2 = A\bar{D}(8) + BCD(2) + \bar{B}C\bar{D}(9)$$

cu un număr de 9 termeni produs diferiţi din totalul de 10 (BCD este utilizat atât de F_0 cât şi de F_2). În continuare se va proceda pentru extragerea corelată. Diagrama V-K a funcţiei produs logic $F_0 \cap F_1 \cap F_2$ este reprezentată în Figura 2.14-b iar a funcţiilor produs logic $F_0 \cap F_1, F_0 \cap F_2, F_1 \cap F_2$ în Figura 2.14-c. Singurul implicant prim comun, BCD din diagrama V-K a funcţiei produs logic $F_0 \cap F_1 \cap F_2$, neconsiderat în produsele logice de două funcţi, este figurat (haşurat spre dreapta) în diagramele V-K ale funcţiilor F_0, F_1, F_2 din Figura 2.14-d. Apoi, din diagramele funcţiilor produs logic de două funcţii: pentru $F_0 \cap F_1$ se identifică implicantul prim comun $\bar{A}B\bar{D}$ care se figurează (haşura spre stânga) prin suprefeţe pe F_0 şi F_1; pentru $F_0 \cap F_2$ se identifică implicantul prim commun $\bar{A}\bar{B}C\bar{D}$ care se figurează (printr-un cerculeţ haşurat spre dreapta) prin suprafeţe pe F_0 şi F_2; pentru $F_1 \cap F_2$ se identifică implicantul prim comun $AC\bar{D}$ care se figurează prin suprafeţe pe F_1 şi F_2.

In continuare, de pe diagrama V-K a fiecărei funcţii iniţiale, se extrage forma redusă după procedeul obişnuit căutând ca, în acoperirea căsuţelor elementare cu valoarea 1, să fie incluşi în primul rând implicanţi primi esenţiali şi implicanţi primi comuni identificaţi. Rezultă expresiile:

$$F_0 = A\bar{B}D(1) + BCD(2) + \bar{A}B\bar{D}(3) + \bar{A}\bar{B}C\bar{D}(4) + B\bar{C}\bar{D}(5)$$
$$F_1 = BCD(2) + \bar{A}B\bar{D}(3) + AC\bar{D}(6)$$
$$F_2 = BCD(2) + A\bar{C}\bar{D}(7) + AC\bar{D}(6) + \bar{A}\bar{B}C\bar{D}(4)$$

Prin minimizarea corelată s-au obţinut un număr de 7 termeni produs diferiţi dintr-un total de 12 termeni produs utilizaţi, deci 5 sunt utilizaţi de cel puţin două funcţii.

Obţinerea unei forme reduse, eventual minimă, pentru o funcţie prin operaţii analitice, paragraful 1.15, sau utilizând diagrama V-K pot constitui metode practicabile doar pentru funcţii de câteva variabile. Pentru funcţii ce depăşesc câteva variabile sunt utilizate alte metode. Oricare ar fi aceste metode, în fond, procesul de reducere parcurge două etape: (1) identificarea tuturor implicanţilor primi ai funcţiei şi (2) apoi selectarea unui set minimal de implicanţi primi care să acopere funcţia. Aceste etape pot fi realizate fie prin metode tabelare, de exemplu metoda Quine-McCluskey, fie prin abordări algoritmice, în termeni de structuri de date şi funcţii de limbaj de nivel înalt [Wakerly ′2001]. Evident, aceste abordări au ca finalitate un program de minimizare inclus intr-un mediu de proiectare, de exemplu programul Espress II sau varianta mai avansată Espress - MV. Includerea unui algoritm "exact" într-un program de minimizare pentru o funcţie cu zeci de variabile şi sute de termeni produs poate necesita un timp de calcul inacceptabil. Foarte frecvent o abordare euristică, în locul unui algoritm exact, poate genera o expresie minimă sau aproape minimă dar cu o reducere a timpului de calcul cu peste un ordin de mărime.

2.2.4 Diagrama de decizie binară, BDD

Informaţia conţinută într-un tabel de adevăr, Figura 2.15-a, poate fi reprezentată şi printr-un graf sub forma unui arbore binar, **arbore de decizie binar, BDT** (**B**inary **D**ecision **T**ree). Un arbore de decizie binar este un graf orientat, aciclic, care prezintă o rădăcină şi două tipuri de noduri: teminale şi neterminale. Fiecare nod x este referit/etichetat prin variabila $var(x)$ şi are arce direcţionate spre doi succesori:

- Low(x), care corespunde tranziţiei când variabilei x i se asignează valoarea 0
- High(x), care corespunde tranziţiei când variabilei x i se asignează valoarea 1

Fiecare nod terminal (frunză) x este caracterizat prin valoarea (x) care poate fi 0 sau 1. Pentru o anumită asignare a valorilor variabilelor unei funcţii (o configuraţie de intrare), parcurgând graful, de la rădăcină pe traseul indicat de valorile respective ale variabilelor, se atinge un nod terminal a cărui valoare este tocmai valoarea funcţiei corespunzătoare respectivei asignării a variabilelor. În Figura 2.15-b este reprezentat BDT pentru funcţia cu tabelul de adevăr din Figura 2.15-a. BDT nu reflectă o reprezentare concisă a unei funcţii booleene; în fond, un BDT nu este altceva decât o formă grafică a tabelului de adevăr, care prezintă multă redundanţă. De exemplu, pentru arborele obţinut există numai 3 subarbori diferiţi cu rădăcina de etichetă y_0,

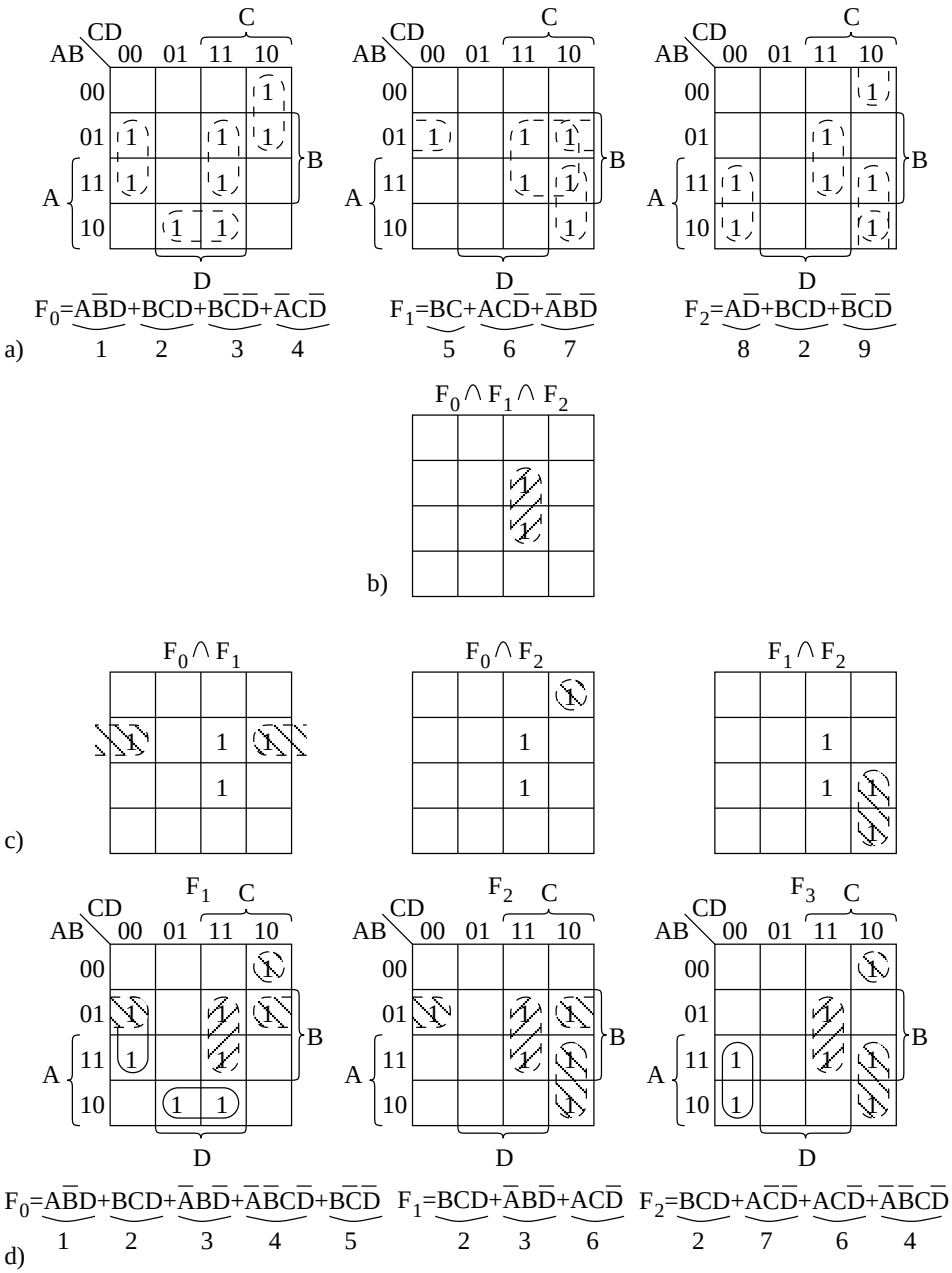


Figura 2.14 Minimizarea corelată a 3 funcţii: a) minimizarea necorelată; b,c) diagramele V-K pentru produsele de câte 3 şi 2 funcţii; d) extragerea corelată a expresiilor reduse ale funcţiilor.

din totalul de 8 subarbori cu rădăcina în y_0. Se poate obţine o formă mult mai concisă de reprezentare a funcţiei, denumită **Diagramă de Decizie Binară, BDD** (**B**inary **D**ecision **D**iagram), dacă din BDT se elimină redundanţa.

Obţinerea unui BDD, prin eliminarea redundanţei, apare în urma aplicării repetate (pâna când nu mai pot fi aplicate) a următoarelor trei proceduri:

1. Eliminarea nodurilor terminale duplicate. Dintre toate nodurile terminale (frunze) duplicate se reţine doar unul, iar spre acesta se redirectează toate arcele care erau directate spre frunzele eliminate.

   Prin această procedură se obţine graful din Figura 2.15-c care are numai 2 noduri terminale în loc de 16.

2. Eliminarea nodurilor neterminale duplicate. Dacă pentru nodurile neterminale x şi y există $var(x) = var(y)$ şi $low(x) = low(y)$, $high(x) = high(y)$, atunci unul din noduri este eliminat iar arcele directate spre nodul eliminat sunt redirectate spre nodul păstrat.

   Inspectând graful din Figura 2.15-c se constată că din cele 8 noduri y_0 doar 3 sunt diferite, deci acesta se poate transforma în graful din Figura 2.15-d.

3. Eliminare testelor redundante. Dacă nodul neterminal x prezintă $low(x) = high(x)$ atunci se elimină nodul iar arcele directate spre acesta se redirectează spre $low(x)$.

Aplicând această procedură pentru eliminarea celor două noduri y_0, din mijloc, de la Figura 2.15-d se transferă testul de redundanţă asupra nodurilor x_0 din mijloc (acestea vor avea $low(x_0) = high(x_0)$). Apoi, aplicând din nou această procedură sunt eliminate nodurile mediane x_0 rezultând graful din Figura 2.15-e. Se observă că nu mai există nici o posibilitate de transformare, nici una din aceste proceduri nu se mai poate aplica.

Definiţia 2.5 Diagrama de decizie binară este **redusă, RBDD** (**R**educed **BDD**), când nu se mai poate realiza nici o transformare prin aplicarea celor trei proceduri de reducere. $\diamond$

O **diagramă de decizie binară** este referită ca **ordonată, OBDD** (**O**rdered **BDD**) dacă, pe oricare traseu parcurs de la rădăcină la frunze, fiecare variabilă este întâlnită cel mult odată şi totdeauna variablilele sunt parcurse în aceeaşi ordine. Nu este necesar ca toate variabilele să fie întâlnite pe fiecare traseu. Pentru funcţia din tabelul de adevăr din Figura 2.15-a, care este a unui CLC comparator pentru cuvinte de doi biţi x_1, x_0 şi y_1, y_0, $f(x_1, x_0, y_1, y_0) = (x_1 = y_1) \cap (x_0 = y_0)$, implicit s-a considerat următoarea ordine $x_1 < y_1 < x_0 < y_0$ şi a rezultat un OBDD cu 8 noduri. La un CLC conparator pentru cuvinte de n biţi păstrând aceeaşi ordine $x_{n-1} < y_{n-1} < x_{n-2} < y_{n-2} < ... < x_1 < y_1 < x_0 < y_0$ rezultă un număr de $(3n + 2)$ noduri. Dar dacă ordinea se stabileşte $x_{n-1} < x_{n-2} < ... < x_1 < x_0 < y_{n-1} < y_{n-2} < ... < y_1 < y_0$ atunci rezultă un OBDD cu $(3 * 2^n - 1)$ noduri. Pentru o astfel de ordonare, $x_1 < x_0 < y_1 < y_0$, $(n = 2)$, BDD este reprezentat în Figura 2.15 - f, numărul de noduri fiind $3 * 2^2 - 1 = 11$. Numărul de noduri pentru un OBDD cu n variabile de intrare, în funcţie de ordonarea aleasă, se situează între limita inferioară, o dependenţă liniară de n, şi limita cea mai dezavantajoasă, o

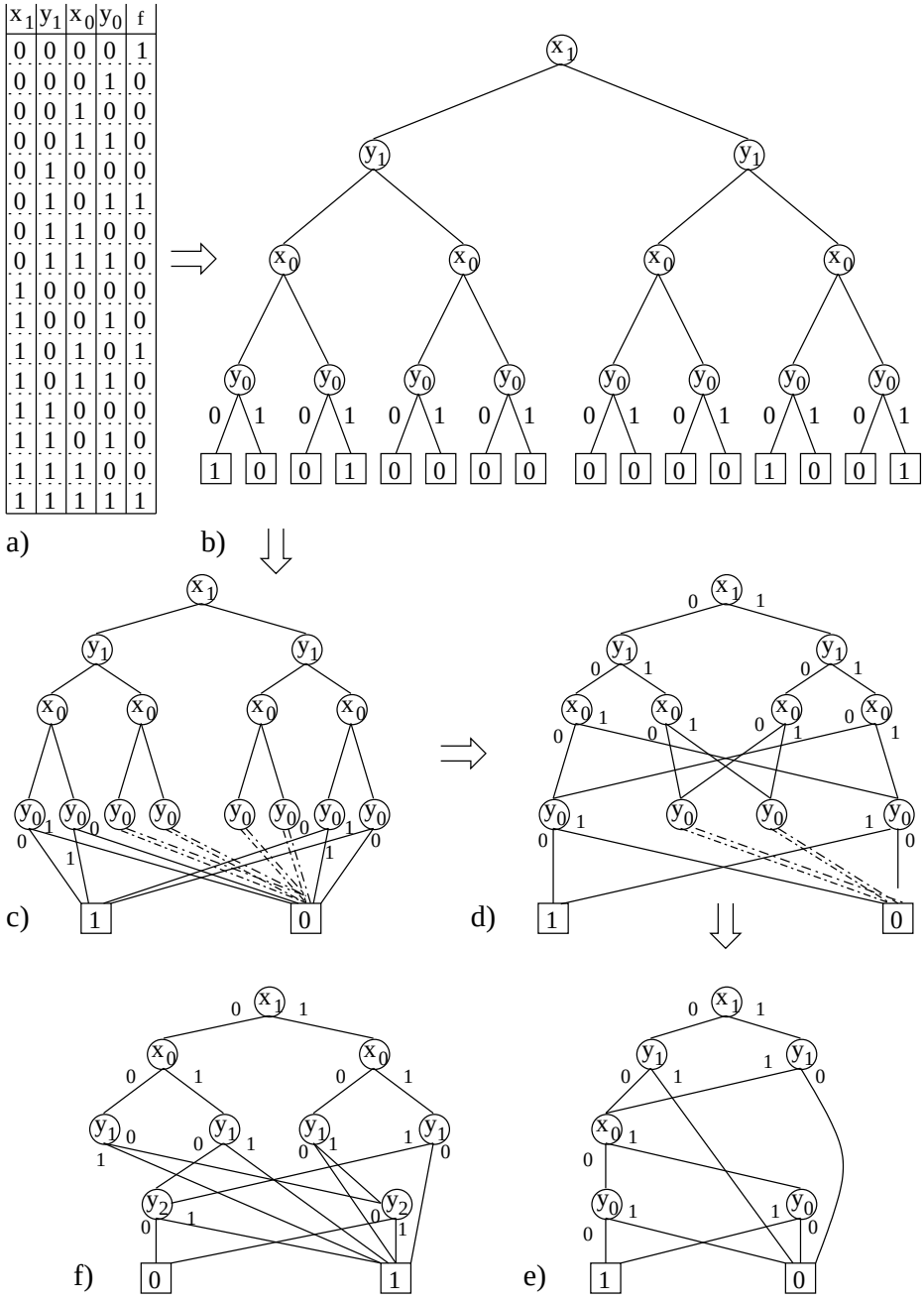


Figura 2.15 Diagrama de decizie binară, BDD, pentru un circuit comparator de 2 biţi: a) tabelul de adevăr; b) arborele de decizie binară, BDT; c,d,e) etapele de trecere de la BDT la diagrama de decizie binară BDD, în forma canonică; f) BDD formă canonică pentru ordonarea variabilelor $x_1 < x_0 < y_1 < y_0$.

dependenţa exponenţială de n. Multe funcţii logice uzuale au totuşi o formă compactă de reprezentare. Găsirea ordonării optime, care să ducă la un număr minim de noduri în OBDD, este o problemă NP completă. Abordările euristice pentru găsirea ordonării optime sunt, de multe ori, cu succes. În acest sens, de abordare euristică, observaţia că variabilele legate/asociate să fie strâns apropiate şi în ordonare duce la o ordonare optimă sau cvasioptimă.

Definiţia 2.6 **BDD** este sub formă **canonică** dacă este redusă şi ordonată. ◇

Pentru o funcţie logică cu ordonare fixată, forma redusă de OBDD este cea canonică şi această formă este unică.

Diagrama de decizie binară considerată ca un suport abstract al unei funcţii booleene introduce posibilitatea ca operaţiile efectuate cu funcţiile booleene să fie implementate sub forma unor algoritmi grafici asupra unor OBDD [Bryant '92] cu aplicaţii în sistemele digitale, logică matematică, inteligenţă artificială. În sistemele digitale OBDD poate fi utilizat ca un instrument în proiectare, verificare şi testare.

2.2.5 Modalităţi neformale de reprezentare

Pentru aplicaţii concrete, de multe ori, definirea unui CLC prin una din metodele formale expuse anterior este foarte greoaie şi ineficientă. Astfel de aplicaţii apar de exemplu, când circuitul are un număr foarte mare de intrări sau circuitul are o structură repetitivă.

Dacă CLC prezintă mai mult de 5,6 variabile, maipularea funcţiei sub formă analitică, diagramă V-K sau tabel de adevăr devine nepractică, în astfel de cazuri doar utilizarea unui program de analiză şi sinteză dintr-un mediu de programare poate fi soluţia.

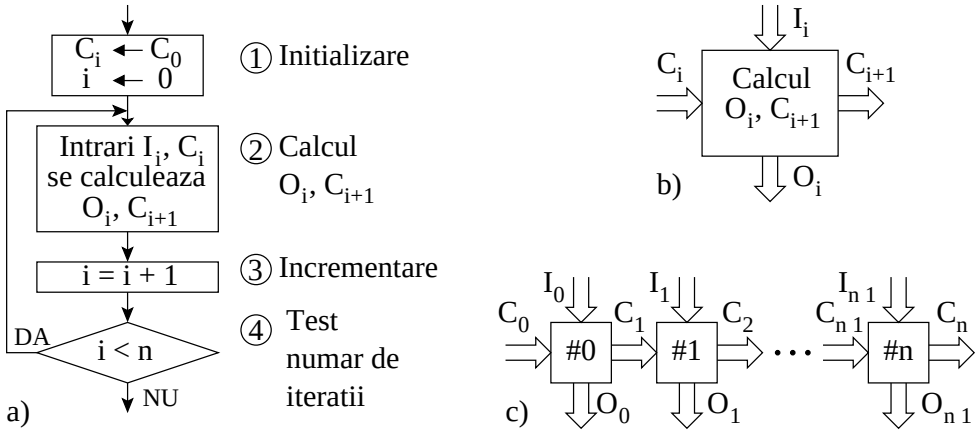


Figura 2.16 CLC iterativ: a) descrierea procesului iterativ al unui circuit prin organigrama unei bucle; b) structura bloc de circuit pentru modelarea calculului din interiorul unei bucle; c) structurarea unui circuit iterativ prin înserierea de celule identice.

Pentru structurile repetitive atenţia trebuie concentrată asupra logicii de funcţionare a celulei componente care apoi, prin replicare, poate forma structura completă.

Un exemplu simplu în acest sens este realizarea circuitelor AND şi OR cu multe intrări. Expresia operatorului sumă sau produs logic pentru multe variabile, folosind axioma asociativităţii, poate fi aplicat repetitiv, obţinându-se o formă care se modelează cu un singur tip de poartă ale cărei intrări sunt ieşiri de la porţi de acelaşi tip. De exemplu, pentru un AND cu n intrări realizat prin înseriere de porţi AND cu două intrări

$$x_{n-1} \cdot x_{n-2} \cdot ... \cdot x_3 \cdot x_2 \cdot x_1 \cdot x_0 = x_{n-1} \cdot (x_{n-2} \cdot (x_{n-3} ... \cdot (x_3 \cdot (x_2 \cdot (x_1 \cdot x_0))...))$$

Acelaşi ANDn poate fi implementat cu AND2 sub formă de arbore binar. Pentru structurile repetitive mai complexe logica de funcţioare a celulei componente conţine operaţii ce se realizează în corpul unei bucle şi care se efectuează de n ori, Figura 2.16-a. În această organigramă pasul de iniţializare (1) se execută o singură dată iar ceilalţi trei paşi se execută de n ori. Pasul (3) de incrementare şi pasul (4) de testare al numărului de iteraţii sunt de fapt "regia buclei", numai pasul (2) realizează practic calculul efectuat de CLC. Pentru pasul (2) se poate concepe o celulă care, pe baza intrărilor I_i, C_i, calculează ieşirea O_i şi transferul C_{i+1} spre celula următoare, Figura 2.16-b. Un CLC iterativ, Figura 2.16-c, practic modelează o desfăşurare a buclei printr-o înseriere de n celule identice, câte una pentru fiecare iteraţie. Cea mai bună cale expunere, a acestor modalităţi de reprezentare, mai puţin formale pentru CLC, este exemplificarea.

Exemplul 2.9 Pentru un cuvânt de n biţi $x_{n-1}x_{n-2}...x_i...x_1x_0$ să se realizeze circuitele combinaţionale care efectuează incrementarea, decrementarea şi complementul faţă de 2.

Soluţie.

1) Incrementarea. Prin adunarea bitului 1 în poziţia cea mai puţin semnificativă, x_0, a cuvântului rezultă totdeauna $\bar{x}_0$, deci circuitul de incrementare are în poziţia x_0 o poartă inversor. Un bit x_i va fi afectat, adică schimbat în în $\bar{x}_i$, de adunarea efectuată în poziţia x_0 numai dacă această adunare a generat un transport şi acest transport s-a propagat pâna în poziţia x_i. Dar transportul se propagă pâna în poziţia x_i doar dacă toţi biţii anteriori lui x_i, începând cu x_0, au valoarea 1. Detectarea şirului de biţi 1 de la x_0 pâna la x_i se realizează printr-un lanţ de porţi AND iar ieşirea acestui lanţ la fiecare poziţie comandă complementarea bitului de la poziţia respectivă, adică se aplică la intrarea unei porţi XOR, Figura 2.17-a. De fapt, incrementatorul poate fi privit ca un circuit numărător în sens direct: un număr se obţine din cel anterior plus 1, incrementatorului îi lipseşte doar componenta care să memoreze numărul anterior (vezi Figura 3.62-b).

2) Decrementarea. Prin scăderea bitului 1 din poziţia cea mai puţin semnificativă x_0 a cuvântului rezultă totdeauna $\bar{x}_0$, deci implementarea pentru ultimul bit se face tot cu o poartă inversor. Dacă un $x_i = 1$ acesta va fi complementat de modificarea în poziţia x_0 doar dacă toţi biţii anteriori, începând cu x_0, au valoarea 0, deoarece numai atunci împrumtul necesar scăderii $x_0 - 1$ se propagă pâna la x_i, schimbând toate zerourile în 1. Detectarea şirului de zerouri se face cu un lanţ de porţi OR iar ieşirea acestui lanţ, la fiecare poziţie, comandă complementarea bitului de la poziţia respectivă aplicându-se la intrarea unei porţi NXOR, Figura 2.17-b. Decrementorul poate fi privit ca un numărător în sens invers.

3) Complementarea faţă de 2. Regula de obţinere a complementului faţă de doi al unui număr negativ este: se complementează toţii biţii, apoi se adună 1.

Se poate obţine complementul faţa de 2 şi prin următorul algoritm: parcurgând cuvântul de la dreapta la stânga se lasă neschimbaţi toţi biţii mai puţin semnificativi pâna la primul bit egal cu 1 inclusiv, apoi toţi biţii care urmează după acest 1 până la cel mai semnificativ se

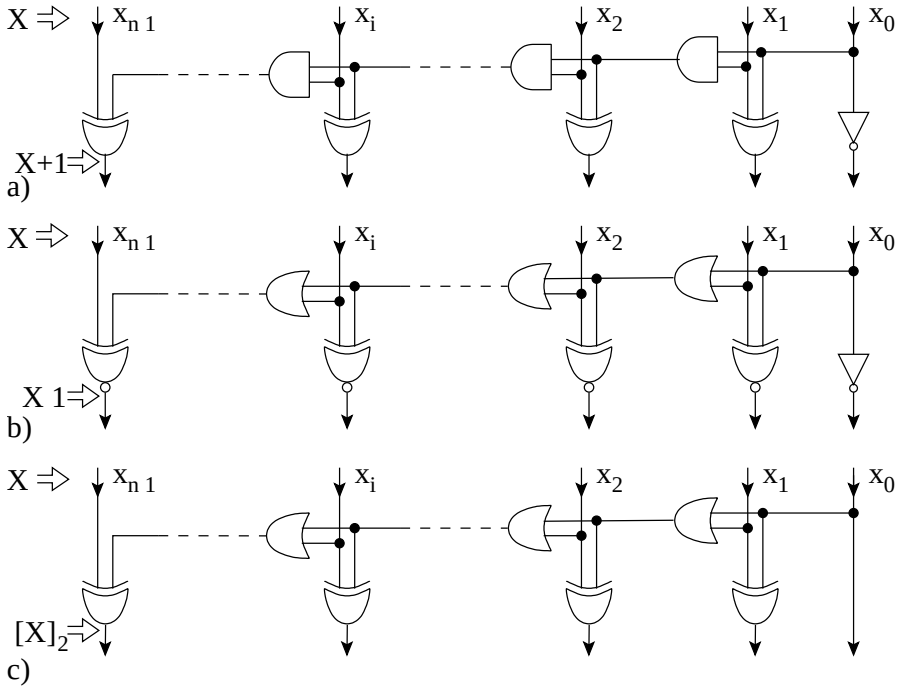


Figura 2.17 Exemple de circuite realizate prin replicare de celule elementare: a) circuitul de incrementare; b) circuitul de decrementare; c) circuitul de complementare faţă de 2.

complementează. De exemplu: $390|_{10} = 01100000110|_2$, ultimii doi biţi rămân neschimbaţi iar ceilalţi se complementează, rezultă că $-390|_{10}$ în complement faţa de 2 este 10011111010.

După acest algoritm implementarea circuitului de complementarea faţă de 2 este directă. Bitul x_0 rămâne neschimbat. Dacă primul bit 1 este în poziţia x_i înseamnă că biţii din primul interval, de la x_0 până la x_i, rămân neschimbaţi iar cei din al doilea interval, de la x_{i+1} pâna la x_{n-1}, se complementează. Fiecare bit al cuvântului este o intrare la o poartă XOR iar pe cealaltă intrare a porţii se aplică 0 pentru biţii care nu se complementează, din primul interval, si respectiv 1 pentru biţii care se complementează, din al doilea interval. Porţile XOR sunt comandate de ieşirile unui lanţ de porţi care au pe ieşiri valoarea 1 doar după primul bit 1 al cuvântului, deci această selectare se poate realiza cu un laţ de porţi OR, Figura 2.17-c.

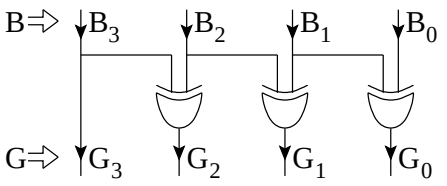
Exemplul 2.10 Pentru numerele naturale exprimate în binar natural, cu lungimea de patru biţi $B_3B_2B_1B_0$, să se realizeze circuitul combinaţional care efectuează conversia în cod Gray $G_3G_2G_1G_0$ şi apoi circuitul care realizează conversia $G_3G_2G_1G_0$ în $B_3B_2B_1B_0$.

Soluţie. Descrierea convertorului de cod binar natural - Gray ca un circuit cu patru ieşiri G_3, G_2, G_1, G_0 şi patru intrări B_3, B_2, B_1, B_0 este dată în tabelul de adevăr din Figura 2.18-a. Acelaşi tabel de adevăr descrie şi conversia Gray - binar natural dacă se consideră G_3, G_2, G_1, G_0 intrări şi B_3, B_2, B_1, B_0 ieşiri. Trebuie specificate diferenţele între tabelele de adevăr din Figura 2.3-a şi Figura 2.18-a. Primul reprezintă conversia BCD - Gray, deci

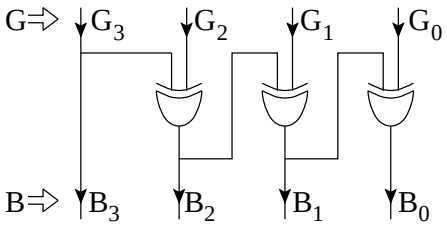
B_3	B_2	B_1	B_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

G_3	G_2	G_1	G_0
0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	1	1
0	1	0	1
0	1	0	0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	1	0
1	0	1	1
1	0	0	1
1	0	0	0

a)



b)



c)

Figura 2.18 Conversia binar natural - Gray şi Gray - binar natural: a) tabel de adevăr pentru expresie; b,c) circuitele de conversie structurate repetitiv pe bază de porţi XOR.

cele 6 combinaţii (1010, 1011, 1100, 1101, 1110, 1111) nu se aplică niciodată pe intrarea convertorului, pe când în al doilea tabel de adevăr conversia binar natural - Gray cele şase combinaţii nu mai sunt indiferente pentru circuitul convertor. Utilizarea tabelelor de adevăr cu 16 linii nu este prea atrăgătoare, în schimb, regulile de conversie expuse în paragraful 2.2.1 şi reprezentate grafic în Figura 2.2-a şi b sunt mult mai practice şi mai uşor de implementat. Uşor de implementat, deoarece pentru ambele conversii se utilizează sumarea a câte doi biţi iar transportul se neglijează ceea ce se realizează cu o poartă XOR. Prin mapare directă, utilizând porţi XOR pentru conversia binar - Gray din Figura 2.2-a rezultă circuitu din Figura 2.18-b, iar pentru conversia Gray - binar din Figura 2.2-b rezultă circuitul din Figura 2.18-c.

Evident, structura circuitului din Figura 2.18-a şi 2.18-b indică şi o exprimare iterativă pentru aceste coversii. Aceste relaţii iterative se pot deduce uşor dacă se notează cuvântul în binar natural prin $B_{n-1}B_{n-2}...B_i...B_2B_1B_0$ iar cuvântul în cod Gray prin $G_{n-1}G_{n-2}$ $...G_i...G_2G_1G_0$. Relaţiile pentru conversii sunt:

$$\begin{array}{l}\text{Binar natural} \rightarrow \text{Gray} \ : \ G_{n-1} = B_{n-1}, \ G_i = B_{i+1} \oplus B_i \ pentru \ i = 0, 1, ...n-2; \\ \text{Gray} \rightarrow \text{Binar natural} \ : \ B_{n-1} = G_{n-1}, \ B_i = B_{i+1} \oplus G_i \ pentru \ i = 0, 1, ...n-2.\end{array} \quad (2.8)$$

Exemplul 2.11 Să se structureze un circuit pentru detectarea şi generarea parităţii unui cuvânt.

Soluţie. Paritatea unui cuvânt de n biţi referă numărul de biţi 1 din acel cuvânt. **Paritatea este pară sau impară** după cum numărul de biţi 1 ai cuvântului este par sau impar. Paritatea poate fi utilizată ca un parametru în determinarea dacă un cuvânt a fost modificat/eronat în urma transmisiei pe o linie de comunicaţie. Se va explica acest procedeu

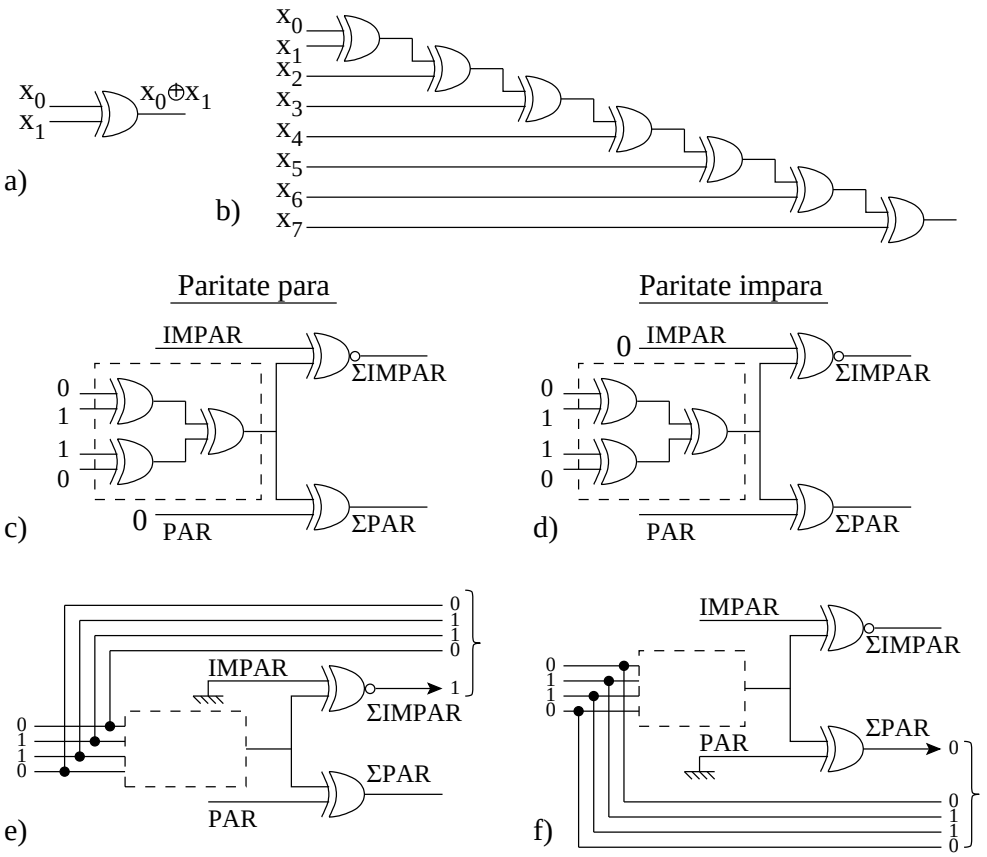


Figura 2.19 Detectarea şi generarea parităţii: a,b) circuite pentru determinarea parităţii unui cuvânt de doi biţi respectiv de 8 biţi; c,d) structură sub formă de arbore din porţi XOR pentru detectarea parităţii pare respectiv impare a unui cuvânt de patru biţi; e,f) structuri pentru generarea de cuvinte de 5 biţi cu paritate impară respectiv pară.

la transmisia serială a unui cuvânt în cod ASCII-7 (American Standard Code for Information Interchange, lungimea de cuvânt de 7 biţi), vezi Exemplul 3.28.

La emisie, înainte de transmisie, se determină paritatea cuvântului de 7 biţi prin numărarea biţilor 1. Când transmisia se face cu paritate pară se adaugă al optulea bit, **bitul de paritate**, de valoare 1 dacă paritatea determinată a fost impară şi se adaugă al optulea bit de valoare 0 dacă paritatea determinată a fost pară. Iar pentru transmisia cu paritate impară se adaugă al optulea bit de valoare 0 dacă paritatea determinată a fost impară respectiv se adaugă 1 dacă paritatea determinată a fost pară.

La recepţie, la o transmisie cu paritate pară, dacă se detectează paritate pară înseamnă că în cuvântul transmis nu s-a modificat nici un bit (deci transmisie corectă), ori s-a modificat un număr par de biţi 1 (transmisie incorectă), iar dacă paritatea detectată este impară transmisia este incorectă. Pentru transmisie cu paritate impară dacă paritatea detectată este impară în cuvântul transmis, nu s-a modificat nici un bit cu valoarea 1 (transmisie corectă), ori s-au modificat un număr par de biţi (transmisie incorectă) iar dacă paritatea

detectată este pară transmisiunea este incorectă. Rezultă că în utilizarea parităţii pentru determinarea corectitudinii transmisiei se consideră implicit că se modifică doar un număr impar de biţi; alte metode pot fi utilizate pentru determinarea corectitudinii transmisiunii şi pentru cazurile când se modifică orice număr de biţi şi, chiar mai mult, pot să şi corecteze eroarea produsă în cuvântul recepţionat (metode detectoare - corectoare).

Sinteza unui detector de paritate, pentru un cuvânt în cod ASCII-7, la emisie ar necesita un tabel cu $2^7 = 128$ linii, iar pentru recepţie cu $2^8 = 256$ linii. Mai mult, deoarece configuraţiile binare de intrare în succesiunea lor alternează par-impar-par..., ar rezulta, pentru sinteza circuitului, o diagramă V-K cu 128 respectiv 256 căsuţe elementare dar cu o umplere de 1-uri în formă de tablă de şah, deci imposibil de minimizat. Acest mod de abordare ar necesita la recepţie o implementare cu 128 porţi AND cu 8 intrări şi o poartă OR cu 128 de intrări! Trebuie găsită o altă modalitate de exprimare şi deci de implementare.

Suma aritmetică, XOR, a unui număr par de biţi 1 dintr-un cuvânt este totdeauna zero şi este totdeauna unu pentru un număr impar de biţi 1. Deci elementul repetitiv în detectarea tipului de paritate este poarta XOR (pentru un cuvânt de doi biţi o singură poartă Figura 2.19-a, iar pentru un cuvânt de opt biţi un lanţ de şapte porţi XOR ca în Figura 2.19-b. Descrierea unei astfel de structuri repetitive rezultă prin aplicarea repetitivă a axiomei asociativităţii operatorului XOR pentru cuvântul de opt biţi:

$$(((((((x_0 \oplus x_1) \oplus x_2) \oplus x_3) \oplus x_4) \oplus x_5) \oplus x_6) \oplus x_7).$$

Dezavantajul acestei implementări rezidă în înserierea a şapte niveluri de porţi XOR. Se poate obţine o implementare cu o propagare numai pe trei niveluri de porţi XOR dacă se aplică asociativitatea întâi pe grupuri de 2 intrări apoi pe grupuri de câte 4 şi de câte 8, rezultând o structură de arbore binar.

$$(((x_0 \oplus x_1) \oplus (x_2 \oplus x_3)) \oplus ((x_4 \oplus x_5) \oplus (x_6 \oplus x_7))).$$

Se va exemplifica, cu o structură sub formă de arbore, pentru detectarea par/impar la un cuvânt de 4 biţi care are pe ieşire o poartă XOR pentru semnalarea de paritate pară $\sum PAR = 0$ şi respectiv o poartă NXOR pentru semnalarea de paritate impară $\sum IMPAR = 0$ (s-a ales un NXOR pentru ca detectarea parităţii impare să fie semnalată tot prin valoarea logică zero, ca şi la paritatea pară). (Încercaţi o structurare pentru 8 intrări). În Figura 2.19-c acest circuit este utilizat pentru verificarea parităţii pare (pentru $PAR = 0,\ IMPAR = -$) şi generează, de exemplu, pentru cuvântul de intrare 0110 la ieşire semnalul activ de paritate pară $\sum PAR = 0$. Acelaşi circuit dar utilizat pentru verificare de paritate impară (pentru $PAR = -,\ IMPAR = 0$) este prezentat în Figura 2.19-d, unde pentru acelaşi cuvânt de intrare 0110 (par) generează un semnal fals de paritate impară $\sum IMPAR = 1$.

Ca generator de paritate, acelaşi circuit este utilizat pentru a produce un cuvânt de 5 biţi cu paritate impară (pentru $IMPAR = 0,\ PAR = -$) în Figura 2.19-e, iar pentru a produce paritate pară (pentru $PAR = 0,\ IMPAR = -$) în Figura 2.19-f. Bitul generat pe ieşirile $\sum IMPAR$ sau $\sum PAR$ se adaugă ca al cincelea bit pentru a realiza paritatea impară respectiv pară a cuvântului de 5 biţi care se transmite.

Se observă că circuitul pentru detectarea parităţii pare este utilizat şi la generarea parităţii pare, de asemenea circuitul pentru detectarea parităţii impare este utilizat şi la generarea parităţii impare.

2.3 REALIZAREA CIRCUITELOR COMBINAŢIONALE

Pentru realizarea unui CLC, in general, se parcurg următoarele etape:

1. Formularea/exprimarea, în termeni cât mai precişi, a problemei care trebuie rezolvată.

2. Pe baza formulării problemei se construieşte tabelul de adevăr care stabileşte relaţia dintre variabilele de intrare şi cele de ieşire. Atenţie trebuie acordată nivelului logic asignat pentru starea activă a fiecărei variabile de intrare şi pentru ieşiri (un semnal X activ în stare L se notează X_L sau uneori cu $\overline{X}$).

3. Se utilizează o metodă de minimizare (diagrama V-K, sau o metodă algoritmică sub forma unui program) pentru a obţine o formă redusă/minimă ca o funcţie disjunctivă, FD, sumă de produse sau ca o funcţie conjunctivă, FC, produse de sume (vezi secţiunea 1.1.5).

4. Implementarea circuitului. Forma FC sau FD se potriveşte astfel încât implementarea să fie realizată cu tipurile de poartă impusă/disponibilă şi în funcţie de tehnologia utilizată. Se desenează structura circuitului.

5. Se testează circuitul şi se elaborează documentaţia.

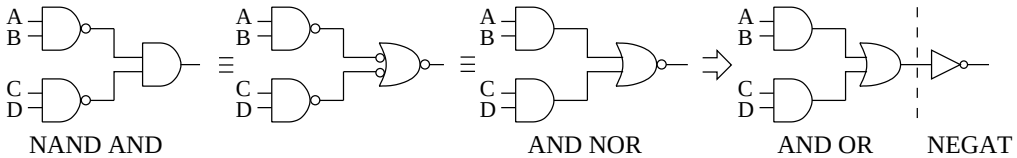
Funcţiile FC sau FD obţinute la punctul 3 pot fi implementate pe două niveluri logice. Aceasta presupune, implicit, că la intrare sunt disponibile variabilele atât sub formă negată cât şi sub formă nenegată. Această presupunere se bazează pe faptul că variabilele sunt memorate cu registre a căror ieşiri sunt disponibile atât negate cât şi nenegate.

Teoretic, se pune întrebarea, utilizând cele patru tipuri de porţi uzuale AND, OR, NAND, NOR, câte combinaţii de implementare pe două niveluri sunt posibile? Implementarea pe două niveluri are mai multe porţi de acelaşi tip pe primul nivel iar pe al doilea nivel o singură poartă; nu este restricţionată utilizarea aceluiaşi tip de poartă pe ambele niveluri. Rezultă în total 16 combinaţii de implementare pe două niveluri, dar dintre acestea opt variante sunt degenerate (AND-AND, AND-NAND, OR-OR, OR-NOR, NAND-OR, NAND-NOR, NOR-AND, NOR-NAND). Variantele degenerate se reduc la un singur operator, de exemplu porţi AND în primul nivel şi poartă AND în al doilea nivel produce o ieşire care este un AND de toate variabilele de intrare. Celelalte opt variante nedegenerate sunt:

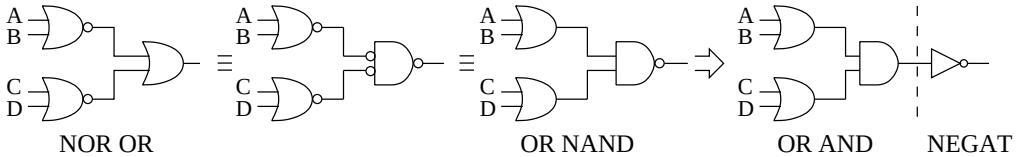
AND-OR	OR-AND
NAND-NAND	NOR-NOR
NOR-OR	NAND-AND
OR-NAND	AND-NOR

Formele situate pe acelaşi rând sunt duale (Definiţia 1.2). Variantele AND-OR şi OR-AND sunt cele de bază şi corespund implementării directe a formelor FD, sumă de produse, repsectiv FC, produs de sume. Variantele cu un singur tip de poartă NAND-NAND şi NOR-NOR se obţin prin transformări respectiv a variantelor de bază AND-OR şi OR-AND. Conversia AND-OR în NAND-NAND apare ca o transformare

naturală dar nu la fel de naturală apare conversia în NOR-NOR, Figura 1.5-a,b. De asemenea, conversia OR-AND în NOR-NOR apare ca o transformare naturală dar mai puţin naturală apare conversia în NAND-NAND, Figura 1.5-c,d.



a)



b)

Figura 2.20 Echivalenţa formelor de impelmentare pe 2 niveluri: a)NAND-AND cu AND-NOR (≡echivalent AND-OR-NEGAT) ; b) NOR-OR cu OR-NAND (≡ OR-AND-NEGAT).

Se poate demonstra că variantele de implementare NAND-AND şi AND-NOR sunt echivalente: $(\overline{AB})(\overline{CD}) = \overline{\overline{(\overline{AB})(\overline{CD})}} = \overline{(AB)+(CD)}$. În plus, aceste doua variante se obţin din varianta de bază AND-OR, sumă de produse, prin negare adică, AND-OR-NEGAT Figura 2.20-a. Varianta AND-OR de implementare este a unei funcţii f care se obţine ca sumă de produse prin acoperirea tuturor căsuţelor elementare cu 1 din diagrama V-K, pe când varianta AND-OR-NEGAT de implementare este a funcţiei negate, $\bar{f}$, care se obţine ca sumă de produse prin acoperirea tuturor căsuţelor elementare cu 0 din diagrama V-K. Dacă din diagrama V-K forma $\bar{f}$ se obţine mai uşor, atunci aceasta se extrage şi apoi prin complementare produce pe f.

De asemenea,există echivalenţă între NOR-OR şi OR-NAND:$(\overline{A+B})+(\overline{C+D}) = \overline{\overline{(\overline{A+B})+(\overline{C+D})}} = \overline{(A+B)(C+D)}$ şi care se obţin din forma de bază OR-AND, produs de sume negat, adică OR-AND-NEGAT, Figura 2.20-b. Dacă extragerea funcţiei f ca produs de sume din diagrama V-K, prin selectarea suprafeţelor cu zero, este o operaţie mai complicată decât extragerea funcţiei $\bar{f}$ ca produs de sume din diagrama V-K prin selectarea suprafeţlor cu unu, atunci se procedează pentru obţinerea lui $\bar{f}$ care, apoi, prin negare generează pe f.

Definiţia 2.7 Pentru un CLC cu n intrări se notează **adâncimea D(n)**, care este egală cu numărul maxim de niveluri logice (porţi) prin care se propagă cel puţin unul dintre semnalele de la intrare până la ieşire. ◇

Adâncimea minimă obţinută pentru implementarea unei funcţii FC sau FD este egală cu 2. Dintre variantele nedegenerate de implementare pe două niveluri sunt recomandate NAND-NAND şi NOR-NOR deoarece:

1. introduc o uniformitate structurală prin utilizarea aceluiaşi tip de poartă, deci facilităţi tehnologice.

2. elimină efectele de propagare. La frecvenţe ridicate, când durata/lăţimea impulsurilor este redusă, transferul unui impuls printr-un lanţ de de porţi neinversoare poate duce la dispariţia acestuia (reducerea lăţimii impulsului după propagarea prin fiecare nivel logic), ceea ce nu se întâmplă la tranferul impulsului printr-un lanţ de porţi inversoare. Un astfel de fenomen se explică prin faptul că, în general, $\tau_{pLH} > \tau_{pHL}$, în consecinţă, la transferul prin fiecare poartă neinversoare lăţimea inpulsului se îngustează cu duarata $\Delta = \tau_{pLH} - \tau_{pHL}$, pe când la transferul printr-un lanţ de porţi inversoare, lăţimea impulsului se reface după fiecare 2 niveluri logice.

Adâncimea D(n) a unui circuit reflectă timpul de propagare care este egal cu durata de propagare printr-o poartă, τ_p, înmulţit cu numărul n de niveluri logice, deci determină performanţa, frecvenţa de lucru a circuitului $f = 1/n \cdot \tau_p$. În acest sens, rezultă că implementările pe două niveluri sunt cele recomandate deoarece determină un timp minim de propagare, $2\tau_p$, prin circuit. Totuşi, acestă recomandare trebuie analizată critic în funcţie de tehnologia de implementare.

Pentru realizarea CLC pe suport de circuit imprimat şi porţi logice discrete implementarea funcţiei pe două niveluri logice este optimă. Este corectă această afirmaţie mai ales când porţile sunt în tehnologie bipolară, la care timpii de propagare sunt mai puţin sensibili la valoarea de încărcare (fan-out). Nu aceeaşi valabilitate o are această afirmaţie pentru implementările integrate în tehnologie CMOS.

În tehnologie CMOS, când întârzierea pe poartă este pronunţat afectată de încărcarea porţii, implementarea pe două niveluri logice s-ar putea să nu fie cea optimă pentru opţinerea unui circuit de viteză ridicată. Alegerea numărului optim $\hat{N}$ de niveluri logice pe un anumit traseu în funcţie de încărcarea traseului este prezentată în secţiunea 1.5.6.2, Tabelul 1.15. Obţinerea unei expresii pentru o implementare multinivel, dintr-o formă redusă dinsjunctivă, se realizează prin utilizarea axiomei distributivităţii. De exemplu, expresia $ACD + AB + B\overline{C}$, implementabilă pe două niveluri AND-OR ca în Figura 2.21-a, prin factorizarea variabilei A va fi implementabilă pe patru niveluri: două porţi AND pe primul nivel, o poartă OR pe nivelul doi, o poartă AND pe nivelul trei şi o poartă OR pe nivelul patru. Dar această structură poate fi transformată pentru o implementare, tot pe patru niveluri, numai cu porţi NAND ca în Figura 2.21-b. Formele de scriere care exprimă aceste implementări sunt:

$$ACD + AB + B\overline{C} = A(CD + B) + B\overline{C} = \overline{\overline{(\overline{\overline{CD}} \cdot B)A}}\,\overline{(B\overline{C})}$$

Prin scalare, (Tabelul 1.11) timpul de propagare pe poartă se micşorează dar datorită faptului că la aceleaşi dimensiuni ale cipului majoritatea traseelor de comunicaţie/ interconectare rămân la aceleaşi lungimi, acestea determină ca o serie de alţi parametrii ai circuitului (căderea de tensiune, densitatea de curent, rezistenţele de contact, timpul de propagare pe linie) să fie afectaţi de degradări (vezi secţiunea 4.5). În plus, diminuarea puterii disipate pe poartă determină o abilitate micşorată a porţii pentru comanda capacităţii liinilor. În asemenea circumstanţe, când scalarea are influenţă mult mai mare asupra timpilor de propagare prin porţi decât asupra timpilor de propagare prin conexiuni, întârzierea medie pe nivel logic este mai puternic determinată de interconexiuni decât de porţi, în consecinţă, un număr redus de porţi înseriate nu determină automat şi performantă de viteză ridicată pentru circuit.

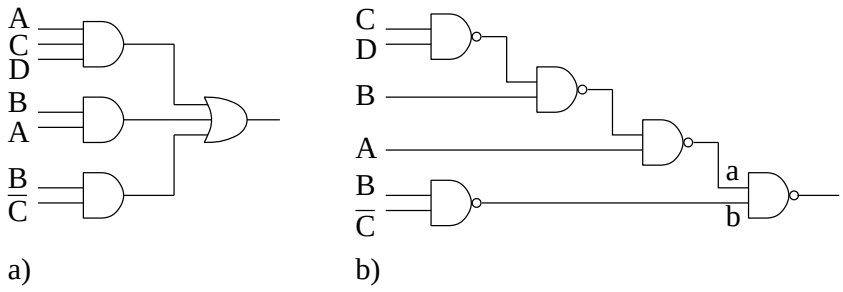


Figura 2.21 Implementarea expresiei $ACD + AB + B\overline{C}$ pe două niveluri logice (a) şi pe patru niveluri logice (b).

Un CLC cu n intrări, pe lângă adâncimea $D(n)$, mai trebuie caracterizat şi prin efortul structural necesar pentru realizarea sa, adică prin dimensiunea sa, notată cu $S(n)$. Dar, acest efort trebuie exprimat printr-un număr ce reflectă şi este specific unei tehnologii de implementare. De exemplu, pentru realizare pe placă de sticlotextolit cu circuite integrate discrete, acel număr ar putea fi: numărul de puncte de wrapping ori de lipit, numărul de circuite integrate sau numărul tuturor intrărilor în circuitele integrate. Dacă se realizează integrat, acel număr ar putea fi: numărul total de porţi, numărul tuturor intrărilor în porţile circuitului sau suprafaţa de siliciu consumată.

Definiţia 2.8 Dimensiunea S(n), asociată unui circuit cu n intrări, se exprimă prin numărul de intrări al tuturor circuitelor integrate din care se configurează acel circuit. În general, $S(n)$ se exprimă ca ordin de mărime. $\diamond$

Conform acestei definiţii, toate circuitele cu patru intrări ($n = 4$) pe două niveluri din Figura 2.20 au $S(4) = 6$ iar cele pe trei niveluri au $S(n) = 7$, iar circuitele din Figura 2.12 au $S(n) = 1 + 2 + (n-2) \cdot 4 = 4 \cdot n - 5 \in O(n)$

Semnalăm corelaţia care se poate face între dimensiunea $S(n)$, dată prin Definiţia 2.8 şi efortul logic exprimat prin Definiţiile 1.16, 1.17 şi 1.18. Conform Definiţiei 2.8 , o poartă cu cât are mai multe intrări, cu atât dimensiunea sa este mai mare şi, la fel, un CLC cu cât este compus din mai multe porţi, şi acestea au dimensiunea mai mare, cu atât dimensiunea rezultată va fi mai mare. Similar, o poartă logică CMOS cu cât are mai multe intrări, are un efort logic $g(n)$ mai mare, relaţiile de calcul din Tabelul 1.13 exprimă clar această dependenţă. De asemenea, cu cât un traseu în tehnologie CMOS este realizat din mai multe niveluri, şi fiecare nivel are efortul logic mai mare, cu atât şi efortul logic G al traseului este mai ridicat. Deci, pentru implementările CMOS s-ar purtea utiliza efortul logic ca o măsură a dimensiunii circuitului.

Între cele două mărimi $S(n)$ şi $D(n)$, care caracterizează un CLC, există o interdependenţă, vizibilă şi intuită de oricare proiectant, dar care nu este exact definită. Practic, se constată că dacă se caută pentru CLC o variantă de implementare mai rapidă, deci cu un $D(n)$ mai mic, se constată că este necesar să se mărească dimensiunea $S(n)$. În [Stefan '00] se propune o formalizare a acestei interdepentenţe: fie varianta1 a a unui circuit cu n intrări care realizează funcţia f caracterizată prin $S_1(n)$ şi $D_1(n)$. Dacă varianta2 a acestui circuit este caracterizată prin $D_2(n) < D_1(n)$ atunci, relaţia între produsele $S(n) \cdot D(n)$, ale celor două variante, este corectă

$$S_2(n) \cdot D_2(n) > S_1(n) \cdot D_1(n) \tag{2.9}$$

adică, mărirea performanţei circuitului (micşorarea adâncimii) cu un anumit factor (de performaţă) implică creşterea dimensiunii circuitului de mai multe ori decât valoarea acestui factor de preformaţă (viteză).

Pentru compararea variantelor de realizare a unui CLC trebuie să existe un parametru sintetic, acesta poate fi exprimat prin **raportul cost/performanţă** (care exprimă cât costa unitatea de performanţă). Costul este determinat de dimensiunea $S(n)$ exprimată nu neapărat în funcţie de numărul de terminale, ca în Definiţia 2.8 , ci de oricare caracteristică ce poate determina dimensiunea. În plus, pe lângă $S(n)$, costul este determinat şi de **complexitate** $C(n)$, definiţia 2.1 . Un circuit chiar de dimensiune mare dar simplu (structură ordonată, repetitiv) poate fi realizat la un cost rezonabil, nu la fel se poate afirma despre costul unui circuit de dimensiune ridicată şi complex (structură întâmplătoare - **random logic**).

Un alt aspect urmărit în realizarea unui CLC este de ordin calitativ, adică o funcţionare corectă, ceea ce implică eliminarea situaţiilor de hazard. De asemenea, uneori, se impune ca circuitul să fie reconfigurabil, adică adaptabil prin modificarea structurării pentru mai multe aplicaţii sau pentru aceeaşi aplicaţie dar în diferite variante. **Reconfigurabilitatea** a devenit un concept curent în sistemele digitale actuale.

2.3.1 Hazardul static

Analiza circuitelor combinaţionale s-a efectuat pâna acum considerănd **regimul static** (intrările şi ieşirile nu au variaţii în timp), pentru aceasta se presupune că din momentul de modificare a valorilor variabilelor de intrare a trecut deja un timp mult mai lung decât timpul de propagare prin circuit (regimul tranzitoriu), deci ieşirea este stabilizată. În regim static, valoarea logică a ieşirii se calculează corect cu funcţia logică a circuitului. Dar, ce se întâmplă în intervalele tranzitorii, pe intervalele de timp când semnalele sunt în propagare de la intrare la ieşire? La aplicarea unei configuraţii pe intrare, de multe ori, valoarea logică reală (obţinută la ieşire pe durata regimului tranzitoriu) nu este identică cu valoarea logică la ieşire calculată cu funcţia logică a circuitului; valoarea logică la ieşire devine egală cu cea calculată numai după consumarea regimului tranzitoriu. Aceste situaţii când CLC, care trebuie să aibă un comportament controlabil, prezintă o funcţionare "hazardată" (necontrolată) fată de analiza în regim static sunt referite cu termenul de hazard static. Fizic, hazardul static se manifestă prin apariţia în semnalul de ieşire a unor **impulsuri parazite (glitch-uri)** fie cu nivel logic H fie cu nivel liogic L. Posibilitatea de producere a glitch-urilor pe durata regimurilor tranzitorii apare ca o consecinţă a două cauze: 1) comutarea asincronă a valorilor variabilelor de intrare (hazardul datorat asincronismul la intrare); 2) existenţa pentru o variabilă de intrare a două trasee de propagare cu întârzieri diferite (hazardul de propagare).

1) Hazardul datorită asincronismului la intrare. Trecerea de la o valoare logică a ieşirii la o altă valoare logică este cauzată de modificarea (comutarea) configuraţiei cuvântului de intrare. Implicit, s-a considerat, până acum, că această comutare a tuturor variabilelor de intrare de la o anumită configuraţie la o alta se face în acelaşi moment de timp, se realizează o comutare sincronă. În realitate, când se trece de la o configuraţie a variabilelor de intrare la o alta există un decalaj între momentele de comutare ale diferitelor variabile, poate numai întâmplător există o

comutare sincronă. De fapt, acest asincronism face ca între configuraţia de intrare prezentă şi configuraţia de intrare următoare să se interpună, pe durata regimului tranzitoriu, una sau mai multe configuraţii de intrare care produc glitch-uri în semnalul de ieşire.

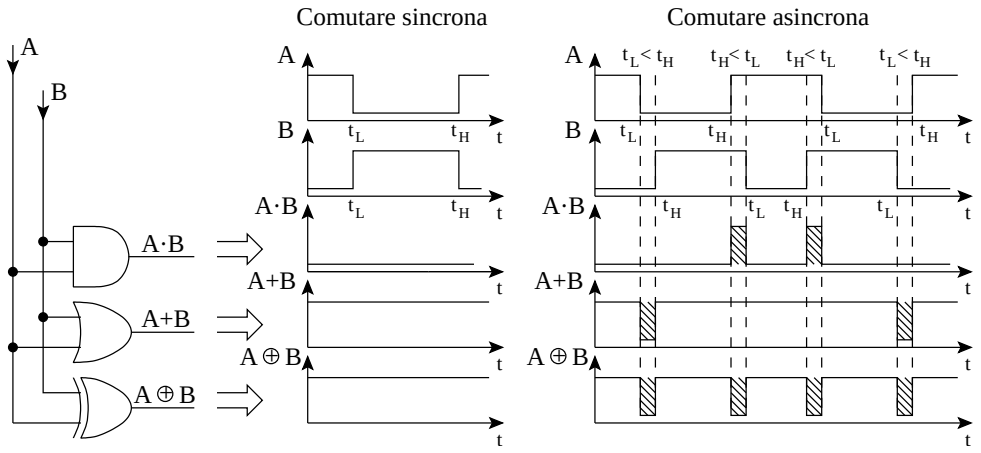


Figura 2.22 Explicativă pentru apariţia hazardului datorită asincronismului la intrările porţilor AND,OR şi XOR

Apariţia hazardului datorită asincronismului intrărilor este exemplificat în Figura 2.22 pentru porţile AND,OR,XOR cu două intrări. Considerând o comutare spre valori opuse ale intrărilor A, B s-a desenat variaţia semnalului de ieşire atât pentru comutare sincronă cât şi pentru comutare asincronă. La comutare asincronă, când tranziţia negativă precede pe cea pozitivă, $t_L < t_H$,, sau când tranziţia pozitivă precede pe cea negativă, $t_H < t_L$, semnalul de ieşire din poartă poate fi încărcat cu glitch-uri (hazard). Acest semnal de ieşire, dacă va fi utilizat ca semnal de intrare pentru un circuit următor, va produce o funcţionare eronată. Se poate evita hazardul de nesincronizare dacă se impune ca în configuraţia de intrare nciodată să nu comute mai mult de o singură variabilă sau înainte de aplicarea configuraţiei pe intrările circuitului aceasta să fie sincronizată (vezi Figura 3.46).

2) Hazardul de propagare. Dacă la o poartă din interiorul circuitului, sau din nivelul de ieşire al circuitului, semnalele aplicate la intrarea circuitului ajung pe trasee diferite, care implică întârzieri diferite, atunci la ieşirea acelei porţi poate apare hazard. De fapt, la poarta generatoare de hazard semnalele se aplică la intervale de timp diferite ceea ce, de fapt, reduce hazardul de propagare tot la un hazard de asincronism, numai că de data aceasta asincronismul nu este la intrările circuitului ci la intrările unei porţi din circuit, adică în interiorul circuitului. De exemplu, la circuitul pe patru niveluri din Figura 2.21-b până la intrarea porţii NAND de pe ultimul nivel variabilele parcurg: A -un nivel logic, B -pe un traseu două niveluri logice şi pe alt traseu un nivel logic, C -trei niveluri logice, $\bar{C}$ un nivel logic, D - trei niveluri logice. Rezultă că chiar dacă configuraţia de intrare se aplică sincron, la intrarea ultimei porţi, cele două semnale A şi B vor avea o variaţie hazardată determinată de timpii de propagare ai porţilor de pe traseele parcurse. Calculul exact al întârzierilor pe fiecare traseu întâmpină dificultăţi deoarece ca dată de catalog este

dat doar timpul de propagare maxim nu şi cel minim, care, în general, un este identic la toate porţile (depinde de dispersia tehnologică) şi este funcţie şi de incărcarea pe ieşirea porţii.

Dar chiar şi la implementările pe două niveluri se poate genera hazard de propagare în cazurile când unele variabile negate sunt generate în interiorul circuitului (de fapt negarea în interiorul circuitului introduce pentru unele semnale al treilea nivel de propagare). În aceste cazuri rezută că implementările de tipul AND-OR, NAND-NAND produc **hazard static 1**, adică pe durata regimuluii tranzitoriu, chiar când doar o singură variabilă de intrare comută, în semnalul de ieşire care ar trebui să fie 1 apare un glitch 0; de asemenea, implementările de tipul OR-AND, NOR-NOR produc **hazard static 0**, adică pe durata regimului tranzitoriu în semnalul de ieşire care ar trebui să fie 0 apare un glitch 1.

Teorema 2.1 Dacă expresia unei funcţii poate fi adusă la forma

$$f(x_{n-1}, x_{n-2}, ..., x_i, ..., x_1, x_0) = x_i + \bar{x}_i$$

atunci, într-un circuit implementat cu porţi logice, se va genera hazard static 1 la comutarea variabilei de intrare x_i din 1 în 0 (pentru valori precizate ale celorlalte variabile de intrare).◇

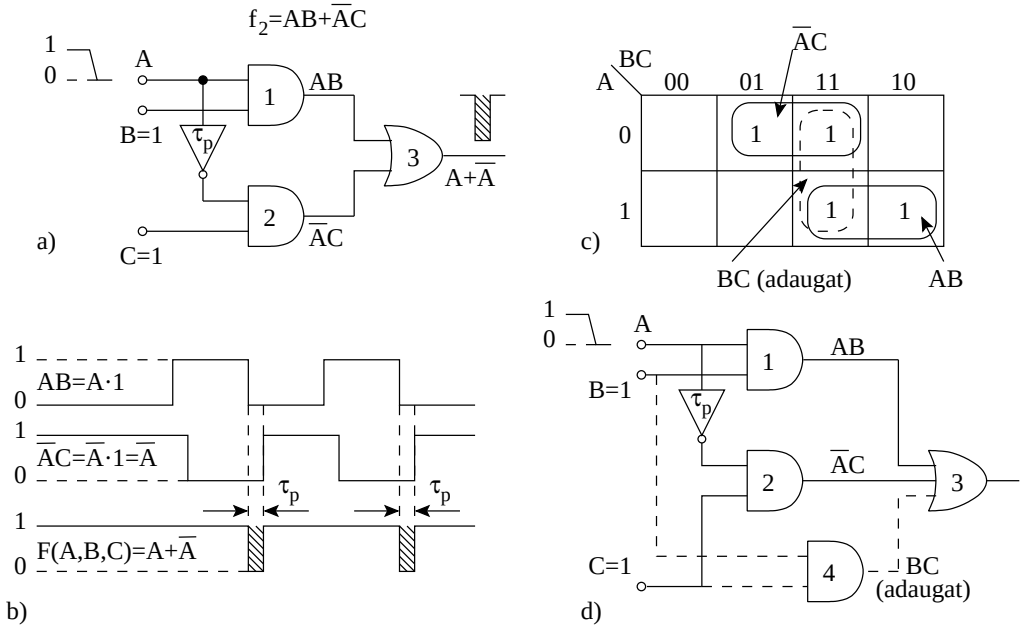


Figura 2.23 Analiza apariţiei hazardului static 1 pentru funcţia $\mathbf{f_2 = AB + \overline{A}C}$: a) structura circuitului; b,c) analiza apariţiei hazardului pe diagrama de semnale respectiv pe diagrama V-K; d) modificarea circuitului (linie întreruptă) pentru eliminarea hazardului.

Se va exemplifica apariţia hazardului pe următoarele două funcţii: $f_1 = A\overline{B} + \overline{A}BC$, $f_2 = AB + \overline{A}C$. Funcţia f_1 nu poate fi adusă pentru nici o combinaţie de

valori ale variabilelor B şi C la forma $A + \bar{A}$ şi la fel nu poate fi adusă la forma $\bar{B} + B$ pentru nici o combinaţie de valori date variabilelor A şi C. A doua funcţie poate fi adusă la forma $f_2 = A + \bar{A}$ dacă $B = C = 1$. Rezultă că circuitul, Figura 2.23-a, care implementează funcţia f_2, prezintă hazard static 1; generarea hazardului se poate analiza cu ajutorul diagramelor de semnal din Figura 2.23-b. Variaţia variabilei A, de la 1 la 0, ajunge la intrarea porţii 3 pe primul traseu prin semnalul produs logic AB de la poarta 1 şi pe al doilea traseu prin semnalul produs logic $\bar{A}C$ de la poarta 2 (dar cu o întârziere τ_p), deci un asincronism la intrarea porţii 3. De asemenea, analiza apariţiei hazardului static 1 se poate face şi pe diagrama V-K din Figura 2.23-c. Funcţia are valoarea logică 1 la ieşire când configuraţia variabilelor de intrare corespunde unui implicant prim, IP, implicant cee a fost selectat pentru exprimarea funcţiei. Mai plastic exprimat, "punctul de funcţionare" al funcţiei se află în interiorul suprafeţei unui implicant prim când la intrarea circuitului se aplică valorile coordonatelor acelui implicant. Dacă prin comutarea unei singure variabile "punctul de functionare" rămâne în interiorul suprafeţei acelui implicant prim nu se generează hazard (ieşirea fiind asigurată în valoare 1), dar dacă "punctul de funcţionare" trece în interiorul asuprafeţei unui alt implicant prim, se generează hazard static 1 (ieşirea, pe durata trecerii între cele două suprafeţe/inplicanţi nu mai este asigurată în 1, devine 0). În diagrama V-K a funcţiei f_2 la comutarea lui A de la 1 la 0 se trece din suprafaţa implicantului prim AB (pentru care ieşirea este în 1) în suprafaţa implicantului prim $\overline{A}C$ (pentru care ieşirea este în 1). Pentru ca prin această comutare "punctul de funcţionare" să rămână totuşi în interiorul unei suprafeţe de 1 (care să asigure ieşirea la valoarea 1) se introduce încă o suprafaţă în diagrama V-K (o suprafaţă "punte" între suprafeţele celor doi implicanţi primi) care corespunde implicantului prim BC. În consecinţă, în structura circuiitului, Figura 2.23-d, trebuie adăugată poarta 4 care generează produsul BC (partea de circuit desenată punctat). Fizic, introducerea implicantului prim neesenţial BC elimină hazardul deoarece asigură ieşirea funcţiei în 1 pe durata de propagare, τ_p, a semnaluilui A prin poarta inversoare când la ieşire ar fi $A + \bar{A} = 0$.

Ca regulă generală, se poate enunţa: nu apare hazard la comutarea unei variabile dacă "punctul de funcţionare" rămâne în interiorul aceleiaşi suprafeţe de 1, dar dacă se trece intr-o altă suprafaţă de 1, se va produce hazard static 1. Pentru eliminarea hazardului este necesară introducerea unui produs în structura circuitului a cărui coordonată determină o suprafaţă "punte" (implicant prim cu valoarea 1) pe diagrama V-K, în interiorul căreia să rămînă "punctul de funcţionare" la comutarea variabilei.

Funcţia $(A+B)(\bar{A}+C)$, care este forma duală a funcţiei analizate anterior, pentru valorile $C = 0,\ B = 0$ se reduce la produsul $A \cdot \bar{A}$, va genera hazard static 0, adică va produce un glitch 1 pe ieşire când variabila de intrare A va comuta de la 0 la 1. Pentru eliminarea acestui glitch este necesar a se introduce în implementarea circuitului a termenului sumă (suplimentar) $B + C$, care va asigura valoarea 0 pe ieşirea funcţiei atunci când A comută de la 0 la 1.

În exprimările anterioare, de multe ori, în loc de sintagma "forma minimă" a funcţiei s-a utilizat forma redusă. Aceasta pentru că chiar dacă prin sinteză se obţine o formă minimă, prin implementare, pentru eliminarea hazardului, se introduc implicanţi neesenţiali şi în final se ajunge la o formă redusă. Circuitul din Figura 2.23-d demonstrează aceasta, pentru eliminarea hazardului static, formei minime $AB+\overline{A}C$ i se adaugă termenul BC, deci implementarea este a unei forme reduse.

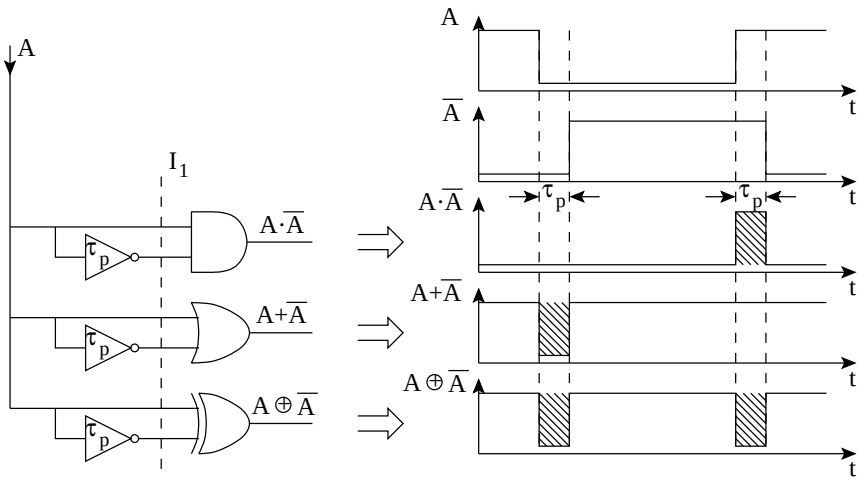


Figura 2.24 Circuite cu porţi AND,OR,XOR pentru detectarea de fronturi.

Producerea asincronismului la intararea unei porţi prin aplicarea unei variabile şi a aceleiaşi variabile negate (prin intermediul unui inversor) poate fi utilizată pentru **detectarea de fronturi**, Figura 2.24. La aceste circuite dacă se consideră punctele de intrare în poartă în poziţia figurată de linia întreruptă notată cu I_1 se observă un asincronism la intrarea porţii; variaţia pe intrarea $\bar{A}$ apare totdeauna întârziată cu timpul τ_p de propagare prin inversor în raport cu variabila pe intrarea A. Poarta AND generează un impuls pozitiv cu durata τ_p pentru un front pozitiv, poarta OR generează un impuls negativ pentru un front negativ iar poarta XOR generează un impuls negativ pentru oricare tip de front.

La ieşirea unui CLC poate apărea pentru o comutare a unei variabile de intrare de la 1 la 0 şi următoarea comutare hazardată $1 \to 0 \to 1 \to 0$ (trei schimbări ale valorii ieşirii) respectiv pentru o comutare de la 0 la 1 pot apărea următoarele trei schimbări ale ieşirii: $0 \to 1 \to 0 \to 1$, acest comportament este referit ca **hazard dinamic**. Hazardul dinamic este cauzat de existenţa în circuit a trei sau mai multe trasee diferite pentru o variabilă de intrare, fiecare traseu având un alt timp de propagare. Astfel de structuri rezultă în urma factorizării expresiilor sumă de produse, Figura 2.21-b, sau când există trasee lungi de interconectare între porţi de viteză ridicată. Hazardul dinamic poate fi evitat dacă se realizează implementări numai pe două niveluri logice, adică aducerea expresiei de implementat la o formă FD.

Noţiunile prezentate până acum în acest capitol împreună cu cele de structurarea porţilor, expusă în capitolul 1, constituie un suport pentru realizarea unui CLC pornind de la formularea funcţionării cerute/impuse până la implementarea într-o anumită tehnologie. Dar, parcurgerea tuturor etapelor, de fiecare dată când se realizează un sistem, apare ca o abordare nerecomandată atât din punct de vedere al efortului cât şi al timpului consumat, deci in final al costului. În acest sens, este recomandabil ca pentru realizarea unui sistem să se utilizeze componente "prefabricate". În cadrul circuitelor combinaţionale pentru anumite funcţii logice, aritmetice sau de comunicaţie foarte des utilizate, există deja circuite care modelează acele funcţii realizate într-o anumită tehnologie, integrate pe scară mică **SSI** (**S**mall **S**cale **I**ntegration),

pe scară medie **MSI** (**M**edium **S**cale **I**ntegration) sau chiar integrate pe scară mare **LSI** (**L**arge **S**cale **I**ntegration) . În stadiul actual, când este necesară realizarea unei astfel de funcţii uzuale, nu se mai implementează circuitul, ci se alege un circuit obtenabil comercial pentru care este necesară cunoaşterea datelor electrice de catalog şi a variantelor de funcţie logică. Pentru un sistem mai complex, funcţia acestuia se caută a fi sintetizată din funcţii uzuale pentru care există deja circuite implementate, deci realizarea sistemului se reduce la selectarea potrivită de componente integrate deja existente şi conectarea lor corespunzator. **Funcţiile uzuale care au un suport circuistic combinaţional sunt de tip: logic, aritmetic şi de comunicaţie**. În continuare, în acest capitol, se vor prezenta unele circuite combinaţionale, de facto standard, care realizează astfel de funcţii.

2.4 CIRCUITE COMBINAŢIONALE PENTRU FUNCŢII LOGICE

Forma redusă, sau cea minimă, a funcţiei unui circuit combinaţional este fie o sumă de produse, fie un produs de sume. Aceste forme pot fi implementate pe organizări cu două niveluri de AND-OR (NAND-NAND) sau OR-AND (NOR-NOR). În consecinţă, este normal ca pentru implementarea unei funcţii reduse, sau minime, să se apeleze la acele structuri, pe două nivele, deja realizate. Pentru toate circuitele combinaţionale prezentate în continuare se va urmări măsura în care acestea pot fi un suport pentru implementarea de funcţii logice pe două niveluri.

2.4.1 Codificatorul

Funcţia de codificare constă într-o translatare de limbaj. Particularizând această foarte generală definiţie, o codificare în binar constă într-o aplicaţie de pe o mulţime cu n elemente disjuncte într-o mulţime de cuvinte binare cu lungime de m biţi; $m = \lceil \log_2 n \rceil$ biţi, $2^m \geq n$ (simbolul $\lceil\ \rceil$ denotă cel mai mic număr intreg egal sau mai mare decât numărul din interiorul simbolului). Deci, circuitul codificator, CDC, este caracterizat de n intrări şi m ieşiri, notat simbolic cu **CDCn:m**. Aplicaţia realizată de circuitul codificator este injectivă, adică fiecărei intrări active, din cele n, îi corespunde doar un singur cuvânt de ieşire cu lungimea de m biţi.

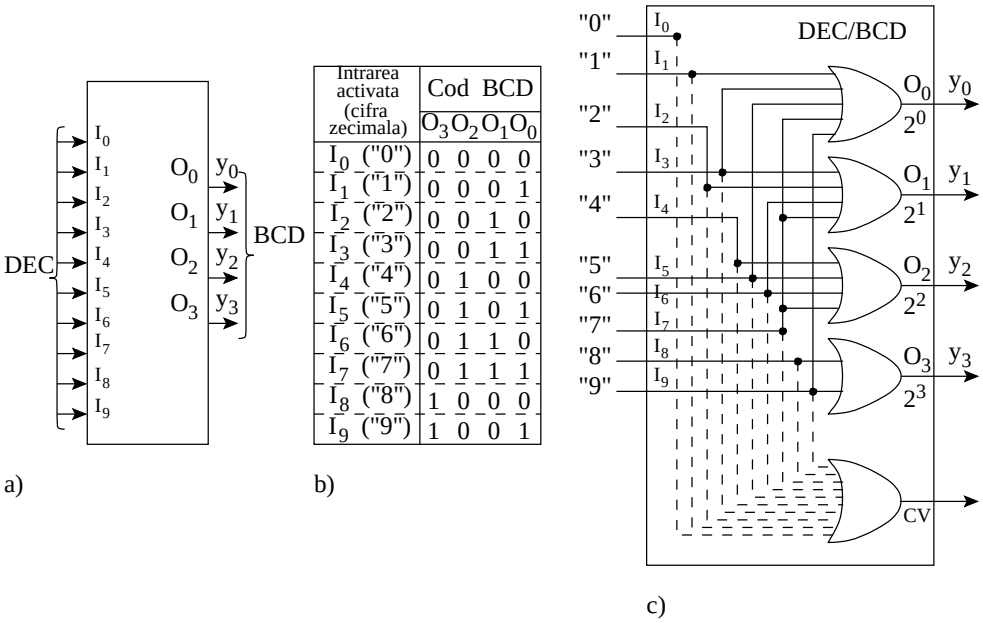
Pentru exemplificare se va prezenta sinteza unui codificator din zecimal (DEC) în cod BCD, cu schema bloc reprezentată în Figura 2.25-a. Circuitul are 10 intrări $(I_0, I_1, I_2, ..., I_9)$ corespunzătoare celor zece cifre zecimale şi generează un cuvânt de ieşire pe 4 biţi ($2^4 > 10$). La aplicarea pe intrare a cifrei zecimale i, prin activarea intrării I_i, la ieşire se generează codul BCD al cifrei i. Detaliat, funcţionarea codificatorului DEC/BCD este descrisă prin tabelul de adevăr din Figura 2.25-b. Se poate deduce expresia logică a ieşirii O_3 în felul următor: O_3 are valoarea logică 1 numai când la intrare se aplică cifra 8 sau cifra 9, adică este activată intrarea I_8 *SAU* I_9, deci

$$O_3 = I_8 + I_9$$

Raţionând în acelaşi mod se deduc ecuaţiile logice şi pentru ceilalţi trei biţi ai

cuvântului de ieşire:

$$O_2 = I_4 + I_5 + I_6 + I_7$$
$$O_1 = I_2 + I_3 + I_6 + I_7$$
$$O_0 = I_1 + I_3 + I_5 + I_7 + I_9$$



Intrarea activata (cifra zecimala)	Cod BCD			
	O_3	O_2	O_1	O_0
I_0 ("0")	0	0	0	0
I_1 ("1")	0	0	0	1
I_2 ("2")	0	0	1	0
I_3 ("3")	0	0	1	1
I_4 ("4")	0	1	0	0
I_5 ("5")	0	1	0	1
I_6 ("6")	0	1	1	0
I_7 ("7")	0	1	1	1
I_8 ("8")	1	0	0	0
I_9 ("9")	1	0	0	1

Figura 2.25 Sinteza circuitului codificator zecimal - binar, DEC/BCD : a) schema bloc; b) tabelul de adevăr; c) structura circuitului.

Structura circuitului codificator, implementat pe baza acestor ecuaţii, este reprezentată în Figura 2.25-c şi se compune din patru porţi OR cu 2,4 şi 5 intrări.

În cazul general de codificare a n elemente, prin activare în 1 logic, pe cuvinte binare cu lungimea de m biţi circuitul codificator constă în m porţi OR cu maximum n intrări, iar când intrările I_i_L sunt active în 0 logic codificatorul este structurat din m porţi NAND cu maxim n intrări.

Observaţia importantă care rezultă din această implementare este: Codificatorul este implementat pe un nivel OR sau operaţia de **codificarea este o funcţie logică OR**.

Analizând implementarea CDC apar două deficienţe. Prima, la ieşire nu se poate face distincţie între cazul când cuvântul de cod pe ieşire are valoarea $O_3O_2O_1O_0 = 0000$, datorită faptului că nu s-a activat nici o intrare, sau cazul când s-a activat intrarea I_0. Se poate face distincţia între cele două cazuri dacă se generează un semnal de ieşire CV care semnalizează cod valid CV = 1, respectiv cod invalid CV = 0. Citirea unui cod invalid CV = 0 apare numai atunci când nu este activată nici o intrare şi este citit un cod corect când una din intrări I_i este activată, deci rezultă ecuaţia logică $CV = (I_0 + I_1 + ... + I_8 + I_9)$, care este implementată în Figura 2.25-c prin traseele cu linii punctate şi o poartă OR.

A doua deficienţă constă în faptul că CDC funcţionează corect numai când o singură intrare este activată, de exemplu la activarea simultană a intrărilor I_3 şi I_4 cuvântul de cod generat este 0111 (incorect!), care ar corespunde aplicării cifrei 7 dar intrarea I_7 nu a fost activată ci doar I_3 şi I_4. Ca circuit codificator DEC/BCD, obtenabil comercial, există circuitul integrat 74xx147.

2.4.2 Codificatorul prioritar, CDCP

A doua deficienţă a circuitului codificator se poate elimina prin introducerea unei priorităţi în generarea codului. La o codificare cu prioritate, fiecărei intrări I_i i se alocă o anumită prioritate în intervalul de la cea mai mică până la cea mai ridicată prioritate. Astfel, la activarea simultană a mai multor intrări codificatorul prioritar va genera numai codul intrării activate care are prioritatea cea mai ridicată.

Pentru exemplificare se va considera un codificator prioritar cu opt intrări $I_0, I_1, ...,$ I_7 care generează pe cele trei ieşiri O_2, O_1, O_0 cuvântul de cod de trei biţi $y_2y_1y_0$ în binar natural, Figura 2.26-a. Pentru acest codificator CDCP8 : 3 alocarea priorităţii pe intrări este de la I_0 spre I_7; intrarea I_0 are prioritatea ce mai mică iar I_7 cea mai ridicată.

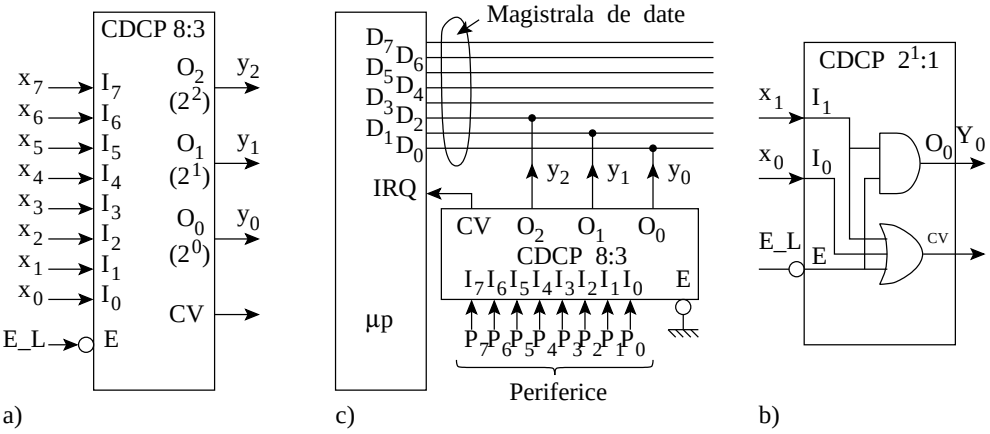


Figura 2.26 Codificatorul prioritar, CDCP: a) schema bloc pentru CDCP2^3:3; b) structura codificatorului prioritar elementar CDCP2^1:1; c) sistem de intreruperi vectorizate implementat pe bază de CDCP8:3 (la un sistem pe bază de microprocesor).

Pentru următorul cuvânt $X = x_7x_6x_5x_4x_3x_2x_1x_0 = 00010111$, aplicat pe intrările corespunzătoare, cuvântul de cod generat este $Y = y_2y_1y_0 = 100$ ceea ce corespunde activării intrării I_4, celelalte intrări activate $I_2 = 1, I_1 = 1$ şi $I_0 = 1$ nu afectează cuvântul de ieşire deoarece au prioritate mai mică decât intrarea I_4. Iar, pentru cuvâmtul de cod generat pe ieşire $Y = y_2y_1y_0 = 101$, care corespunde activării intrării $I_5 = 1$, configuraţia cuvântului de intrare este de forma $x_7x_6x_5x_4x_3x_2x_1x_0 =$ 001xxxxx, ceea ce înseamnă că intrările de la I_0 la I_4, care au prioritate mai mică decât I_5, pot fi activate sau nu (indiferente). Pentru codificatorul prioritar, între cuvântul X aplicat pe intrare şi cel generat pe ieşirea Y, interpretate ca numere întregi în binar natural, se poate scrie următoarea relaţie:

$$Y = \lfloor \log_2 X \rfloor \tag{2.10}$$

Codificatorul calculează la ieşire un număr întreg Y care este partea întreagă $\lfloor \log_2 X \rfloor$ a logaritmului în baza doi a numărului X aplicat pe intrare. Deci, circuitul codificator prioritar realizează şi o funcţie aritmetică.

În continuare se va realiza sinteza logică a circuitului CDCP2^3:3. Sinteza circuitului, pornind de la tabelul de adevăr, se poate realiza doar cu un program de calcul deoarece trebuie definite trei funcţii O_2, O_1, O_0 fiecare de opt intrări $I_7, ..., I_i, ..., I_1, I_0$ (ar necesita trei tabele de adevăr cu 256 linii!). Sinteza poate fi mult uşurată dacă se utilizează observaţia: un cuvânt binar la ieşire, $y_2y_1y_0|_2 = i|_{10}$, este generat de toate cuvintele de intrare X care au biţii egali cu 0 în poziţiile superioare lui i, bitul egal cu 1 în poziţia i şi biţi de valoare indiferentă în poziţiile inferioare lui i. În spiritul acestei observaţii se pot introduce următoarele variabile intermediare H_i (care definesc expresia logică pentru care intrarea $I_i, 0 \leq i \leq 7$, îşi generează codul pe ieşire):

$$\begin{aligned}
&H_7 = I_7\\
&H_6 = I_6 \cdot \bar{I}_7\\
&H_5 = I_5 \cdot \bar{I}_6 \cdot \bar{I}_7\\
&.\\
&.\\
&H_0 = I_0 \cdot \bar{I}_1 \cdot \bar{I}_2 \cdot \bar{I}_3 \cdot \bar{I}_4 \cdot \bar{I}_5 \cdot \bar{I}_6 \cdot \bar{I}_7
\end{aligned}$$

Cu aceste variabile intermediare sinteza codificatorului prioritar se realizează similar ca cea a codificatorului din secţiunea 2.4.1. Din tabelul cuvintelor de cod (numărul lor fiind limitat de data aceasta la opt), pentru fiecare intrare, Figura 2.25-b, se deduc expresiile pentru O_2, O_1 şi O_0 sub forma unor sume logice în felul următor:

$$\begin{aligned}
&O_2 = H_4 + H_5 + H_6 + H_7\\
&O_1 = H_2 + H_3 + H_6 + H_7\\
&O_0 = H_1 + H_3 + H_6 + H_7\\
&CV = I_1 + I_2 + I_3 + I_4 + I_5 + I_6 + I_7
\end{aligned}$$

La grupul funcţiilor O_2, O_1, O_0 s-a adăugat şi semnalul de cod valid la ieşire, CV, dedus la sinteza codificatorului. Pentru ca circuitul CDCP2^3 : 3 să fie flexibil în aplicaţii mai trebuie înzestrat cu un semnal de validare funcţionare circuit, E_L (Enable, activ în L, $E = \overline{E\_L}$). Adică, circuitul va genera semnale numai când este validat de către semnalul $E\_L = 0$. Aceasta implică pentru semnalele O_2, O_1, O_0 şi CV ca generarea lor să fie condiţionată de activarea lui $E\_L = 0$. Din relaţiile anterioare, prin substituţiile corespunzătoare, se obţin următoarele relaţii logice pentru structurarea circuitului CDCP2^3:3:

$$\begin{aligned}
&O_2 = E(I_7 + I_6 + I_5 + I_4)\\
&O_1 = E(I_7 + I_6 + I_3\bar{I}_4\bar{I}_5 + I_2\bar{I}_4\bar{I}_5)\\
&O_0 = E(I_7 + I_5\bar{I}_6 + I_3\bar{I}_4\bar{I}_6 + I_1\bar{I}_2\bar{I}_4\bar{I}_6)\\
&CV = E(I_0 + I_1 + I_2 + I_3 + I_4 + I_5 + I_6 + I_7)
\end{aligned} \tag{2.11}$$

Aplicaţii CDCP2^n:n sunt numeroase, majoritatea lor se bazează pe funcţia sa aritmetică, relaţia 2.10, adică se determină dintr-o mulţime de elemente active pe acela căruia i s-a alocat prioritate maximă.

O astfel de aplicaţie pentru CDCP este implementarea întreruperilor vectorizate la un microprocesor, μP, Figura 2.26-b. Principial, succesiunea în realizarea întreruperilor vectorizate este următoarea:

- când un periferic P_i $(i = 0, 1, ..., 7)$ necesită serviciul μP activează intrarea I_i;
- codificatorul prioritar CDCP8:3 va genera pe ieşire (legată la magistrala de date) codul zecimal-binar al intrării I_i şi va activa semnalul de cerere de întrerupere (IRQ – Interrupt ReQuest) al μP, $IRQ = CV$;
- μP sesizând, prin IRQ, o cerere de întrerupere de la un periferic va citi de pe magistrala de date codul ("vectorul") perifericului respectiv;
- cu ajutorul vectorului transmis μP, acesta va calcula adresa din memorie unde se află subrutina care va rezolva problema perifericului solicitant. Se rulează subrutina;
- după servirea perifericului se trece la servirea altui periferic dacă semnalul IRQ este activat.

Într-un sistem de calcul pentru o funcţionare economică, şi eventual în timp real, fiecărui periferic i se acordă o anumită prioritate în raport cu celelalte. În această aplicaţie, printr-o implementare pe baza de CDCP8:3, fiecărui periferic P_i i se fixează prioritatea prin modul în care este conectat la una dintre cele opt intrări $I_0, I_1, ..., I_7$ (se consideră că prioritatea cea mai mare o are I_7 iar cea mai mică o are I_0). Dacă simultan IRQ este activat de perifericele P_2, P_3 şi P_6 numai vectorul 101 va fi înscris pe magistrala de date corespunzător perifericului P_6, deci acesta va fi servit. După servirea perifericului P_6 vor fi servite în ordine perifericele P_3 şi P_2 dacă acestea menţin activate I_3 respectiv I_2 (IRQ=1).

Uneori, în aplicaţii sunt necesare codificatoare prioritare cu mai mult decât opt intrări, pentru care nu există circuite de tip MSI obtenabile. Dacă, de exemplu, sunt necesare 16 intrări $(x_0, x_1, ..., x_7, x_8, ..., x_{15})$ se va structura un CDCP2^4:4 prin înserierea a două circuite CDCP2^3:3 ca în Figura 2.27. La primul CDCP2^3:3 se repartizează intrările $x_0, x_2, ..., x_7$, codificabile prin cuvintele de ieşire $y_3y_2y_1y_0$ =0000, 0001, ...,0111, iar la al doilea CDCP2^3:3 se repartizează intrările $x_8, x_9, ..., x_{15}$, codificabile prin cuvintele de la ieşire $y_3y_2y_1y_0$ =1000, 1001, ...,1111. Din analiza cuvintelor de cod se constată că bitul y_3 are valoarea 1 totdeauna când cel puţin o intrare din intervalul $x_8 \div x_{15}$ este activată şi are valoarea 0 când nici una dintre intrările din acest interval nu este activată. Deoarece fiecare dintre codificatoare produce numai trei biţi de cod O_2, O_1, O_0 bitul al patrulea din cod, y_3, va fi determinat de activarea a cel puţin unei intrări din intervalu $x_8 \div x_{15}$, deci poate fi calculat prin semnalul CV de la al doilea codificator, $y_3 = CV_2$. Ceilalţi trei biţi de cod y_3, y_2, y_1 sunt generaţi fie pe ieşirile O_2, O_1, O_0 ale primului codificator, când se activează intrări din intervalul $x_0 \div x_7$, fie pe ieşirile O_2, O_1, O_0 ale celui de-al doilea codificator, când se activează intrări din intervalul $x_8 - x_{15}$, deci rezultă că se obţine cuvântul de ieşire printr-un SAU între cuvintele corespunzătoare celor două codificatoare. Se observă că o activare în intervalul $x_8 \div x_{15}$, a semnalului $CV = 1$ (legat la intrarea de validare a primului codificator care este activă în zero, E_L=0) va devalida pe primul codificator $CV = E\_L = 1$, acesta va fi validat numai când nu este activată nici o intrare din intervalul $x_8 \div x_{15}$, $CV = E\_L = 0$.

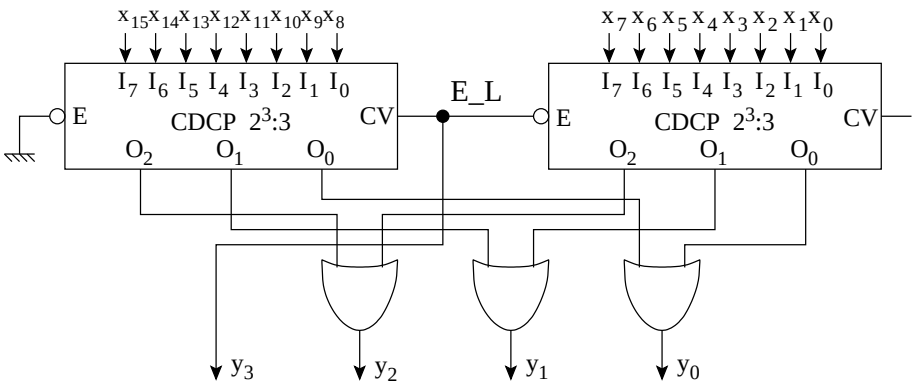


Figura 2.27 Structurarea unui codificator prioritar cu 16 intrări ca o extensie serie de la un codificator prioritar cu 8 intrări.

Extensia la un codificator cu 32 de intrări, adică un CDCP2^5:5, se poate obţine printr-o înseriere a două CDCP2^4:4, fiecare cu o structurare ca cea din Figura 2.27. Pentru o extensie care nu este o dublare a numărului de intrări, de exemplu, la o extensie de la 16 intrări la 24 intrări se înseriază un un CDCP2^3:3 cu un CDCP2^4:4. Deoarece bitul al cincelea y_4, din cuvântul de cod de ieşire $y_4y_3y_2y_1y_0$, are valoarea 1 numai când există o activare în intervalul de intrări $x_{16} \div x_{23}$, în rest are valoarea 0 numai, acest bit se obţine din semnalul CV de la CDCD 2^3:3. De asemenea, bitul y_3 are valoarea 0 pe intervalul $x_{15} \div x_{23}$ în rest se calculează ca în Figura 2.27. Cu aceste observaţii se poate structura uşor codificatorul prioritar cu 24 intrări.

Ecuaţiile logice pentru cel mai simplu CDCP2^1:1, cu două intrări I_0, I_1 şi ieşirea O_0, referit **codificator prioritar elementar, CDCPE**, se obţin din relaţiile 2.11 prin particularizarea $I_2 = I_3 = I_4 = I_5 = I_6 = I_7 = 0$

$$\begin{aligned} O_0 &= E \cdot I_i \\ CV &= E(I_0 + I_1) \end{aligned} \tag{2.12}$$

cu structurarea din Figura 2.26-b.

Pornind de la CDCP2^1:1, prin dublarea repetată a numărului intrărilor şi corespunzător creşterii cu un bit a cuvântului de ieşire, se pot defini şi, respectiv, structura recursiv codificatoarele CDCP2^2:2, CDCP2^3:3, CDCP2^4:4,..., CDCP2^n:n. Se va nota pentru codificatorul CDCP2^n:n, cu 2^n intrări şi un cuvânt de cod de n biţi, dimensiunea respectiv adâncimea prin $S(n)$ şi $D(n)$. Structurarea recursivă pornind de la CDCPE se poate face prin extensie paralelă. În [Stefan ′00]se demonstrează că pentru extensia paralelă caracteristicile obţinute sunt $S(n) \in 2^n$ şi $D(n) \in O(2^n)$, adică o dimensiune care depinde linar de numărul de intrări (2^n), accepatbil pentru implementare, şi o adâncime tot liniară în funcţie de numărul de intrări, deci performanţa de viteză scăzută pentru n ridicat. Iar pentru structurarea serie, a unui CDCP2^n:n porinind de la CDCPE, caracteristicile sunt $S(n) \in 2^n$ şi $D(n) \in O(n)$, de data această rezultă şi pentru performanţa de viteză o mărime acceptabilă.

Obtenabil comercial există codificatorul cu 8 căi de intrare cu ieşirea în cod binar natural, 74xx148, a cărui structură şi funcţionare pot fi regăsite în codificatorul generic prezentat în Figura 2.26.

2.4.3 Decodificatorul, DCD

Funcţia de decodificare este o aplicaţie de pe o mulţime de 2^n cuvinte binare într-o mulţime de 2^n elemente distincte, prin decodificare fiecărui cuvânt binar cu lungimea le n biţi i se asignează un element distinct. Deci, funcţia de decodificare apare ca aplicaţia inversă celei de codificare (ce asignează unui element dintr-o mulţime de 2^n elemente distincte un cuvânt de cod). Circuitul care modelează funcţia de decodificare este notat prin **DCDn:2^n** deoarece prezintă n intrări (un cuvânt de intrare cu lungimea de n biţi) şi 2^n ieşiri (distincte).

Ca exemplificare se va prezenta circuitul decodifcator pentru cuvinte cu lungime de 2 biţi, DCD2:4. Tabelul de adevăr al circuitului DCD2:4 este prezentat în Figura 2.28-a. Pe lânga cuvântul de intrare x_1x_0, circuitului i se aplică şi o un semnal de validare E_L ($E = \overline{E}\_L$) astfel că funcţionarea circuitului, obţinerea unei ieşiri active din cele patru O_3, O_2, O_1 şi O_0, este posiblilă numai când şi semnalul de validare este activ $E\_L = 0$. Fiecărui cuvânt de intrare x_1x_0 din cele patru (00,01,10,11) îi corespunde o ieşire activă (respectiv: $O_0 = 1, O_1 = 0, O_2 = 1, O_3 = 1$), deci **funcţionarea circuitului apare ca o identificare de cod**. Din tabelul de adevăr pentru fiecare ieşire se obţine expresia logică de forma $O_0 = E \cdot \bar{I}_1\bar{I}_0$; $O_0 = E \cdot \bar{I}_1 I_0$; $O_0 = E \cdot I_1\bar{I}_0$; $O_0 = E \cdot I_1 I_0$. Se observă ca o ieşire O_i devine activă când pe intrare se aplică mintermul P_i, $i = 0, 1, 2, 3$. Se poate generaliza, un circuit decodificator DCDn:2^n va genera ieşirea O_i activă atunci când pe intrare se aplică mintermul P_i, $i = 0, 1, 2, ..., 2^n - 1$. Decodificatorul 2:4 este reprezentat ca schemă bloc în Figura 2.28-b, iar ca structură în Figura 2.28-c.

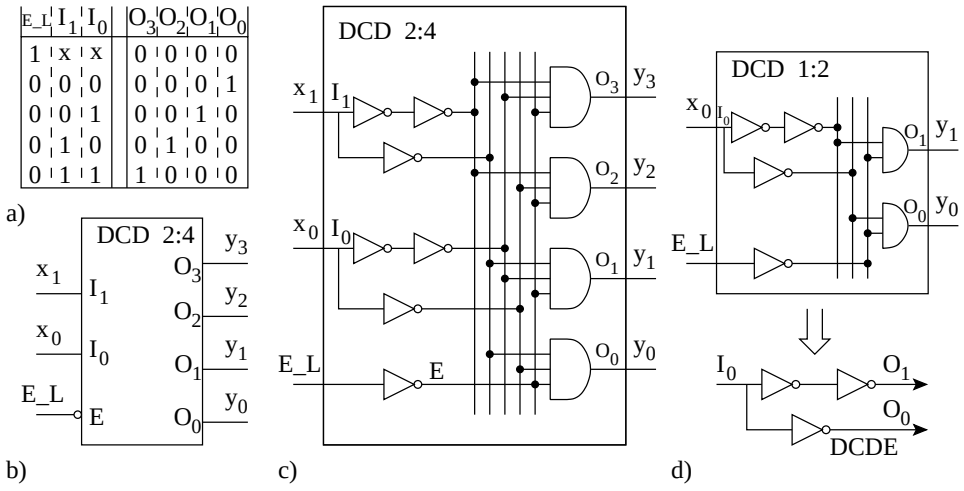


Figura 2.28 Decodificatorul DCD2:4: a) tabelul de adevăr; b) reprezentare, schemă bloc; c) structură compusă din decodificatoare elementare şi porţi AND; c) structurarea decodificatorului elementar, DCD1:2.

Structura **decodificatorului elementar DCDE** (DCD1:2), cu o singură intrare şi două ieşiri, se poate deduce pornind de la DCD2:4, prin eliminarea intrării I_1, fie din structura circuitului, fie din tabelul de adevăr (şi în acest ultim caz efectuând sinteza). În structura obţinută pentru DCDE, Figura 2.28-d, dacă se elimină semnalul

de validare E_L nu mai sunt necesare cele două porţi AND, circuitul decodificator elementar se reduce la cele trei porţi inversor. Pentru intrarea $I_0 = 0$ ieşirea activă, $O_0 = 1$ se obţine printr-un buffer inversor, iar pentru, $I_0 = 1$, ieşirea activă $O_1 = 1$ se obţine prin înserierea a două porţi buffer inversor; aceasta este de fapt o structură de circuit furcă $2 - 1$, Figura 1.67. Se adoptă un circuit furcă pentru ca întârzierile celor două ieşiri O_1 şi O_0 să fie egale (poate fi necesar un circuit furcă $(n-1) - n$ în funcţie de efortului electric H). Din punct de vedere logic DCDE se reduce doar la un singur inversor, dar cu o astfel de structurare O_1 se obţine nebufferat, $O_1 = I_0$, iar O_0 se obţine bufferat prin inversor, $O_0 = \overline{I_0}$. Această variantă de structurare se evită în practică deoarece pentru $I_0 = 1$, ieşirea este conectată direct la intrare şi, în plus, semnalele pe cele două ieşiri au timp de propagare diferiţi. Pentru DCDE (furcă 2-1) adâncimea şi dimensiunea au respectiv valorile $D(1) = 2,\ S(1) = 3$.

Pornind acum invers de la DCDE spre DCD2:4, se observă că DCD2:4 se structurează din două decodificatoare elementare şi patru porţi AND3. Un DCD3:8 s-ar structura din 3 DCDE şi opt porţi AND4. Extinzând la DCDn:2^n structurarea ar fi din nxDCDE şi 2^n porţi AND(n+1). Neluând în considerare intrarea de validare E (care nu reduce din generalizare, dar porţile AND vor avea o intrare mai puţin) pentru DCDn:2^n se obţin caracteristicile de adâncime $D(n)_{DCD}$ şi dimensiune $S(n)_{DCD}$:

$$\begin{aligned} D(n)_{DCD} &= D(n)_{AND} + D_{DCDE} = 1 + 2 = 3 \in O(1) \\ S(n)_{DCD} &= 2^n \cdot S(n)_{AND} + n \cdot S_{DCDE} = 2^n \cdot n + n \cdot 3 \in O(n \cdot 2^n) \end{aligned} \tag{2.13}$$

Această structurare (ca o extensie pornind de la DCDE) pentru DCDn:2^n, care poate fi referit cu adâncime constantă, prezintă o foarte bună caracteristică de viteză dar o dimensiune $\in O(n \cdot 2^n)$ ce poate ridica unele probleme la implementare. De fapt, valorile calculate prin relaţiile 2.13 trebuie luate ca valori minimale. Aceste valori minimale pot fi utilizate pentru evaluările implementărilor în tehnologia bipolară şi când n nu are valori prea ridicate, este cazul circuitelor docodificatoare realizate ca circuite integrate MSI sau circuite discrete, cu n cel mult 4 sau 5. Pentru implementări tip VLSI, este cazul memoriilor de capacitate ridicată când $n \geq 30$, în tehnologia CMOS (pentru care timpii de propagare depind puternic de încărcare) structurarea decodificatorului se va modifica faţă de structura cu adâncimea constantă prezentată în Figura 2.28-c. În primul rând DCDE va fi un circuit furcă cu un număr de niveluri logice (Tabelul 1.16) în funcţie de efortul electric. În al doilea rând o poartă AND cu n intrări, de exemplu când $n = 32$, pe un singur nivel logic este tehnologic foarte dificil de realizat, deci poarta se structurează sub formă de arbore din porţi AND, care au un număr mai mic de intrări, rezultând un arbore cu mai multe niveluri de AND. Dacă, de exemplu, poarta AND se structurează logic sub formă de arbore din porţi AND2 atunci se obţin $log_2 n$ niveluri de AND2 iar adâncimea DCD2:n ar fi $D(n) = D_{DCDE} + log_2 n$. Dar, de fapt, adâncimea arborelui de porţi AND rezută în funcţie de efortul F (Tabelul 1.15) al traseului prin arborele de porţi AND care substituie poarta ANDn.

Exemplul 2.12 Pentru cele trei structuri de poartă AND (notate cu varianta a, b şi c) reprezentate în Figura 2.29 să se estimeze care este cea mai rapidă. Se consideră că la fiecare structură capacitatea de intrare C_{in} este de 4 unităţi; estimarea se va face pentru două valori ale efortului electric $H = 1$ şi $H = 12$. Apoi structura mai rapidă să se dimensioneze.

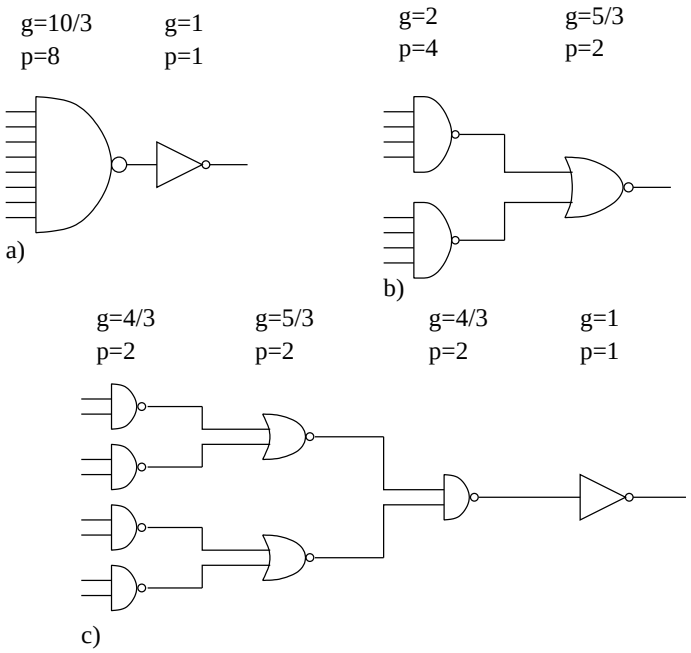


Figura 2.29 Trei variante de structurarea unei porţi AND cu opt intrări.

Soluţie. Se calculează efortul logic total ca produs dintre efortul logic de pe fiecare nivel (valorile se obţin din Tabelul 1.13) pentru fiecare dintre cele trei variante şi se obţine: $G_a = 10/3 \cdot 1 = 3,33$; $G_b = 2 \cdot 5/3 = 3,33$; $G_c = 4/3 \cdot 4/3 \cdot 4/3 \cdot 1 = 2.96$. Analizând numai după efortul logic ar rezulta că varianta c ar fi cea mai rapidă, dar întârzierea depinde de efortul total, $F = GBH$, şi nu numai de efortul logic. Aplicând relaţia 1.102, pentru calculul întârzierii minime D pe un traseu, rezultă:

$$\begin{aligned} &\text{varianta } a: && \hat{D} = 2(2,33 \cdot H)^{1/2} + 9 \\ &\text{varianta } b: && \hat{D} = 2(2,33 \cdot H)^{1/2} + 6 \\ &\text{varianta } c: && \hat{D} = 4(2,96 \cdot H)^{1/4} + 7 \end{aligned}$$

în care introducând cele două valori pentru $H = 1$ şi pentru $H = 12$. Se obţin valorile:

	$H = 1$			H=12		
Varianta	$NF^{1/N}$	P	$\hat{D} = NF^{1/N} + P$	$NF^{1/N}$	P	$\hat{D} = NF^{1/N} + P$
a	3,65	9,0	12,65	12,64	9,0	21,64
b	3,65	6,0	9,65	12,64	6,0	18,64
c	5,25	7,0	12,25	9,77	7,0	16,77

Deci pentru efortul electric $H = 1$ ($C_{in} = 4, C_0 = H \cdot C_{in} = 4$) varianta b are întârzierea cea mai mică, iar pentru $H = 12$ ($C_{in} = 4, C_0 = 12 \cdot 4 = 48$) varianta c are întârzierea cea mai mică.

Dimensionarea variantei c (H=12) : $\hat{f} = F^{1/4} = (2,96 \cdot 12)^{1/4} = 2,44$. Pornind de la ultima poartă (inversor) care are $C_0 = 12 \cdot 4 = 48$ unităţi, rezultă sarcina pentru nivelul al treilea (NAND2), $C_{in4} = 48 \cdot 1/2,44 = 19,66$. Sarcina pentru nivelul al doilea (NOR2) este $C_{in_3} = (19,66 \cdot 4/3)/2,44 = 10,73$, iar sarcina pentru primul nivel (NAND2) $C_{in_2} =$

$(10,73 \cdot 5/3)/2,44 = 7,33$. În final, ca verificare, se poate calcula capacitatea de intrare $C_{in} = (7,3 \cdot 4/3)/2,44 = 4$.

Dimensionarea variantei b (H=1) : $\hat{f} = F^{1/2} = (3,33 \cdot 1)^{1/2} = 1,83$. Pornind de la ultima poartă NAND2 care are sarcina $C_0 = H \cdot C_{in} = 1 \cdot 4 = 4$ rezultă capacitatea de sarcină pentru primul nivel (NAND4), $C_{in_2} = (4 \cdot 5/3)/1,83 = 3,64$ valoare care este mai mică decât capacitatea de intrare la primul nivel $C_{in} = 4$. Această valoare subunitară pentru $h_1 = 3,64/4 = 0,91$ apare datorită faptului că s-a egalizat efortul pe fiecare nivel, iar nivelul unu are o poartă care prezintă un efort logic $g_1 = 2$, rezută că $h_1 g_1 = 0,91 \cdot 2 = 1,82$.

Valorile capacităţilor de intrare C_i, calculate pentru fiecare nivel, se vor repartiza pe dimensiunile porţilor tranzistoarelor de intrare.

Din analiza structurii unui decodificator rezultă: decodificatorul este structurat pe un nivel AND, sau **decodificarea este o funcţie logică AND** (**identificare de cod/mintermi**). Un DCDn:2^n generează intern, şi sunt disponibili la ieşire, toţi cei 2^n mintermi de n variabile. În consecinţă, **un DCDn : 2^n poate fi utilizat pentru implementarea unei funcţii logice cu n variabile, dată sub forma FND** (sumă de termeni canonici produs), deoarece generează toţi termenii canonici produs de n variabile, iar nivelul OR al implementării se adaugă din exterior (o poartă OR care colectează termenii canonici care au valoarea 1). O formă disjuntivă FD, pentru implementare trebuie întâi extinsă la o formă FND (sumă de termeni canonici) deoarece la ieşirea decodificatorului oricum sunt disponibili toţi termenii cononici produs.

Circuitele decodificator, pe lângă suportul de nivel AND în implementarea funcţiilor logice, poate modela şi o funcţie aritmetică. Dacă cuvântul de intrare X, cu lungimea de n biţi şi cuvântul generat pe ieşire Y, cu lungimea de 2^n biţi, sunt interpretate ca numere întregi, exprimate în binar natural, atunci între aceste două numere există relaţia:

$$Y = 2^X \tag{2.14}$$

adică, decodificatorul generează la ieşire un număr care este egal cu doi la o putere egală cu numărul aplicat la intrare. Deoarece funcţia de exponenţiere 2.14 este inversa funcţiei de logaritmare 2.10 — se confirmă afirmaţia de la începutul acestui paragraf, aceea că: **decodificarea şi codificarea sunt fiecare funcţia inversă a celeilalte (codifcarea este un nivel OR, iar decodificarea este un nivel AND)**.

Identificarea de cod efectuată de decodificator se realizează pe un nivel AND. De exemplu, poarta AND care implementează produsul logic $x_3\bar{x}_2\bar{x}_1x_0$ este un identificator pentru codul 1001 deoarece numai pentru acest cuvânt generează 1 pe ieşire. Pentru implementări se recomadă, vezi secţiunea 2.3, operatorii negaţi, deci o poartă NAND în loc de AND ceea ce înseamnă că ieşirile decodificatorului sunt active în 0 şi nu în 1 (O_i_L). Dar ca identificator de cod poate fi utilizată şi poarta NOR, care generează la ieşire 1 numai când intrările sunt 0, deci faţă de identificatorul de cod pe baza AND/NAND la cel pe bază de NOR intrările sunt active în 0 şi nu în 1 (I_i_L).

O organizare de decodificator, ca identificator de cod pe bază de NOR, este cea cunoscută sub numele de decodificator Lyon-Schediwy [Sutherland ′99]. La acest decodificator toate ieşirile sunt în L în afară de una, cea activă, care este în H. O poartă NOR CMOS realizează cu uşurinţă ieşirea L (prin tranzistoarele nMOS conectate în paralel din ramura înspre V_{SS}), dar sunt dificultăţi să genereze la ieşire un nivel H prin toate tranzistoarele pMOS înseriate din ramura conectată la V_{DD}. Deoarece la

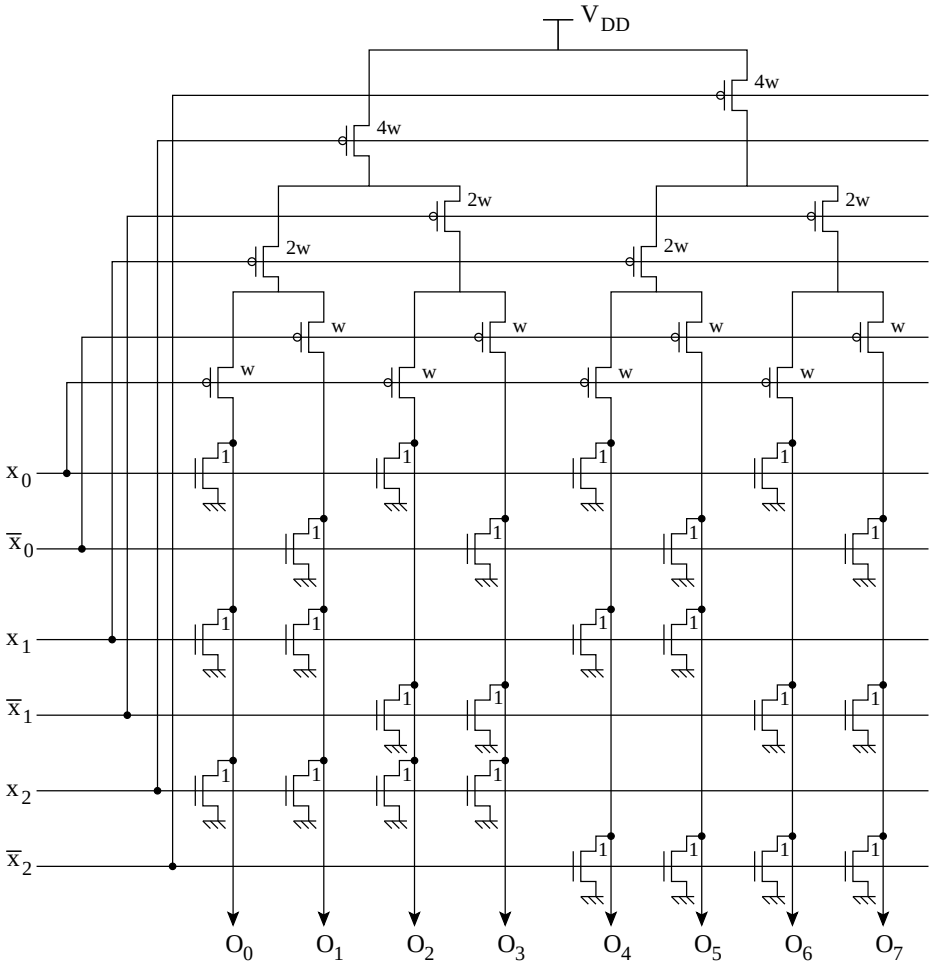


Figura 2.30 Decodificator 3:8 tip Lyon-Schedwy (identificator de cod pe bază de porţi NOR).

acest codificator tot timpul numai o ieşire este în starea H, (deci toate celelalte reţele de tip pMOS ale porţilor NOR nu conduc) este posibil ca unele din tranzistoarele de tip pMOS să intre în structura mai multor porţi NOR, prin aceasta se reduce numărul de tranzistoare. În figura 2.30 este structurat, în acest sens, un DCD3:8 care este compus din opt identificatoare de cod, opt porţi NOR. Mai mult, pentru a asigura efort logic egal pentru cele trei semnale de intrare x_2, x_1 şi x_0 (negate şi nenegate) lăţimea tranzistoarelor de canal nMOS este unitate iar lăţimea celor cu canal pMOS este mai mare de W (lăţimea de canal) ori înmulţit cu puterile lui 2. Cu o astfel de structurare şi cu tranzistoarele dimensionate în acest mod, decodificatorul Lyone-Schediwy are performanţe de timp mult mai bune decât organizarea de codificator ca în Figura 2.28-c la care, de fapt, într-o implementare poarta AND4 nu este un singur nivel ci este substituită printr-un arbore cu două sau trei niveluri (Exemplul 2.12).

Exemplul 2.13 Celula sumator complet, a cărei funcţionare este dată în tabelul 1.6, să se implementeze pe un circuit DCD3:8.

Soluţie. Din Tabelul 1.6 prin sinteză pe bază de 1, forma FCD, rezultă:

$$s_i = \sum_0^7 (1,2,4,7), \quad C_i = \sum_0^7 (3,5,6,7)$$

sau expresiile complementare:

$$\bar{s}_i = \sum_0^7 (0,3,5,6), \quad \bar{C}_i = \sum_0^7 (0,1,2,4)$$

Pentru implementare cu un DCD3:8, care generează toţi cei opt mintermi de trei variabile, va fi necesar doar să se adauge în exterior nivelul de OR care colectează termenii ce au valoarea 1. Se va face implementarea atât pentru cazul când ieşirile decodificatorului sunt active în H cât şi în L. În aceste sens se vor transforma potrivit expresiile pentru s_i şi C_i.

1. Implementare pe bază de DCD3:8 cu ieşirile active în H

a) Expresiile s_i şi C_i se scriu:

$$s_i = P_1 + P_2 + P_4 + P_7; \qquad C_i = P_3 + P_5 + P_6 + P_7$$

Implementarea se reduce la colectarea mintermilor respectivi printr-o poartă OR4, adăugată în exterior, ca în Figura 2.31-a.

b) Uneori funcţia negată se sintetizează prin mai puţini minterimi (nu este cazul pentru celula sumator complet) ceea ce poate constitui un avantaj, micşorarea dimensiuni porţii adăugate în exterior. Expresiile pentru $\bar{s}_i$ şi $\bar{C}_i$ se scriu

$$\bar{s}_i = P_0 + P_3 + P_5 + P_6 \rightarrow \overline{\bar{s}_i} = \overline{P_0 + P_3 + P_5 + P_6}$$

$$\bar{C}_i = P_0 + P_1 + P_2 + P_4 \rightarrow \overline{\bar{C}_i} = \overline{P_0 + P_1 + P_2 + P_4}$$

pentru care corespunde implementarea din Figura 2.31-b. Colectarea mintermilor corespunzători se face printr-o poartă NOR4 adăugată în exterior.

2. Implementarea pe bază de DCD $3:8$ cu ieşirile active în L

a) Expresile s_i şi C_i devin:

$$\overline{\bar{s}_i} = \overline{\overline{P_1 + P_2 + P_4 + P_7}} = \overline{\bar{P}_1 \cdot \bar{P}_2 \cdot \bar{P}_4 \cdot \bar{P}_7}$$

$$\overline{\bar{C}_i} = \overline{\overline{P_3 + P_5 + P_6 + P_7}} = \overline{\bar{P}_3 \cdot \bar{P}_5 \cdot \bar{P}_6 \cdot \bar{P}_7}$$

Colectarea în exterior a mintermilor corespunzători se face printr-o poartă NAND4, Figura 2.31-c.

b) Expresiile negate pentru $\bar{s}_i$ şi $\bar{C}_i$ se scriu sub forma:

$$\bar{s}_i = P_0 + P_3 + P_5 + P_6 \rightarrow \overline{\bar{s}_i} = \overline{P_0 + P_3 + P_5 + P_6} = \bar{P}_0 \cdot \bar{P}_3 \cdot \bar{P}_5 \cdot \bar{P}_6$$

$$\bar{C}_i = P_0 + P_1 + P_2 + P_4 \rightarrow \overline{\bar{C}_i} = \overline{P_0 + P_1 + P_2 + P_4} = \bar{P}_0 \cdot \bar{P}_1 \cdot \bar{P}_2 \cdot \bar{P}_4$$

De data aceasta colectarea în exterior a mintermilor se face printr-o poartă AND4, Figura 2.31-d.

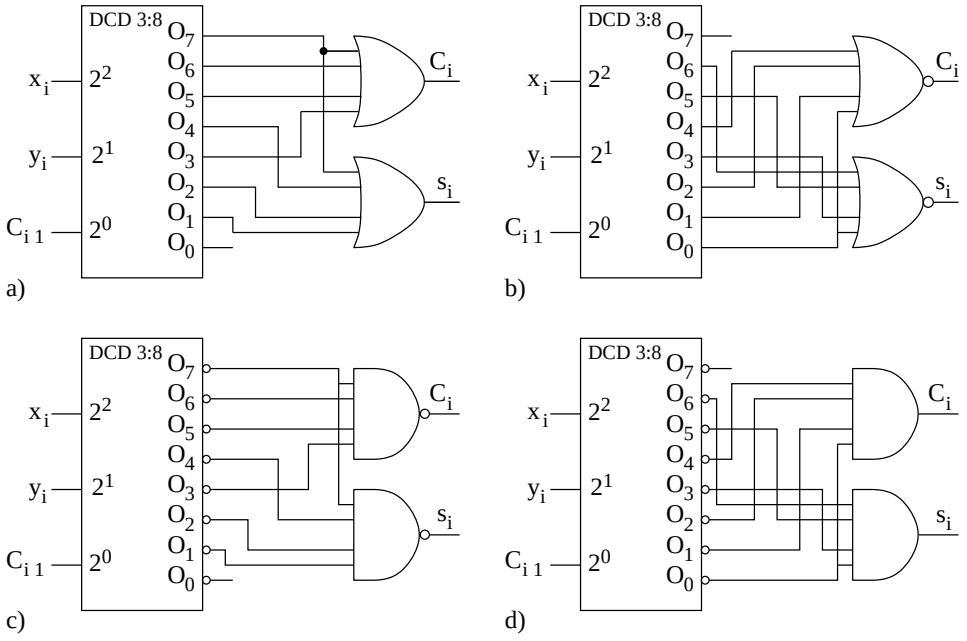


Figura 2.31 Implementarea celulei sumator complet pe un DCD3:8: a,b) pe decodificator cu ieşiri active în H; c,d) pe decodificator cu ieşiri active în L.

Anterior s-a structurat DCDn:2^n cu adâncimea constantă ($D(n) = 3$), în mod inductiv, pornind de la DCDE şi porţi AND adăugând succesiv câte un bit la cuvântul de intrare şi s-au obţinut caracteristicile de adâncime şi dimensiune exprimate prin relaţiile 2.13. Se poate face o structurare şi definire a DCDn:2^n, pornind în sens invers, divizându-l întâi în două DCD$n/2$:$2^{n/2}$ iar ieşirile acestor decodificatoare sunt intrările pe liniile şi coloanele unei matrice formată din $2^{n/2} \cdot 2^{n/2}$ porţi AND2, Figura 2.32. Cuvântul de intrare $x_{n-1}x_{n-2}...x_{n/2}x_{n/2-1}...x_1x_0$ se separă în două semicuvinte, semicuvântul $x_{n-1}x_{n-2}...\ x_{n/2}$ va genera o ieşire activă pe liniile matricei iar semicuvântul $x_{n/2-1}...x_1x_0$ va genera o ieşire activă pe coloanele matricii, în consecinţă poarta AND2 de la intersecţia liniei şi coloanei active va avea ieşirea 1, restul de $2^n - 1$ porţi vor avea ieşirea în zero. Această modalitate de decodificare matriceală (**decodificare bidimensională**) se aplică şi la decodificatoarele rezultate; fiecare din cele două decodificatoare DCD$n/2$:$2^{n/2}$, unul pentru linii iar altul pentru coloane, se structurează la rândul său prin divizarea în două decodificatoare DCD$n/4 : 2^{n/4}$ unul pentru linii, altul pentru coloane şi $2^{n/2}$ porţi AND2. Se continuă această divizare cu doi, a fiecărui decodificator, până când se ajunge la ultimul nivel de structurare compus din două DCDE care comandă o matrice de 4 porţi AND2. Carecteristicile şi definiţia multiplexorului DCD$n : 2^n$ structurat recursiv sunt date în [Ştefan ′00].

Definiţia 2.9 Un DCD$n : 2^n$ se structurează prin divizare în două DCD$n/2$:$2^{n/2}$ şi o matrice de porţi AND2, de dimensiunea $2^{n/2} \cdot 2^{n/2}$, ale căror intrări sunt toate

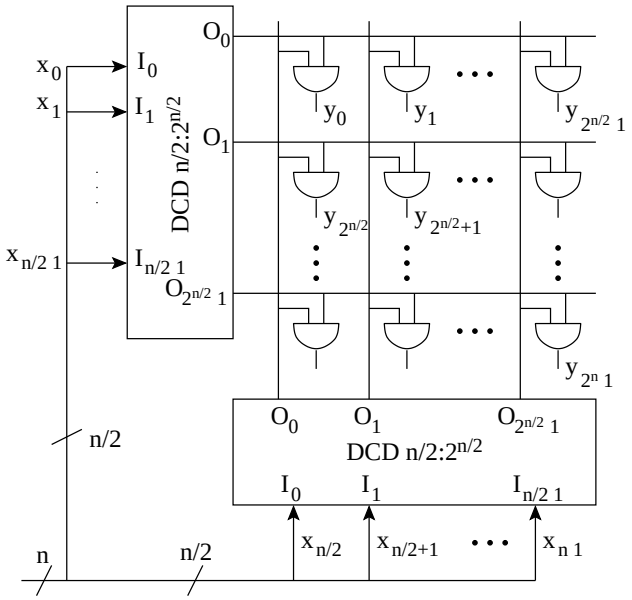


Figura 2.32 Explicativă pentru definirea recursivă a unui DCDn:2^n (decodificare bidimensională).

perechile de ieşiri din cele două decodificatoare. Cele două DCD$n/2$:$2^{n/2}$ de asemenea se structurează prin divizare în acelaşi mod. Procesul de divizare recursivă se opreşte când se ajunge la DCDE. $\diamond$

Dimensiunea definiţiei pentru DCDn:2^n este constantă şi independentă de valoarea lui n deci conform definiţiei 2.1 decodificatorul nu este un circuit complex ci unul simplu, deci realizabil. Dar pentru a declara realizabilitatea circuitului trebuie luată în considerare şi dimensiunea sa. Cu valorile pentru decodificatorul elementar $D_{DCD}(1) = 2$, $S_{DCD}(1) = 3$ şi ţinând cont că prin Definiţia 2.9 după fiecare divizare se generează două DCD$n/2 : 2^{n/2}$ şi 2^n porţi AND2 se poate calcula dimensiunea $S_{DCD}(n)$ şi adâncimea $D_{DCD}(n)$ în felul următor:

$$\begin{aligned} S_{DCD}(n) = &\ 2 \cdot S_{DCD}(n/2) + 2^n \cdot 2 = \\ = &\ 2 \cdot [2 \cdot S_{DCD}(n/4) + 2^{n/2} \cdot 2] + 2^n \cdot 2 = ... && \in O(2^n) \\ D_{DCD}(n) = &\ D_{DCD}(n/2) + D_{AND}(2) = [D_{DCD}(n/4) + 1] + 1 = ... && \in O(log\ n) \end{aligned}$$

Comparând caracteristicile acestei structuri (compusă numai din DCD(1) şi AND2) cu cele alestructurii de adâncime constantă (compusă numai din DCD(1) şi ANDn) se constată că dimensiunea a scăzut de la $O(n \cdot 2^n)$ la $O(2^n)$ iar adâncimea a crescut de la $O(1)$ la $O(\log n)$ deci, teoretic, această structurare este mai uşor realizabilă în schimb asigură performanţe de viteză mai scăzute (Degradarea performanţei de viteză cu mai mult decât s-a micşorat dimensiunea corespunde corelării dintre $S(n)$ şi $D(n)$, relaţia 2.9). Dar, în practică, pentru implementările de tip VLSI un DCD cu adâncimea constantă, ca în Figura 2.28-c, nu poate fi realizat, porţile ANDn se structurează cu porţi mai simple sub formă de arbore. De asemenea, şi la structurarea prin

divizări porţile AND2 formează un arbore. Soluţia de mediere ar fi structurare prin divizare a $DCD(n)$, dar după prima sau a două divizare DCD($n/2$) sau DCD($n/4$) să fie un decodificator de tip Lyon-Schedwy.

Decodificatoarele eficiente din punct de vedere al timpului de propagare sunt necesare pentru adresarea memoriilor şi pentru selectarea registrelor din băncile de registre ale microprocesoarelor. Structurile de decodificator pentru aceste aplicaţii ajung să aibă o valoare de efort total de valori ridicate (vezi Exemplul 1.30). Dar aspectele care trebuie luate în considerare la proiectarea decodificatoarelor pentru astfel de aplicaţii sunt numeroase iar minimizarea efortului logic, pentru a obţine şi o viteza, nu este exclusiv. De exemplu, consideraţiile de layout sunt importante deoarece adesea decodificatorul trebuie să se încadreze în suprafaţa repartizata pentru memorie, suprafaţa sa poate fi limitată, deci iată o constrângere. De asemenea limitarea puterii disipate, poate fi o altă constrâgere importantă, de obicei un decodificator rapid poate necesita o putere disipată ridicată sau prea multe tranzistoare.

În realizarea de sisteme pe placa cu circuite discrete pentru decodficatoarele DCDn:2^n, cu n intrări şi 2^n ieşiri, referite şi cu termenul decodificator/multiplexor, (vezi secţiunea 2.4.5) există următoarele circuite MSI: 74xx138 - DCD3:8; 74xx139 - 2× DCD2:4; 74xx154 - DCD4:16. Alte circuite decodificatoare pot să nu aibă definite toate cele 2^n cuvinte de n biţi pe intrare sau pot să nu utilizeze toate cele 2^n ieşiri cum este cazul următoarelor circuite integrate: 74xx42 - Decodificator din BCD în zecimal; 74xx47 şi 74xx49 - Decodificatoare din BCD pentru elemente de afişaj cu 7 segmente (vezi Exemplul **2.14**).

2.4.3.1 Convertorul de cod

Convertorul de cod este un circuit combinaţional care pentru un cuvânt de n biţi aplicat la intrare generează un cuvânt de m biţi la ieşire, deci circuitul realizează o aplicaţie (mapare) între o mulţime a cuvintelor de intrare cu valori în mulţimea cuvintelor de ieşire. Definirea în acest mod a circuitului convertor de cod corespunde cu definiţia dată în secţiunea 2.1 circuitului combinaţional cu ieşiri multiple şi este reprezentat ca schema bloc in Figura 2.1-a. Rezultă că oricare CLC cu ieşiri multiple este, în fond, un convertor de cod (transcodor).

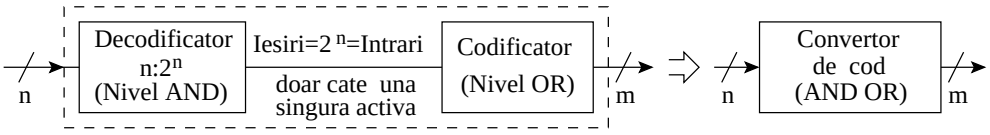


Figura 2.33 Organizarea de principiu pentru un convertor de cod.

Structura unui convertor de cod constă dintr-o înseriere decodificator-codificator, adică două nivele logice AND-OR, Figura 2.33. Codul de intrare de n biţi este aplicat nivelului de decodificare AND rezultând o singură ieşire activă din cele 2^n. Această ieşire activată este aplicată ca o intrare pentru nivelul de codificare OR care generează un cuvânt de m biţi pe ieşire. De fapt, înserierea decodificator-codificator formează un circuit cu cele două niveluri logice necesare pentru modelarea oricărei funcţii logice sub formă de sumă de produse (AND-OR).

2.4.4 Multiplexorul

Circuitul multiplexor realizează în primul rând o funcţie de comunicaţie prin selectarea oricărei linii de intrare I_i, dintr-un număr de 2^n linii ($I_{2^n-1}, I_{2^n-2}, ..., I_i, ...I_1, I_0$) şi conectarea acesteia la o singură ieşire, de unde, uneori, este referit ca **circuit selector**. Funcţia de comunicaţie — selectarea — realizată cu circuit multiplexor este similară cu cea realizată de către un selector rotativ mecanic, Figura 2.34-a. Selectorul rotativ mecanic conectează la ieşire, în funcţie de poziţia contactului rotativ, una dintre liniile de intrare. La circuitul multiplexor, cu 2^n intrări de date şi o ieşire, reprezentat ca schema bloc în Figura 2.34-b (notat prin MUX2^n:1), selectarea liniei de intrare de date I_i, pentru conectarea acesteia la ieşirea O, se face prin aplicarea pe intrările de selectare a indexului i exprimat ca număr în binar natural (cuvânt de selectare). Evident, pentru a putea selecta toate intrările numărul **cuvintelor de selectare** ($S_{n-1}S_{n-2}...S_i...S_1S_0$) distincte trebuie să fie mai mare sau egală cu numărul total al intrărilor. În general, circuitul MUX2^n:1 pe lângă cele 2^n intrări de date şi un cuvânt de selectare, cu lungimea minimă de n biţi, mai prezintă şi o intrare de validare E (în general activă în stare L, $E = \overline{E}\_L$).

Particularizând MUX2^n:1 la $n = 2$ se obţine multiplexorul cu patru căi de intrare (I_3, I_2, I_1, I_0), selectabile cu un cuvânt de doi biţi S_1, S_0, cu schema bloc şi tabelul de adevăr prezentate în Figura 2.34-c şi 2.34-d. Din tabelul de adevăr a MUX4:1 se deduce următoarea funcţie logică

$$y = \overline{E}\_Ld_3x_1x_0 + \overline{E}\_Ld_2x_1\overline{x_0} + \overline{E}\_Ld_1\overline{x_1}x_0 + \overline{E}\_Ld_0\overline{x_1x_0} \tag{2.15}$$

şi este similară cu cea exprimată de relaţia 1.82 sau 1.83. Această expresie este forma canonică normală disjunctivă pentru oricare dintre cele 16 funcţii de două variabile. Structura circuitului MUX4:1 este cea din Figura 2.34-e. În această structură se distinge clar un circuit decodificator 2:4, ca cel din Figura 2.28-c, ale cărui intrări x_1, x_0 sunt biţii cuvântului de selectare iar cele patru ieşiri din porţile AND sunt colectate dintr-un nivel OR. Se poate interpreta că cele patru porţi AND ale decodificatorului, prin intrările de selectare şi cea de validare E, selectează care din cele patru intrări de date (d_3, d_2, d_1, d_0) este conectată la ieşire prin nivelul adăugat OR.

Un circuit DCD care este un nivel de AND, ce generează toţi mintermii canonici de n variabile, poate modela funcţii logice prin colectarea în exterior prin porţi OR a termenilor canonici care au valoarea 1 pentru funcţia respectivă (vezi Exemplul 2.13). Circuitul multiplexor, care este pe lângă nivelul de AND şi un nivel de OR (intern), colectează în interior toţi mintermii canonici de cele n variabile de selectare, deci poate modela oricare funcţie de n variabile rezultând astfel ca un circuit logic universal. Valoarea 0 sau 1 a unui termen canonic P_i în exprimarea funcţiei se fixează prin valoarea coeficientului d_i aplicat pe intrarea de date I_i.

Din structura MUX4:1, prin eliminarea intrării de selectare x_1, se obţine MUX2:1 cu un singur bit de selectare x_0 şi două intrări de date I_1, I_0, care este referit ca **multiplexorul elementar MUXE**, Figura 2.34-f. Dacă se consideră intrarea de validare permanent activă ($E\_L = 0$) se constată că structura de MUXE este identică cu cea a celulei pentru implementarea recurentă, sub formă de arbore, a oricărei funcţii logice dezvoltată recurent, Figura 2.6-a. Pornind de la structurarea MUX2:1 şi MUX4:1 se poate extinde structurarea la un MUX2^n:1, care se compune dintr-un DCDn:2^n cu adâncime constantă, similar ca structură cu cel din Figura 2.28-c,

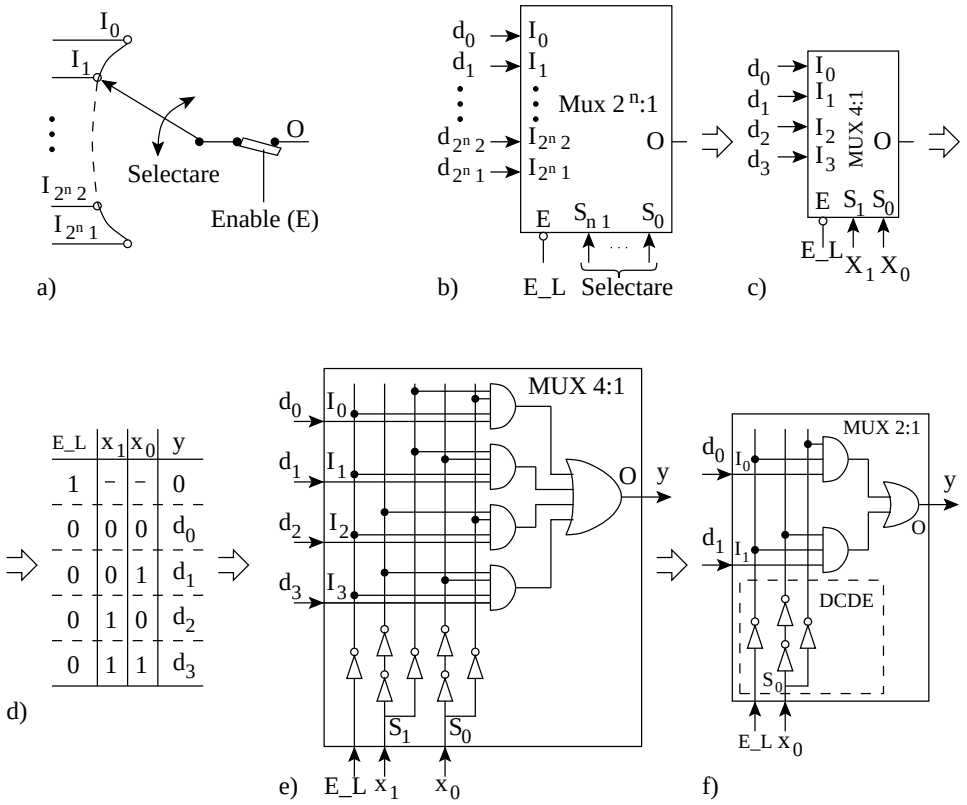


Figura 2.34 Multiplexorul, MUX: a) selectorul rotativ mecanic - analogul mecanic al multiplexorului; b) simbol de reprezentare pentru MUX2^n:1; c,d,e) schema bloc, tabelul de adevăr şi structura circuitului MUX4:1; f) structura multiplexorului elementar, MUX2:1.

şi o poartă OR cu 2^n intrări. DCDn:2^n din cadrul MUX2^n:1 se compune din n decodificatoare elementare DCDE şi 2^n porţi AND cu $n+1$ intrări (n intrări de la cuvântul de selectare, E şi o intrare de date d_i). Pentru structurarea în acest mod a MUX2^n : 1, referită multiplexor cu adânacime constantă, se pot calcula caracteristicile de dimensiune $S_{MUX(n)}$ şi adâncime:

$$S_{MUX(n)} = \begin{array}{l} n \cdot S_{DCDE} + 2^n \cdot S_{AND(n+1)} + 1 \cdot S_{OR(2^n)} = \\ n \cdot 3 + 2^n \cdot (n+1) + 1 \cdot 2^n \in O(n \cdot 2^n) \end{array}$$

Adâncimea are două valori, prima pentru transferul pe traseul cuvânt de selectare-ieşire care este de patru niveluri logice şi a doua pe traseul de intrare de date I_i la ieşire care este de două niveluri logice. Această diferenţă trebuie luată în considerare la comanda multiplexorului dacă semnalele pe intrările de date şi pe intrările de selectare nu se aplică simultan, atunci este recomandat ca cele de selectare să fie primele aplicate deoarece parcurg patru niveluri faţă de două pentru cele de date. De fapt, aceste caracteristici de dimensiune şi adâncime sunt, teoretic, valori minime asimptotice. Practic, pentru n de valoare ridicată este dificil de realizat porţi AND(n+1) şi OR2^n

pe un singur nivel logic; în plus, mai ales în tehnologia CMOS timpul de propagare depinde pregnant de încărcare (vezi secţiunea 1.5.6).

Definiţia 2.10 Un MUX 2^n:1, selectat prin cuvântul $x_{n-1}x_{n-2}...x_1x_0$, se structurează dintr-un MUXE selectat cu x_{n-1} şi are ca intrări de date ieşirile de la două MUX2^{n-1}:1 selectate prin cuvântul $x_{n-2}...x_1x_0$. Apoi, asupra celor două multiplexoare MUX2^{n-1}:1 se aplică acelaşi procedeu de structurare. Procedeul recursiv se opreşte când se ajunge la MUX-uri selectate prin x_0. $\diamond$

Prin structurarea unui MUX2^n:1, conforma Definiţiei 2.10 , se obţine o reţea arborescentă de $(2^n - 1)$MUX2:1 având n niveluri, Figura 2.35-a. Structura aceasta de reţea este identică cu structura arborescentă pentru implementarea unei funcţii canonice normale disjunctive dezvoltate recurent din Figura 2.7. Cele 2^n intrări de date ale MUX2^n:1 sunt intrările în multiplexoarele elementare "frunze" iar ieşire este ieşirea multiplexorului elementar "rădăcină"; se observă că transferul este pe trasee de la frunze înspre rădăcină. Pe fiecare traseu se parcurge acelaşi număr de MUXE. Această structură recursivă a unui MUX2^n:1 are următoarele caracteristici de dimensiune şi adâncime: $D_{MUX(n)} \in O(n),\ S_{MUX(n)} \in O(2^n)$.

Implementarea logică a multiplexoarelor în circuitele VLSI apare foarte simplă când se utilizează elemente de trecere nMOS, porţi de transmisie CMOS. Pornind de la funcţia sa de transmisie, multiplexorul, poate fi conceput din 2^n ramuri de transmisie care au un capăt comun (ieşirea) iar la celălalt capăt al fiecărei ramuri se aplică data corespunzătoare d_i care trebuie transmisă. Fiecare ramură este compusă din n elemente de trecere înseriate (circuit AND) care va fi deschisă doar pentru o singură configuraţie a biţilor din cuvântul de intrare $x_{n-1}x_{n-2}...x_1x_0$. În acest mod de implementare, un multiplexor elementar este compus numai din două ramuri pe care există câte un element de trecere, unul dintre acestea este comandat de x_0 iar celălalt de $\bar{x}_0$, semnalele de comandă fiind generate de un DCDE.

O astfel de structură de MUX4:1 implementată cu porţi de transmisie este prezentată în Figura 1.53-c. În structura din această figură există redundanţă, ultimele două porţi, una pe ramura lui d_{i_0} şi cealaltă pe ramura lui d_{i_2}, au un terminal comun (ieşirea) şi sunt comandate cu acelaşi semnal $\bar{B}$, deci pot fi substituite printr-o singură poartă (cele două ramuri se unesc înainte de această poartă şi împreună prin această ajung la terminalul comun). Aceeaşi simplificare se poate face şi cu porţile comandate de B de pe liniile d_{i_1} şi d_{i_3}. Va rezulta o reţea arborescentă cu două MUXE selectate de A, $\bar{A}$ urmate de un alt MUXE selectat de B, $\bar{B}$. În Figura 2.35-b este prezentată structurarea unui MUX8:1, sub formă de reţea arborescentă din MUXE, realizată cu tranzistoare de trecere nMOS. Se observă că acest circuit se obţine prin maparea 1:1, cu MUXE cu tranzistoare de trecere, a structurii recurente MUX2^n:1, din Figura 2.35-a, prin particularizarea pentru $n = 3$. Implementările sub formă de reţea arborescentă sunt recomandate în sistemele integrate de performanţă ridicată şi de putere redusă. Totuşi, pentru n de valoare ridicată timpul de propagare pe traseele de la frunze la rădăcină devine foarte mare. În lungul lungul unui traseu cu n tranzistoare de trecere întârzierea este egală cu n^2RC, unde RC este întârzierea dată de reţeaua echivalentă a tranzistorului de trecere.

În Definiţia 2.10 structurarea prin recurenţă s-a specificat cu un pas egal cu unu, adică pentru câte un bit al cuvântului de selectare rezultând în fiecare nivel numai MUXE. Dar se poate face structurarea pe grupuri de biţi din cuvântul de selectare, de exemplu, dacă se iau grupuri de câte doi sau trei biţi rezultă pentru nivelurile

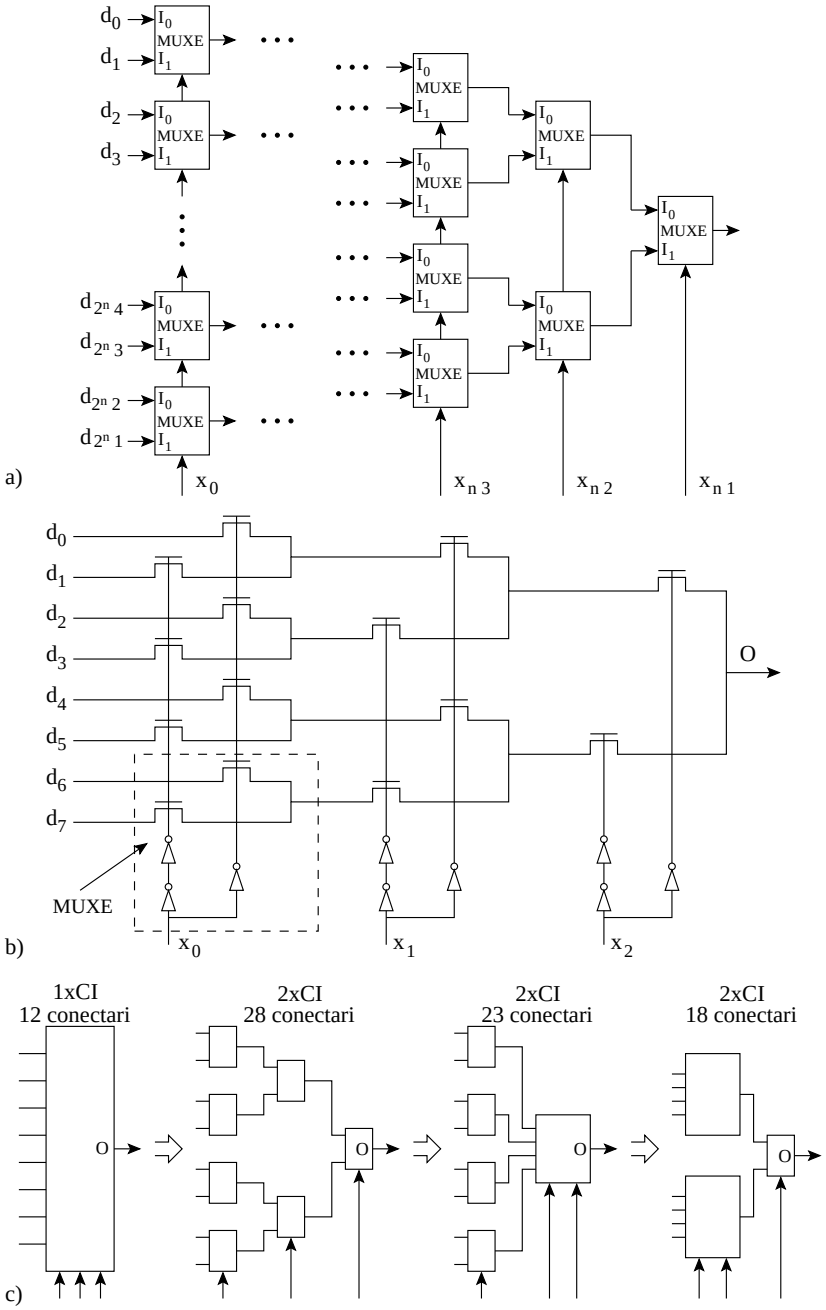


Figura 2.35 Structurarea arborescentă a unui multiplexor conform Definiţiei 2.10: a) MUX2^n:1 structurat din 2^n MUX2:1; b) MUX8:1 realizat cu tranzistoare de trecere; c) variante de structurare a MUX8:1 pe bază de MUX4:1 şi MUX2:1

respective din reţeaua arborescentă MUX2^2:1, MUX2^3:1 în loc de MUXE. Varianta aceasta de structurare arată posibilitatea de obţinere a multiplexoarelor de capacitate mare din circuite multiplexoare de capacitate inferioară, de exemplu pe baza celor mai utilizate în practică MUX4:1 sau MUX8:1. În Figura 2.35-c sunt prezentate variante de compunere a unui MUX8:1 din altele de capacităţi mai mici (4:1 şi 2:1). Care variantă este mai bună? Pentru a răspunde la această întrebare trebuie introdus un criteriu de evaluare. Acest criteriu de evaluare ia în considerare numărul de circuite integrate utilizate, CI (există 4× MUX2:1/cip; 2× MUX4:1/cip;1× MUX8:1/cip;) şi costul(care poate fi proporţional cu numărul de terminale/conectări). Numărul de terminale poate determina costul deoarece fiecare terminal trebuie lipit sau este un punct de wrapare. Din cele patru variante de structurare ale MUX8:1/cip apare că pentru criteriul ales cea mai bună variantă este un singur MUX8:1/cip şi cea mai slabă printr-o reţea numai de MUX2:1. Dar să nu ne grăbim, vom vedea că în unele implementări realizarea unui multiplexor de capacitate mai mare din circuite de capacitate 2:1 poate fi cea mai avantajoasă, vezi Exemplul 2.17.

Pentru implementările de sisteme există următoarele multiplexoare sub formă de circuite integrate MSI:

74xx150 − 1× MUX16:1
74xx151 − 1× MUX8:1
74xx153 − 2× MUX4:1, iar 74xx253 - 2 × MUX4:1 are ieşirea TSL
74xx157 − 4× MUX2:1, iar 74xx257 - 2 × MUX2:1 are ieşirea TSL

2.4.4.1 Aplicaţii cu circuite multiplexoare

Circuitul multiplexor poate modela o funcţie de comunicaţie, (a) selectare de date şi o funcţie canonică normală disjunctivă, (b) implementare de funcţii logice. Aplicaţiile cu circuite multiplexoare se reduc la realizarea acestor două tipuri de funcţii.

a). Selectarea datelor. Circuitul MUX2^n:1 poate fi utilizat pentru conversia paralel-serie a unui cuvânt de 2^n biţi. Biţii cuvântului de serializat se aplică pe intrările de date $I_{2^n-1}, I_{2^n-2}, ..., I_i, ..., I_1, I_0$, iar prin configuraţia cuvântului de selectare $x_{n-1}x_{n-2}...x_1x_0$ un bit de pe o anumită intrare de date este aplicat la ieşire. Evident, serializarea biţilor cuvântului de date se poate realizare în oricare secvenţă dorită. Dacă este necesară serializarea succesivă începînd cu I_0 pâna la I_{2^n-1} atunci secvenţele configuraţiilor cuvântului de selectare sunt în ordinea de creştere a numerelor naturale, care se pot obţine de la un numărător modulo 2^n, iar pentru această serializare sunt necesare 2^n tacte de clock.

Uneori în arhitectura unui sistem apare un desen ca cel din Figura 2.36-a care simbolizează transferul unor cuvinte cu lungimea de k biţi de la sursele $P_{2^n-1}, P_{2^n-2}, ..., P_i, ..., P_1, P_0$ (porturi sursă) într-un singur loc de destinţie (care poate fi un port, o magistrală etc.). Acest transfer poate fi inplementat cu k multiplexoare 2^n:1 puse în paralel, toate fiind comandate de acelaşi cuvânt de selectare, Figura 2.36-b. Biţii cuvântului de la portul P_i se aplică, câte unul, numai la intrarea de date I_i de la fiecare dintre cele k multiplexoare, deci un port este legat la toate intrările de date cu acelaşi număr de la cele k multiplexoare. La aplicarea unui cuvânt de selectare i toţii biţii de pe intrările de date I_i ale celor k multiplexoare sunt transferaţi la ieşirile acestora, deci cuvântul cu lungimea de k biţi din portul P_i este transferat în portul

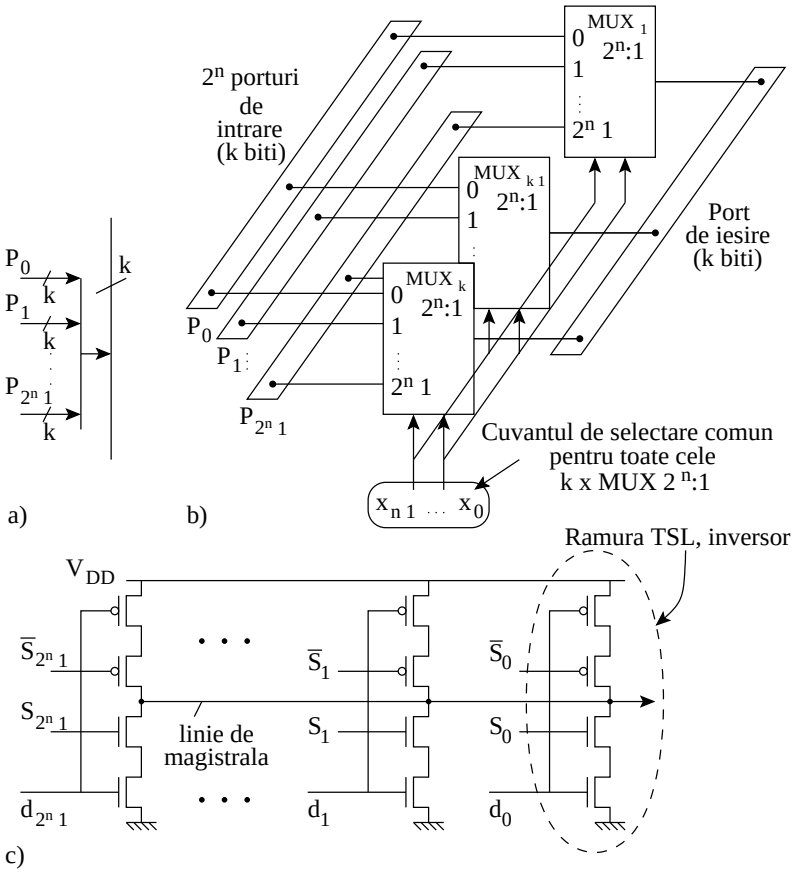


Figura 2.36 Selectarea secvenţială a datalor: a) reprezentare simbolică a unui punct de selectare a datelor dintr-un sistem; b) selectarea unui port de intrare, dintr-un număr de 2^n porturi cu lungimea de k biţi, şi conectarea la un port de ieşire prin intermediul unui grup de $k\times$MUX2^n:1; c) structură în tehnologie CMOS de MUX2^n:1 cu ieşirea TSL pentru comanda unei linii de magistrală.

de ieşire Pentru implementările VLSI când se lucrează cu lungimi de cuvând de 32, 64 biţi, deci tot atâtea multiplexoare puse în paralel, încărcarea electrică a semnalelor se selectare este foarte mare ceea ce implică o bufferare, vezi Exemplul 1.28.

Pentru implementările în tehnologie CMOS o structură recomandată de multiplexor inversor cu 2^n căi este prezentată în Figura 2.36-c, care se compune din ramuri inversor TSL, Figura 1.46-c, toate conectate la o linie de magistrală. Pentru semnale de selectare ale ramurii i, $S_i\bar{S}_i = 01$ (generate de un decodificator), atât ramura n cât şi ramura p ale inversorului CMOS sunt blocate, deci ieşirea la linia de magistrală este în starea HZ, iar pentru $S_i\bar{S}_i = 10$ ramura este în funcţionare normală de inversor, data de intrare este transferată negat la magistrală, $\overline{d_i}$. Efortul logic total al multiplexorului este $2^n(4+4\cdot\gamma)/(1+\gamma) = 4\cdot 2^n$, cu efortul logic pe o intrare de date $(2+2\cdot\gamma)/(1+\gamma) = 2$; aceeaşi valoare 2 este şi pentru o preche $(S_i\bar{S}_i)$ de semnale de

selectare. Rezultă că efortul logic pe o intrare este constant şi nu depinde de numărul de intrări, ceea ce ar sugera că se pot realiza multiplexoare rapide cu oricâte linii de intrare. Totuşi, când lungimea k a cuvintelor selectate este de valoare ridicată perechile de semnale de selectare $S_i\bar{S}_i$ necesită un efort ridicat ($k\cdot$ MUX2^n:1), desi aceasta nu are un efect de marirea întârzierii pentru transferul datei d_i. Când se iau în considerare şi capacităţile parazite se constată că această structură nu este rapidă pentru un număr de intrări (ramuri inversor) ori cât de mare, în general se limitează la structura cu cel mult patru ramuri.

Exemplul 2.14 Să se structureze un sistem pentru afişarea a două cifre zecimale a căror cod BCD este înscris în două porturi A, B. Se vor utiliza elemente afişoare cu 7 segmente cu catodul comun.

Soluţie. Pentru iluminarea unei cifre trebuie conectate simultan la tensiunea $+V$ toţi anozii LED-urilor (segmentelor) care configurează caracterul iar catodul la masă, Figura 2.37. Conectarea simultană a segmentelor luminoase este realizată prin ieşirile (a, b, c, d, e, f, g), active în H, ale unui convertor de cod BCD-7 segmente 7449. Deoarece se utilizează un singur decodificator 7449 pentru ambele elemente afişoare este necesar ca informaţia să fie aplicată alternativ cu o frecvenţă de minim 30Hz (T$\geq$3,33ms) pentru ca, prin inerţia ochiului, cifra să apară afişată continuu. Pentru aplicarea alternativă, cu frecventă de 30Hz a conţinutului celor două porturi sursă A şi B pe intrările convertorului BCD-7 segmente, Tabelul 2.2, secţiunea 2.4.7, se utilizează 4×MUX2:1 ca selector de date (circuitul 74157). Conectarea la masă a catodului comun din fiecare dintre cele două afişoare A şi B se realizează printr-un DCD2 : 4 (1/2 74139) cu ieşirile active în L şi comandat sincron cu ieşirea cuvintelor A şi B prin multiplexor. Sincronizarea este realizată cu semnalul de ceas CLK, care se aplică pe intrarea I_0 a decodificatorului şi pe intrarea comună de selectarea $G1$ a grupului de patru multiplexoare. Pe palierul pozitiv, CLK=1, se afişează cuvântul din portul A iar pe palierul CLK=0 se afişează cuvântul din portul B.

---

b). Implementarea funcţiilor logice. Circuitul multiplexor MUX2^n:1, prin structura sa, produce pe nivelul de AND (decodificator) toţi termenii cononici produs de variabilele de selectare, fiecare minterm P_i fiind înmulţit cu coeficientul d_i respectiv, iar prin nivelul interior de OR însumează toate produsele obţinute, deci poate modela orice funcţie de n variabile sub forma sa canonică normală disjunctivă, relaţia 1.10. Un MUX2^n:1 apare ca suport pentru implementarea tuturor celor 2^{2^n} funcţii de n variabile, particularizarea pentru o anumită funcţie f_j^n se face prin aplicarea pe intrarea de date a setului de 2^n coeficienţi ai funcţiei, deci **multiplexorul este un circuit universal pentru implementarea oricărei funcţii!** Implementarea funcţiei rezultă ca o mapare directă a coeficienţilor funcţiei citiţi din tabelul de adevăr sau diagrama V-K pe intrările de date corespunzătoare ale multiplexorului. Implementarea funcţiei pe multiplexor nu necesită minimizarea acesteia, dimpotrivă, dacă o funcţie este dată în formă redusă/minimă va trebui expandată pâna la obţinerea formei normale disjunctivă din care rezultă care intrări de date ale multiplexorului sunt 1 şi care sunt 0 (adică valorile coeficienţilor funcţiei).

Pentru implementarea celor 16 funcţii de două variabile x_1, x_0 coeficienţii funcţiei f_i^2 se citesc din Figura 1.2-a, fie din relaţia 1.82 (linia a i-a din matricea coeficienţilor) care se aplică unui MUX4:1, Figura 2.38. Din totalul de 16 MUX4:1, care modelează cele 16 funcţii, şase pot fi eliminate deoarece funcţiile pe care le modelează se reduc

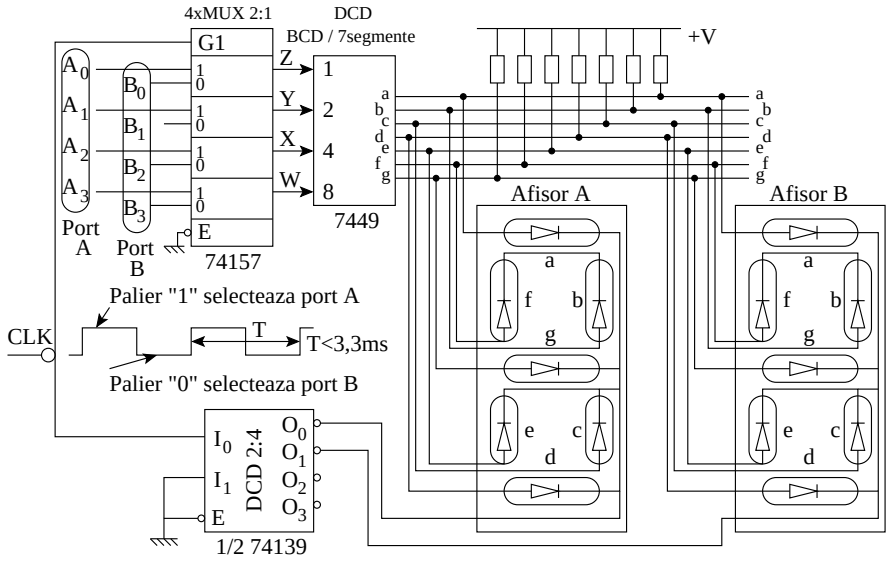


Figura 2.37 Structură pentru multiplexarea informaţiei pe afişoare cu 7 segmente.

la valori banale $(0, 1, x_1, x_0, \bar{x}_1, \bar{x}_0)$, existente în sistem fără a mai fi necesar să fie calculate.

Dar, implementarea celor 16 funcţii de două variabile poate fi efectuată considerând funcţii de o singură variabilă x_0, iar variabila x_1 — variabilă reziduu — este introdusă în coeficienţii funcţiei. Coeficienţii funcţiei pot avea doar valori din mulţimea $\{0, 1, x_1, \bar{x}_1\}$, valori care nu trebuie calculate, deoarece există în sistem. Avantajul practic al reducerii funcţiei la o funcţie de o singură variabilă rezultă printr-o implementare pe un circuit mai simplu, MUX2:1, Figura 2.39. Evident, şi pentru implementarea pe MUX2:1 cele şase valori banale $(0, 1, x_1, x_0, \bar{x}_1, \bar{x}_0)$ ale funcţiilor de două variabile nu necesită multiplexoare.

Pentru această implementare elegantă a unei funcţii pe multiplexor, când n este ridicat, rezultă un multiplexor de capacitate mare. Există două modalităţi de a reduce, eventual, capacitatea multiplexorului. În primul rând, se încearcă să se introducă mai multe variabile reziduu rezultând o funcţie de mai puţine variabile. Această variantă duce la simplificarea structurii numai atunci când coeficienţii obţinuţi cu variabile reziduu au forme banale sau au expresii ce pot fi calculate cu porţi simple AND/NAND, NOR/OR sau XOR/NXOR. Foarte important în obţinerea unor expresii simple pentru coeficienţii cu variabile reziduu este modul de alegere, dintre variabilele funcţiei, care să fie varaibilele reziduu şi care să fie variabilele funcţiei de ordin mai mic. Nu există un algoritm care să exprime modul de alegere al variabilelor pentru o implementare optimă. Pentru o implementare cvasioptimă, sau optimă, intuiţia şi exerciţiul proiectantului în manipularea funcţiilor analitice, tabelelor de adevăr sau diagramele V-K pot fi factori de succes.

În al doilea rând, se poate reduce capacitatea multiplexorului când, fie pentru funcţia exprimată prin variabile reziduu, fie fară, se descompune multiplexorul de

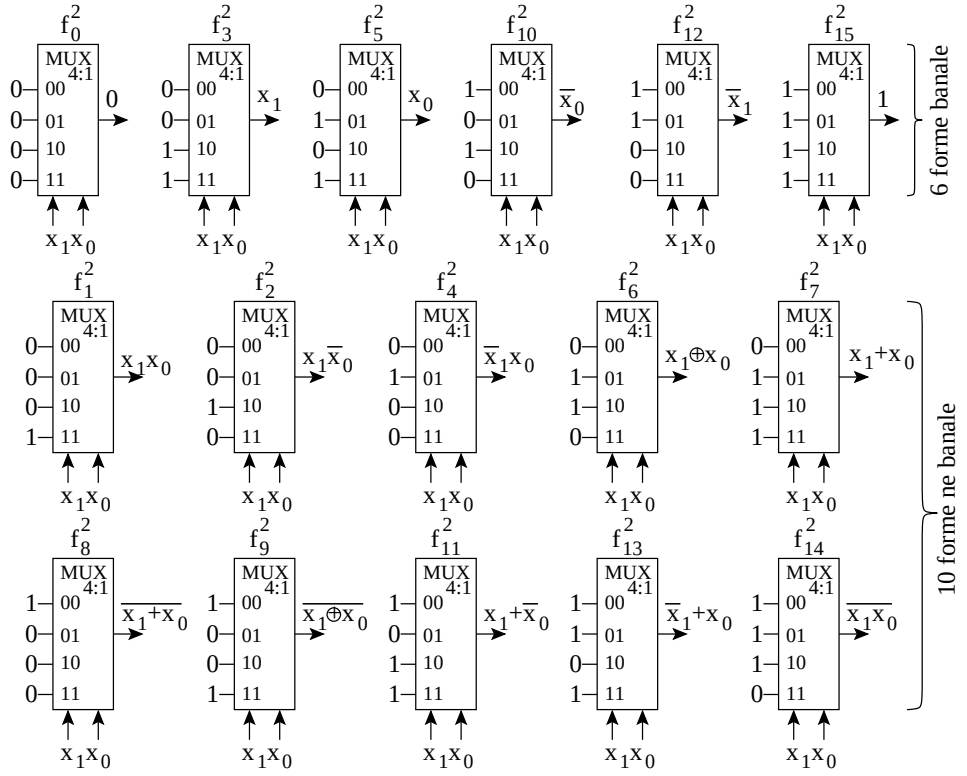


Figura 2.38 Implementarea celor 16 funcţii de două variabile pe circuite MUX4:1.

capacitate ridicată printr-o reţea arborescentă de multiplexoare de capacitate mai mică, uzual din MUXE. Analizând ieşirile acestor multiplexoare elementare, situate la nivelul zero (nivelul frunzelor) sau la niveluri puţin mai ridicate decât acesta, se poate constata că unele dintre aceste niveluri produc valori banale sau funcţii foarte simple, astfel se pot elimina multiplexoarele respective. Poate rezulta în urma analizei o structură hard cu mult mai simplă în raport cu multiplexorul iniţial. Şi la implementarea pe structuri arborescente modul de alegere al variabilelor reziduu poate duce la simplificări hard importante. În exemplele următoare se vor utiliza toate aceste modalităţi pentru implementare.

Exemplul 2.15 Următoarea funcţie

$$f(A,B,C,D,E) = \sum_{0}^{31}(1,2,3,4,5,6,7,10,14,20,22,28)$$

să fie implementată numai cu circuite MUX4:1.

Soluţie. Pentru a utiliza numai MUX4:1 funcţia se va transforma ca o funcţie de două variabile D, E cu coeficienţi care cuprind variabilele reziduu A, B, C. Aceşti coeficienţi vor fi şi ei transformaţi încât să fie funcţii de două variabile B, C iar variabila A introdusă ca variabilă reziduu. Rezultă o structură de reţea arborescentă cu două niveluri, pe nivelul

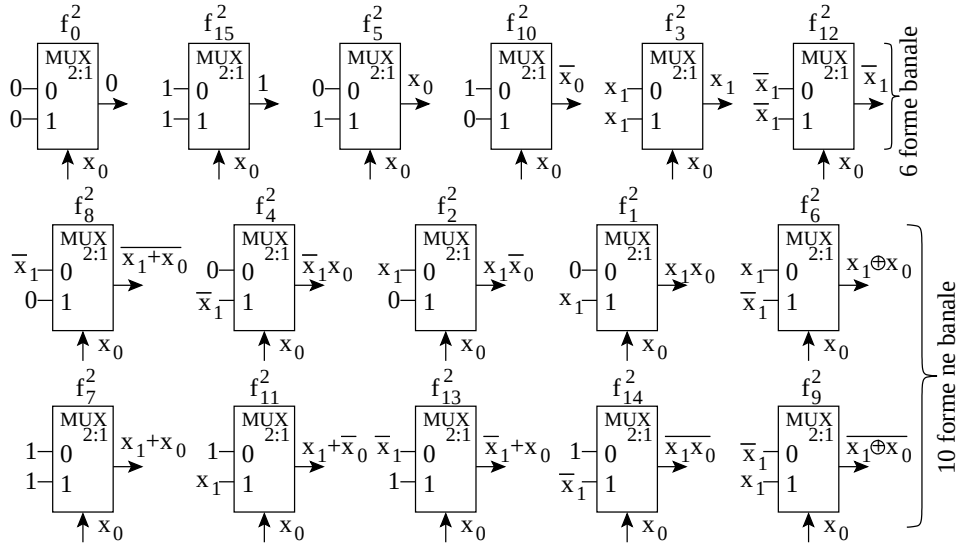


Figura 2.39 Implmentarea celor 16 funcţii de două variabile pe circuite MUX2:1 (variabila x_1 este introdusă ca variabilă reziduu în coeficienţii funcţiilor f_i^2).

zero ("frunze") multiplexoarele 4:1 vor fi selectate de variabilele B, C având ca şi coeficient pe intrările de date valori din mulţimea $\{0, 1, A, \bar{A}\}$ iar pe nivelul unu un multiplexor 4:1 selectat de variabilele D, E

$$f(A,B,C,D,E) = P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_{10} + P_{14} + P_{20} + P_{22} + P_{28} =$$

$$\overline{ABC}DE + \overline{ABC}D\overline{E} + \overline{ABC}DE + \overline{A}B\overline{C}\overline{DE} + \overline{A}B\overline{C}D\overline{E} + \overline{A}BCD\overline{E} +$$
$$+\overline{A}BCDE + \overline{A}B\overline{C}D\overline{E} + \overline{A}BCD\overline{E} + A\overline{B}C\overline{DE} + A\overline{B}CD\overline{E} + ABC\overline{DE} =$$
$$= (\overline{A}B\overline{C} + ABC)\overline{DE} + (\overline{ABC} + \overline{A}B\overline{C})\overline{D}E + (\overline{ABC} + A\overline{B}C + \overline{A}B\overline{C} +$$
$$+\overline{A}BC)D\overline{E} + (\overline{ABC} + \overline{A}BC)DE =$$
$$= (0 \cdot \overline{BC} + 1 \cdot \overline{B}C + 0 \cdot B\overline{C} + A \cdot BC)\overline{DE} +$$
$$+(\overline{A} \cdot \overline{B}\bar{C} + \overline{A} \cdot \overline{B}C + 0 \cdot B\overline{C} + 0 \cdot BC)\overline{D}E +$$
$$+(\overline{A} \cdot \overline{BC} + 1 \cdot \overline{B}C + \overline{A} \cdot B\overline{C} + \overline{A} \cdot BC)D\overline{E} +$$
$$+(\overline{A} \cdot \overline{BC} + \overline{A} \cdot \overline{B}C + 0 \cdot B\overline{C} + 0 \cdot BC)DE$$

Această expresie este implementată în Figura 2.40. Deoarece expresiile coeficienţilor produselor $\bar{D}E$ şi DE sunt identice se va calcula cu un singur MUX4:1, în nivelul zero, iar rezultatul se aplică pe intrările 01 şi 11 ale multiplexorului din nivelul unu.

Exemplul 2.16 Fie următoarea funcţie de 5 variabile

$$f(A,B,C,D,E) = \sum_0^{31}(8,9,10,11,13,15,17,19,21,23,24,25,26,27,29,31).$$

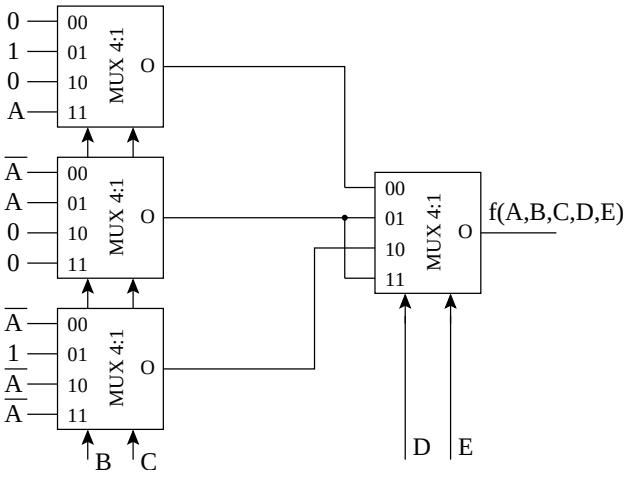


Figura 2.40 Implementarea printr-o structură arborescentă numai cu MUX4:1 a funcţie $\mathbf{F(A,B,C,D,E)} = \sum_{\mathbf{0}}^{\mathbf{31}}(\mathbf{1,2,3,4,5,6,7,10,14,20,22,28})$.

Să se implementeze utilizând un număr minim de circuite integrate. Se presupune că sunt disponibile, la intrare, variabilele atât negate cât şi nenegate.

Soluţie. Din diagrama V-K, Figura 2.41-a rezută forma minimă a funcţiei

$$f = AE + B\bar{C} + BE = \overline{\overline{AE} \cdot \overline{B\bar{C}} \cdot \overline{BE}}$$

care se implementează ca în Figura 2.41-b cu 3xNAND2 şi 1xNAND3. Deoarece nu există circuite integrate care să aibă simultan ambele tipuri de porţi se vor utiliza două circuite integrate: unul pentru 3xNAND2 şi unul pentru 1xNAND3; pe fiecare circuit rămânând porţi nefolosite. Deoarece în forma minimă obţinută nu mai apare variabila D se va implementa în continuare o funcţie numai de 4 variabile A, B, C, E. Pentru implementare pe multiplexor forma minimă se expandează la forma normală disjunctivă, operaţie care este prezntată în diagrama V-K din Figura 2.41-c. Implementarea pe un MUX16:1 se face direct prin maparea coeficienţilor din diagrama V-K pe intrările de date. Dar se poate utiliza un MUX8:1 dacă variabila A se va alege ca variabilă reziduu, Figura 2.41-e (un singur circuit integrat).

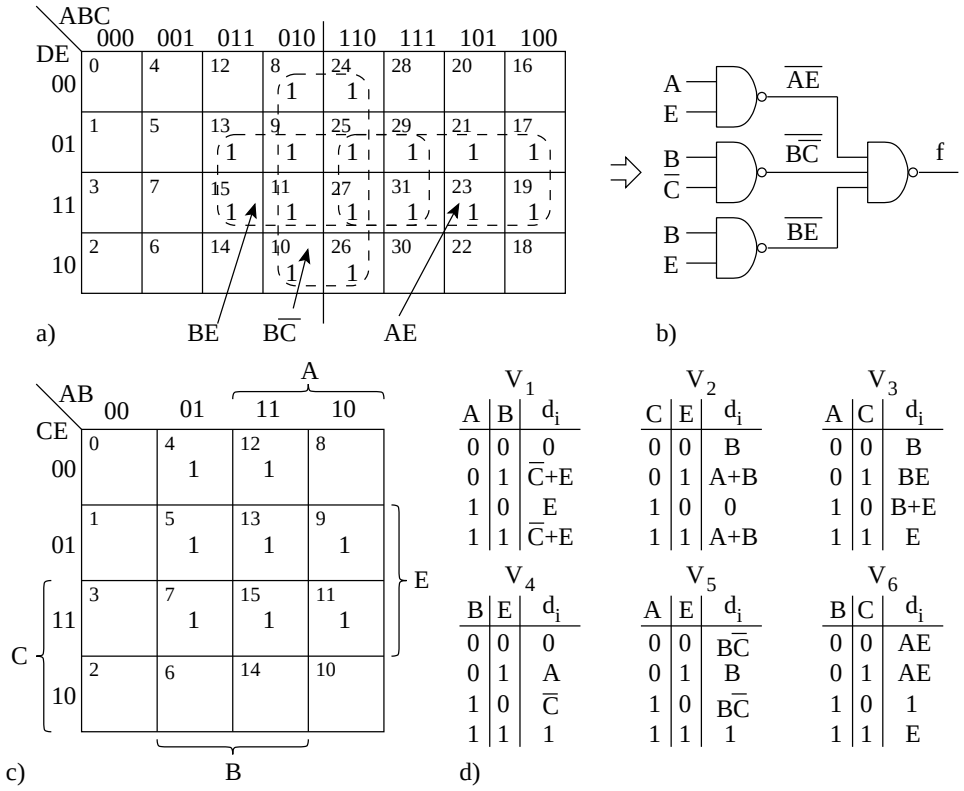
Pentru implemntarea pe MUX4:1 trebuie introduse în coeficienţi două din variabile reziduu. Dar care variantă, de câte două perechi de variabile (o pereche de variabile pentru selectarea multiplexorului iar celalaltă pereche de variabile introduse ca variabile reziduu), duce la implementarea cea mai simplă? În total cu cele patru variabile se pot realiza următoarele şase variante de perechi distincte (prima pereche corespunde variabilelor reziduu iar a doua variabilelor de selectare):

$$V_1 \rightarrow CE, AB; V_2 \rightarrow AB, CE; V_3 \rightarrow BE, AC;$$

$$V_4 \rightarrow AC, BE; V_5 \rightarrow BC, AE; V_6 \rightarrow AE, BC$$

Pentru toate variantele, valorile coeficienţilor cu variabile reziduu se pot deduce din diagrama V-K din Figura 2.41-c. De exemplu, pentru varianta V_1 (variabile de selectare AB) se procedează în felul următor:

- pentru suprafaţa $AB = 00$, toţi mintermii din căsuţele elementare ale acestei suprafeţe au valoarea 0, deci $d_0 = 0$.



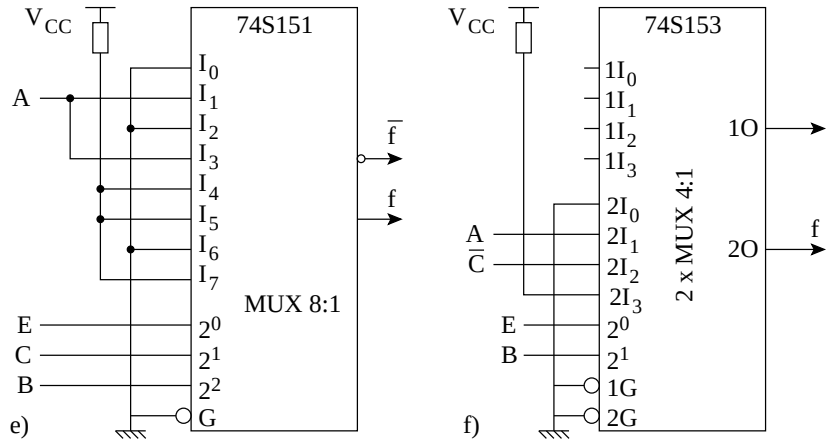


Figura 2.41 Modalităţi de implementare a funcţiei $\mathbf{f(A,B,C,D,E)} = \sum_{\mathbf{0}}^{\mathbf{31}}\mathbf{(8,9,10,11,13,15,17,19,21,23,24,25,26,27,29,31)}$.

- pentru suprafaţa $AB = 01$, mintermii din căsuţele elementare 4,5 şi 7 cuprinse în interiorul acestei suprafeţe au valoarea 1. Aceştia extraşi din diagramă prin scriere împreună în funcţie de C, E rezută $\bar{C}\bar{A}B + E\bar{A}B = (\bar{C} + E)\bar{A}B$ deci $d_1 = \bar{C} + E$.
- pentru suprafaţa $AB = 11$, numai mintermii din căsuţele 12,13 şi 15 au valoarea 1, prin scrierea impreună a lor rezultă $\bar{C}AB + EAB = (\bar{C} + E)AB$, deci $d_3 = \bar{C} + E$.
- pentru suprafaţa $AB = 10$, numai mintermii din căsuţele 9,11 au valoarea 1. Prin scrierea impreună a lor rezultă $EA\bar{B}$, deci $d_2 = E$.

Valorile rezultate ale coeficienţilor cu variabile reziduu (C, E) sunt prezentate în tabelul V_1 din Figura 2.41-d. În mod asemănător se procedează şi pentru celelalte cinci variante. Analizând tabelele tuturor variabilelor rezultă că cea mai simplă formă de implementare corespunde lui V_4, coeficienţii sunt numai din mulţimea valorilor banale $0, 1, \bar{A}, A, \bar{C}, C$. Impelementarea pe circuitul 74S153 (2xMUX4:1) este prezentată în Figura 2.41-f (un singur circuit integrat, dar un MUX4:1 din componenţa sa rămâne neutlizat).

Exemplul 2.17 Funcţia de patru variabile dată în diagrama V-K din Figura 2.42-a să se implementeze pe un MUX8:1 structurat ca în variantele din Figura 2.35-c.

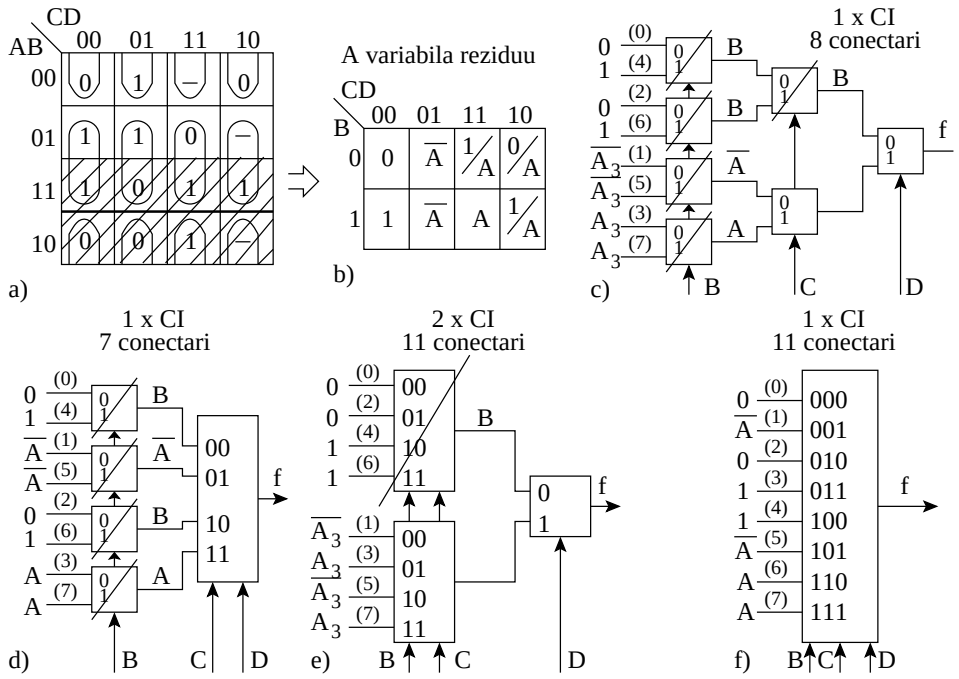


Figura 2.42 Impelementarea unei funcţii de patru variabile pe diferite structurări de MUX8:1.

Soluţie. Funcţia de patru variabile pentru a fi implementată pe un MUX8:1 trebuie exprimată cu o variabilă reziduu, deci diagrama V-K din Figura 2.42-a este transformată în diagrama V-K din Figura 2.42-b. Prin această transformare, pentru valorile indiferente ale lui funcţiei se consideră atât 0 cât şi 1 ceea ce determină două valori ale coeficientului în

unele căsuţe elementare ale diagramei cu variabile reziduu. Pentru implementarea funcţiei se va lua acea valoare a coeficientului care duce la o structură de circuit mai simplă.

Coeficienţii funcţiei din diagrama V-K cu varaibile reziduu se aplică pe intrările structurilor de MUX8:1. Coeficientul funcţiei din căsuţa cu numărul i, exprimat în binar, se aplică pe intrarea de date notată cu numărul i ca în Figura 2.42-f. Dar la o structură de arbore intrarea a i-a este aceea la care parcurgând un traseu de la frunză la rădăcină şirul cifrelor binare, întâlnite la intrarea de date a fiecărui multiplexor, formează tocmai numărul i exprimat în binar natural. De exemplu, în Figura 2.42-c, la intrarea la care începe traseul prin a cărei parcurgere se întâlneşte şirul de biţi $1, 0, 1$ se va aplica coeficientul din căsuţa a cincea ($5 = 101|_2$) din diagrama V-K, adică $\bar{A}$. Pentru uşurinţă în realizarea mapării, la intrările multiplexoarelor, s-au înscris, în paranteze, şi numărul în zecimal al coordonatei căsuţei din diagrama V-K al cărui coeficient se aplică pe acea intrare de date.

Se observă că la reţelele arborescente care conţin multiplexoare 2:1 sau 4:1, uneori, se calculează valori banale, care oricum există în sistem şi nu mai trebuie calculate, deci multiplexoarele respective se elimină. Efectuând aceste eliminări şi calculând indicatorii de eficienţă (numărul de circuite integrate, CI şi numărul de conectări) afirmaţia de la Figura 2.35-c, că impelementările cu multiplexoare de capacitate mai mare sunt mai eficiente, este contrazisă. Din implementările din Figura 2.42 rezultă că inplementările pe reţelele arborescente, având multiplexoare de capacitate mică, sunt mai eficiente. Această afirmaţie trebuie luate sub rezerva că obţinerea unei structuri simple depinde foarte mult de modul potrivit de asignare a variabilelor reziduu.

Circuitul multiplexor prezintă o limitare intrinsecă — termenii canonici produs, generaţi pe nivelul AND, sunt utilizaţi doar o singură dată (prin colectarea în interior la o poartă OR). În general, la implementarea unei funcţii costul implementării, în mare parte, este determinat de generarea termenilor produs. Nu există aceeaşi situaţie la decodificator/demultiplexor unde termenii produs, datorită faptului că sunt colectaţi în exterior în nivel OR, pot fi utilizaţi în mai multe porţi OR dacă au un fan-out corespunzător, deci impelementarea mai multor funcţii. Soluţia care apare este cea a utilizării împreună a multiplexorului cu decodificatorul.

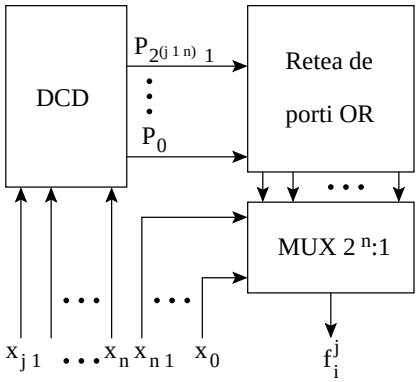


Figura 2.43 Impelementarea unei funcţii f_i^j pe o structură compusă decodificator-multiplexor.

Pentru o astfel de implementare combinată, decodificator-multiplexor, se consideră o funcţie de j variabile f_i^j. Se partajează variabilele în n variabile ale funcţiei

$(x_{n-1}, x_{n-2}, ..., x_1, x_0)$ şi $(j-1)-n$ variabile reziduu $(x_{j-1}, x_{j-2}, ..., x_{n+1}, x_n)$. Coeficienţii de variabile reziduu sunt sume de produse de aceste variabile care se pot calcula cu un DCD$(j-1-n)$:$2^{(j-1-n)}$ şi o reţea externă se porţi OR, apoi aceşti coeficienţi sunt aplicaţi pe intrările de date ale multiplexoarelor MUX2^n:1, Figura 2.43, la care selectarea se face prin variabilele $x_{n-1}, x_{n-2}, ..., x_1, x_0$.

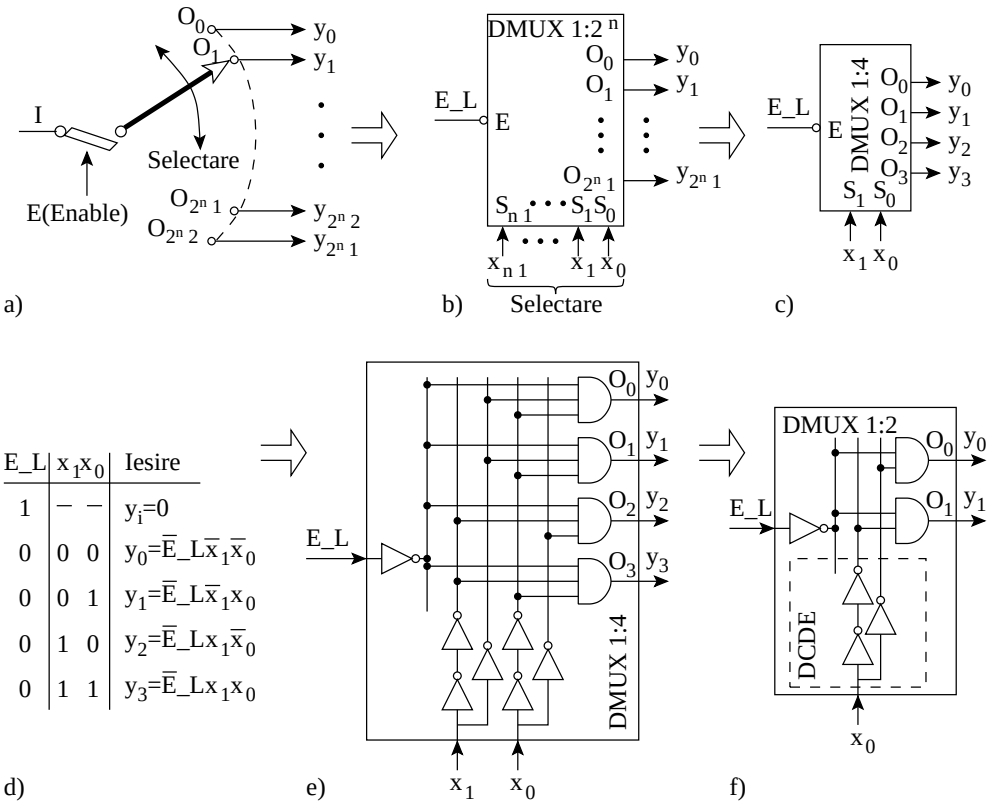
2.4.5 Demultiplexorul

Circuitul demultiplexor DMUX poate realiza o funcţie de comunicaţie prin conectarea unei linii de intrare I la oricare dintre liniile de ieşire $O_{2^n-1}, O_{2^n-2}, ..., O_1, O_0$. Funcţia aceasta de comunicaţiee — distribuirea unei intrări la oricare din ieşiri – pe care o realizează un cirucuit DMUX poate fi efectuată şi de un selector rotativ mecanic, Figura 2.44-a. La distribuitorul mecanic selectarea ieşirii, la care se conectează intrarea, se face prin poziţia contactului rotativ, iar la circuitul demultiplexor DMUX1:2^n, cu o linie de intrare şi 2^n linii de ieşire, printr-un cuvânt se selectarea cu lungimea de n biţi $x_{n-1}, x_{n-2}, ..., x_1, x_0$, Figura 2.44-b. Operaţia de demultiplexare, distribuirea unei intrări la oricare din cele 2^n ieşiri, apare ca o operaţie inversă multiplexării (selectarea unei intrări din cele 2^n şi conectarea la o singura ieşire).

Particularizând DMUX1:2^n, pentru $n = 2$, se obţine demultiplexorul cu o singură cale de intrare E (activă în L) şi patru căi de ieşire O_0, O_1, O_2, O_3, comandat pe selectare prin cuvântul x_1x_0, a cărui schemă bloc şi tabel de adevăr sunt prezentate în Figura 2.44-c şi 2.44-d. Din tabelul de adevăr rezultă expresia logică pentru fiecare ieşire: $y_0 = \bar{E}\_L\bar{y}_1\bar{y}_0$; $y_1 = \bar{E}\_L\bar{y}_1y_0$; $y_0 = \bar{E}\_Ly_1\bar{y}_0$; $y_0 = \bar{E}\_Ly_1y_0$ care corespund implementării din Figura 2.44-e. Se pot da următoarele două interpretări: la ieşirea O_i selectată de configuraţia cuvântului de selectare se transmite intrarea E_L complementată (funcţia de comunicaţie), sau dacă se consideră intrarea permanent activată $E\_L = 0$ ieşirea O_i este activă când configuraţia cuvântului de selectare este P_i; deci conform ultimei interpretări se pot genera la ieşire toţi cei patru mintermi P_0, P_1, P_2, P_3 (funcţia de circuit logic). Ultima interpretare ne arată identitatea funcţionării şi identitatea structurării, vezi secţiunea 2.4.3 şi Figura 2.28-c, între decodificator şi demultiplexor.

Se poate generaliza, un decodificator DMUX1:2^n este un distribuitor al valorii complementate E_L la ieşirea O_i, selectată prin cuvântul de selectare, iar un DCDn:2^n este un decodificator al cuvântului de selectare, când intrarea de validare este activată $E\_L = 0$, Figura 2.44-b. Deci se poate scrie identitatea DMUX1:2^n (pentru $E\_L = 0$) $\equiv$ DCDn:2^n. Aceasta explică de ce pentru decodificare sau pentru demultiplexare există un singur circuit integrat referit în cataloage ca decodificator/demultiplexor. Şi, în general, la acest circuit semnalul de validare se obţine printr-o conjuncţie între mai multe semnale de validare, unele active în stare H altele în stare L. Obţinerea validării, din conjuncţia mai multor semnale poate fi utilă într-un sisitem pentru că se poate condiţiona funcţionarea acelui sistem de realizarea (activarea) simultană a mai multor semnale (condiţii).

De exemplu, circuitul decodificator/demultiplexor 74xx138 cu tabelul de adevăr şi, cu simbolul de reprezentare date în Figura 2.45 are trei intrări de validare, una G_1 la care semnalul aplicat este activ în stare H, iar celelalte două G_{2A} şi G_{2B} la care semnalele aplicate sunt active în stare L.



E_L	x_1x_0	Iesire
1	– –	y_i=0
0	0 0	$y_0=\overline{E}\_L\bar{x}_1\bar{x}_0$
0	0 1	$y_1=\overline{E}\_L\bar{x}_1x_0$
0	1 0	$y_2=\overline{E}\_Lx_1\bar{x}_0$
0	1 1	$y_3=\overline{E}\_Lx_1x_0$

Figura 2.44 Demultiplexorul, DMUX: a) distribuitorul rotativ mecanic — analogul mecanic al demultiplexorului; b) simbol de reprezentare pentru DMUX1:2^n;c,d,e) simbol de reprezentare, tabel de adevăr şi structura circuitului pentru DMUX1:4; f) structura demultiplexorului elementar, DMUX1:2.

Funcţionarea logică este directă — o ieşire este activă dacă şi numai dacă validarea este activă şi este aplicat cuvântul de selectare corespunzător. Astfel ecuaţia logică, în notaţii de semnale interne din Figura 2.45-b, de exemplu pentru O_5, se scrie uşor:

$$O_5 = \underbrace{G_1 \cdot G_{2A} \cdot G_{2B}}_{validare} \cdot \underbrace{S_2 \cdot \bar{S}_1 \cdot S_0}_{selectare}$$

Dar ţinând cont de cerculeţele de negaţie, care indică faptul că semnalele exterioare corespunzătoare sunt active în L (între semnalul exterior şi cel interior în circuit există un inversor $G_{2A} = \bar{G}_{2A}\_L$, $G_{2B} = \bar{G}_{2B}\_L$), ecuaţia logică anterioară se poate scrie în funcţie de semnalele exterioare (aplicate sau generate)

$$\begin{aligned} Y_5\_L = \bar{O}_5 \quad &= \overline{G_1 \cdot \bar{G}_{2A}\_L \cdot \bar{G}_{2B}\_L \cdot \ x_2 \cdot \bar{x}_1 \cdot x_0} = \\ &= \bar{G}_1 + G_{2A}\_L + G_{2B}\_L + \bar{x}_2 + x_1 + \bar{x}_0 \end{aligned}$$

În acest caz, pentru o validare permanentă, intrarea G_1 se conctează la V_{CC} iar celelalte două intrări G_{2A} şi G_{2B} se conectează la masă.

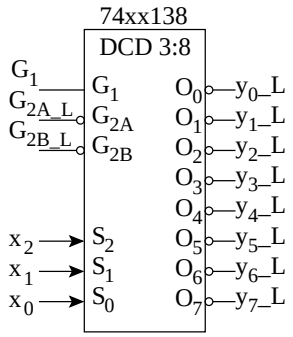
INTRARI

G_1	G_{2A_L}	G_{2B_L}	S_2	S_1	S_0
0					
	1				
		1			
1	0	0	0	0	0
1	0	0	0	0	1
1	0	0	0	1	0
1	0	0	0	1	1
1	0	0	1	0	0
1	0	0	1	0	1
1	0	0	1	1	0
1	0	0	1	1	1

IESIRI

O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0
1	1	1	1	1	1	0	1
1	1	1	1	1	0	1	1
1	1	1	1	0	1	1	1
1	1	1	0	1	1	1	1
1	1	0	1	1	1	1	1
1	0	1	1	1	1	1	1
0	1	1	1	1	1	1	1

a)



b)

Figura 2.45 Circuitul 74xx138, DMUX1:8: a) tabelul de adevăr; b) simbolul de reprezentare.

Din structura DMUX1:4, prin eliminarea intrării de selectare x_1, se obţine structura **demultiplexorului elementar DMUXE**, reprezentat în Figura 2.44-f. Dar de la DMUX1:4, prin extensie, se poate obţine şi demultiplexorul cu 2^n căi de ieşire DMUX1:2^n prin extindere la n a numărului de decodificatoare elementare şi de asemenea prin extinderea la n a numărului de porţi AND(n+1). Caracteristicile pentru o astfel de structurare a DMUX1:2^n sunt cele ale DCDn:2^n adică o dimensiune $S_{DMUX(n)} \in O(n2^n)$ şi o adâncime $D_{DMUX(n)} = 3 \in O(n)$. Adâncimea, după cum s-a analizat şi la DCD, doar teoretic este 3, pentru că în realitate o poartă AND(n+1) nu poate fi structurată pe un singur nivel logic.

Dar un DMUX1:2^n se poate defini (şi proiecta) printr-o structurare recursivă de DMUXE; considerând de data aceasta că şi ieşirile sunt active în stare L (pentru a se putea comanda intrarea de validare a următorului DMUX fără a mai adăuga inversoare).

Definiţia 2.11 Un DMUX1:2^n, selectat prin cuvântul $x_{n-1}x_{n-2}...x_1x_0$, se structurează dintr-un DMUXE selectat cu x_{n-1} şi ale cărui ieşiri comandă intrările a două DMUX1:2^{n-1} selectate prin cuvântul $x_{n-2}x_{n-3}...x_1x_0$. Apoi, asupra celor două DMUX1:2^{n-1} se aplică acelaşi procedeu de structurare, procedeul recursiv de structurare se opreşte când se ajunge la DMUX-uri selectate prin x_0. $\diamond$

Prin modul de structurare, conform aceste definiţii, Figura 2.46, se obţine o reţea arborescentă de $(2^n - 1)\times$DMUXE distribuite pe n niveluri; pe nivelul zero "frunze" selectat prin x_0 sunt $2^{n/2}\times$DMUXE iar pe nivelul n un singur DMUXE, "rădăcină", selectat de x_{n-1}. Această reţea binară are o structurare semănătoare cu cea obţinută la structurarea MUX2^n:1, după Definiţia 2.10 , diferenţa constă în sensul de transfer al datelor. La MUX2^n:1 datele de intrare se aplică la frunze iar ieşirea este prin rădăcină, pe când la reţeaua arborescentă a DMUX1:2^n se aplică o singură dată de intrare la rădăcină iar ieşirea este prin frunze. Caracteristicile acestei structurări sunt: dimensiunea $S_{DMUX(n)} \in O(2^n)$ iar adâncimea $D_{DMUX(n)} \in O(n)$.

Acest mod de structurare iterativă (la fel ca la multiplexoare) se poate realiza nu numai cu pasul un bit de selectare ci repartizând pe un nivel q biţi de selectare,

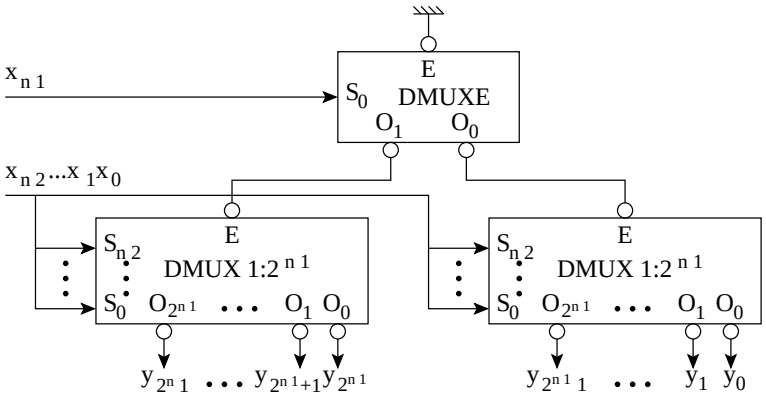


Figura 2.46 Structura recurentă a DMUX1:2^n.

aceasta înseamnă k niveluri şi pe fiecare nivel sunt DMUX1:2^q, $n = k \cdot q$. Modul acesta de structurare poate fi utilizat şi pentru obţinerea unui DMUX1:2^n pe k niveluri din circuite DMUX1:2^q obtenabile comercial.

Perechea formată din cele două circuite cu funcţii inverse, un MUX2^n:1 comandă (conectat printr-un singur fir la) un DMUX1:2^n poate fi utilizată pentru transferul serial al conţinutului unui port sursă, cu lungimea de 2^n biţi, într-un alt port destinaţie de aceeaşi lungime. Biţii portului sursă sunt aplicaţi la cele 2^n intrări de date ale MUX2^n:1, iar cele 2^n ieşiri ale DMUX1:2^n sunt aplicate la intrările portului destinaţie. Ambele cuvinte de selectare sunt generate sincron de la un numărător binar modulo 2^n, deci pe duarata a 2^n tacte de ceas aplicate numărătorului se efectuează transferul serial, bit-cu-bit din portul sursă în portul destinaţie.

2.4.6 Memoria numai cu citire, ROM

Funcţia de memorare o posedă acele circuite digitale care pot stoca (înmagazina) şi regenera, la comandă, informaţia sub formă de cuvânt.

Organizarea logică a memorării datelor, dar şi cea pe suportul fizic, este sub forma unei matrice cu **A-număr de linii (adrese)** şi **D-număr de coloane**, Figura 2.47-a. În fiecare nod al matricei poate fi stocat un bit. Se consideră că pe fiecare linie a matricei este stocată informaţia sub forma unui cuvânt cu lungimea D biţi, de unde şi termenul uzual de **linie de cuvânt**; numărul de biţi ai cuvântului este egal cu numărul de coloane. Referirea la o linie de cuvânt se face prin adresa sa, care, în general, este numărul de ordine al liniei matricei. De exemplu, în acestă figură, la adresa 0 (prima linie) este stocat cuvântul de n biţi $D_{n-1}D_{n-2}...D_1D_0 = 10...11$, la adresa 1 (linia a două) cuvântul 11...01, iar la adresa $2^n - 1$ (ultima linie) cuvântul 01...10. La activarea adresei unei linii de cuvânt se va genera la ieşire, pe coloane — **linii de bit** — cuvântul care este înscris (în locaţia de) la adresa respectivă. **Capacitatea memoriei**, exprimată în biţi, rezultă ca fiind egală cu produsul $A \times D$. În general, capacitatea memoriei se exprimă în număr de adrese înmulţit cu lungimea cuvântului (1bit, un byte, un cuvânt) stocat la o locaţie, de exemplu: 1Kbit (1Kadrese x1bit, 1K= $2^{10} = 1024$); 1Mbyte (1Madrese x1byte, 1 M= 2^{20}); 1Gcuvânt (1Gadrese

x1cuvânt, 1G= 2^{30}); lungimea D în biţi a cuvântului se specifică sau se subînţelege.

Un circuit care, prin structura sa, poate să mapeze organizarea matriceală a datelor este circuitul **ROM** (**R**ead **O**nly **M**emory); acest circuit este un suport NUMAI pentru citirea datelor înscrise, nu şi pentru modificarea (înscrierea) acestora. Referirea corectă a circuitului este circuitul ROM şi nu memoria ROM, totuşi ultima sintagmă s-a fixat; în consecinţă, în această carte se vor utiliza ambele exprimări. La circuitul ROM stocarea informaţiei este non-volatilă deoarece la pierderea tensiunii de alimentare informaţia nu este pierdută. Nevolatilitatea informaţiei se datoarează faptului că suportul unui bit, într-un nod al matricei, este prezenţa sau absenţa în acel nod a unei conexiuni (realizată printr-un: fuzibil, diodă, sau tranzistor). Această invaliditate a circuitului ROM de a nu putea fi înscris, cauzată de o structurare mai simplă, este eliminată de către un alt circuit de memorie, RAM, prezentat în secţiunea 3.6.

Pentru un ciruit ROM organizarea logică de principiu este prezentată în Figura 2.47-b. Liniile de cuvânt ale matricei sunt ieşirile de la DCDn:2^n, deci activarea unei locaţii rezultă în urma aplicării unui cuvânt de adresă $A_{n-1}A_{n-2}...A_1A_0$ la intrarea decodificatorului. Cuvântul de date de la locaţia adresată, înainte de a fi generat în exterior, de exemplu, pe o magistrală, est bufferat; în cazul conectării la o magistrală bufferele de ieşire trebuie să fie de tip TSL. Comanda bufferelor TSL este realizată prin conjuncţia a două semnale de comandă CS_L (Chip Select) şi OE_L (Output Enable). Matricea memoriei prezintă în fiecare nod o diodă şi are înscris acelaşi conţinut ca şi cel înscris în matricea de la figura Figura 2.47-a. Pentru un bit 1, înscris într-un nod, dioda respectivă este conectată între linie şi coloană, iar pentru un bit 0 înscris fuzibilul este ars în acest nod, deci dioda nu realizează conexiunea între linie şi coloană. Conform conexiunilor la diode, la activarea primei linii va fi citit cuvântul $D_{m-1}D_{m-2}...D_1D_0 = 10...11$ pentru linia a două cuvântul 11...01, iar pentru linia de adresă $2^n - 1$ cuvântul 01...10.

Aplicaţiile cu circuite ROM pot fi încadrate în două grupe: a) aplicaţii de tip aritmetic şi b) aplicaţii de tip logic.

a). Aplicaţiile de tip aritmetic se reduc la implementarea unor tabele de date, **LUT** (**L**ook-**U**p **T**able). LUT-ul este o reprezentare a unei funcţii prin valorile sale numerice, înscrise în diferite locaţii ale circuitului ROM. Adresa locaţiei se determină prin configuraţia binară a valorilor variabilelor de intrare ale funcţiei. Deci pentru o anumită configuraţie a variabilelor funcţiei — adresa unei locaţii — din locaţia respectivă se obţine, la ieşirea circuitului ROM, cuvântul binar care reprezintă valoarea funcţiei.

b). Aplicaţiile de tip logic se reduc la implementarea unei funcţii logice — în general cu ieşiri multiple — pe cele două niveluri AND şi OR ale circuitului ROM, deci acesta este un circuit combinaţional.

În structura matricei o linie de bit, prin diodele conectate la rezistenţa R, constituie un circuit max, ca cel din Figura 1.11-b, care în logică pozitivă este o poartă OR. Această poarta OR, de fapt, colectează la intrarea bufferului de ieşire acele ieşiri ale DCDn:2^n la care există diodă cu fuzibilul nears, adică însumează logic unii din termenii canonici de variabilele $A_{n-1}, A_{n-2}, ..., A_1, A_0$. De exemplu, linia de bit D_1 realizează valoarea funcţiei logice care însumează mintermii $1 \cdot P_0 + 0 \cdot P_1 + ... + 1 \cdot P_{2^n-1}$. Dacă se priveşte circuitul ROM ca un CLC cu ieşiri multiple atunci acesta este un convertor de cod (transcodor), pentru o configuraţie a cuvântului de

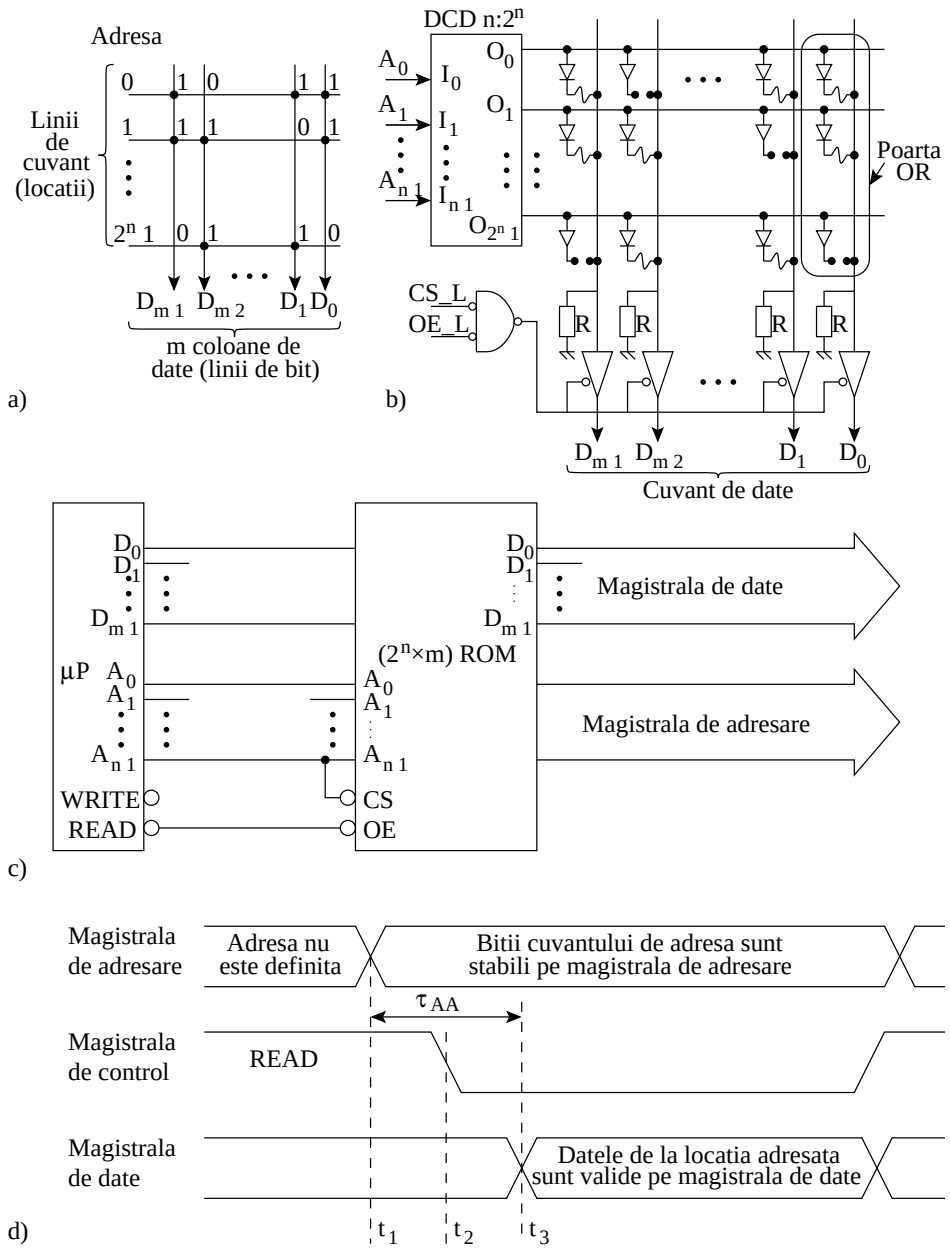


Figura 2.47 Memoria numai cu citire, ROM: a) structurare matricială, de principiu, pentru informaţia sub formă de cuvinte binare; b) structurarea matriceală a unui circuit ROM; c) conectarea la magistralele μP a unui circuit ROM; d) diagramele de timp ale semnalelor de control pentru efectuarea operaţiei de citire.

de intrare, $A_{n-1}, A_{n-2}, ..., A_1, A_0$, se generează o configuraţie a cuvântului de ieşire, $D_{m-1}, D_{m-2}, ..., D_1, D_0$, care, de fapt, realizează o dependenţă intrare ieşire ca circuitul cu schema de principiu din Figura 2.33.

La fel ca şi multiplexorul circuitul ROM este un circuit logic universal pentru că prezintă atât nivelul de AND cât şi nivelul de OR dar, spre deosebire de multiplexor, la ROM nivelul de OR este programabil şi poate fi un circuit cu ieşiri multiple (generează un cuvânt de date). ROM-ul având posibilitatea de a fi programat, pe nivelul de OR (codificator), poate fi mai eficient folosit, adică toţi cei 2^n maxtermi generaţi pe nivelul de AND (decodificator) pot fi utilizaţi pentru implementarea mai multor funcţii şi nu numai pentru una singură ca la multiplexor. Funcţia de implementat nu trebuie minimizată deoarece nivelul decodificator produce toţi cei 2^n termeni canonici. Implementarea unei funcţii se face prin maparea directă a tabelului de adevăr pe matricea OR programabilă a ROM-ului; pe linia de bit (coloana) alocată funcţiei respective pentru termenii canonici care au valoare 1 fuzibilul diodei rămâne intact, iar pentru termenii care au valoare 0 conexiunea se exclude prin arderea fuzibilului.

Pentru aplicaţiile în care memoria numai cu citire este utilizată ca un LUT aceasta trebuie inclusă într-un sistem pe baza de μP, Figura 2.47-c. Memoria ROM se conectează la cele trei magistrale ale sistemului în modul următor: intrările de adresă la magistrala de adresare, ieşirile de date la magistrala de date, iar la magistrala de control, care în acest caz este reprezentată doar de semnalul READ generat de μP, se conectează intrarea de validare a ieşirii, OE. La generarea unui cuvânt de adresă de către μP, din locaţia de adresă respectivă, se obţine cuvântul care se depune pe magistrala de date de unde este citit de către μP. Pentru conexiunile din acest sistem circuitul de memorie pot fi citite numai cuvintele de adresă care au bitul $A_{n-1} = 0$, deoarece acest bit este utilizat pentru activarea semnalului de selectare a circuitului, CS, activ în L. În procesul de citire a memoriei trebuie îndeplinită o anumită secvenţialitate în aplicarea cuvântului de adresă şi a semnalelor de control CS_L şi OE_L.

În Figura 2.47-d sunt prezentate diagramele de semnale pentru efectuarea operaţiei de citire. La momentul t_1 cuvântul adresă generat de microprocesor se aplică pe magistrala de adresare la intrarea memoriei şi începe decodificarea adresei. În momentul t_2, prin semnalul READ, generat tot de μP, se comandă bufferele de ieşire pentru trecerea lor din HZ în starea lor normală. Abia începând din momentul t_3 datele din locaţia de memorie sunt validate pe magistrala de date şi pot fi citite de către μP. Se defineşte timpul de acces, τ_{AA}, ca intervalul de timp din momentul momentul aplicării cuvântului de adresă la intrarea memoriei (t_1) până când datele la ieşire sunt valide (t_3), deci când datele pot fi citite. Acest parametru important al circuitelor ROM are valori cuprinse între $20 \div 90ns$ pentru cele în tehnologie bipolară şi între $70 \div 400ns$ pentru cele CMOS. Pentru o proiectare se recomandă consultarea foii tehnice a circuitului respectiv deoarece în această prezentare simplificată au fost omişi alţi parametrii de timp (vezi Figura 3.92-a).

La implementarea funcţiilor logice pe memorii numai cu citire este normală tendinţa de a se utiliza circuite ROM cât mai simple şi mai ieftine, deci de capacitate cât mai mică. În acest sens se recomandă ca reducerile la funcţiile logice de implementat să se facă în primul rând pe numărul variabilelor de intrare. Deoarece capacitatea memoriei în biţi este $2^n \times m$, micşorarea intrărilor cu o unitate reduce capacitatea de două ori ($2^n \cdot m / 2^{n-1} \cdot m = 2$) pe când micsorarea ieşirilor cu o unitate duce la o

capacitate numai de $m/(m-1)$ ori mai mică, $(2^n \cdot m)/2^n \cdot (m-1) = m/(m-1)$). Posibilitatea de micşorare a numărului de variabile aplicate la o memorie numai cu citire apare la acele funcţii care sunt parţial definite la intrare (unele configuraţii binare nu au sens pentru funcţie) sau pentru mai multe configuraţii binare de intrare funcţia are aceeaşi ieşire. Configuraţiile care generează aceeaşi ieşire sunt grupate într-o clasă de echivalenţă care necesită pentru exprimare doar o singură configuraţie binară (cod). Dacă la o funcţie cu n variabile un număr de n_1 variabile ($n_1 < n$) pot fi grupate într-un număr de clase de echivalenţe care pot fi exprimate cu p biţi ($p < n_1$) atunci numărul de intrări ale memoriei pentru implementare scade de la n la $n - n_1 + p$.

Exemplul 2.18 Pe un circuit ROM să se implementeze următoarea funcţie:

$$f(x_5, x_4, x_3, x_2, x_1, x_0) = \sum_{0}^{63}(4, 5, 20, 29, 41, 42, 45, 57, 53, 58, 61, 63)$$

Soluţie. Implementarea se poate face direct prin înscrierea valorii 1 în locaţiile ale căror

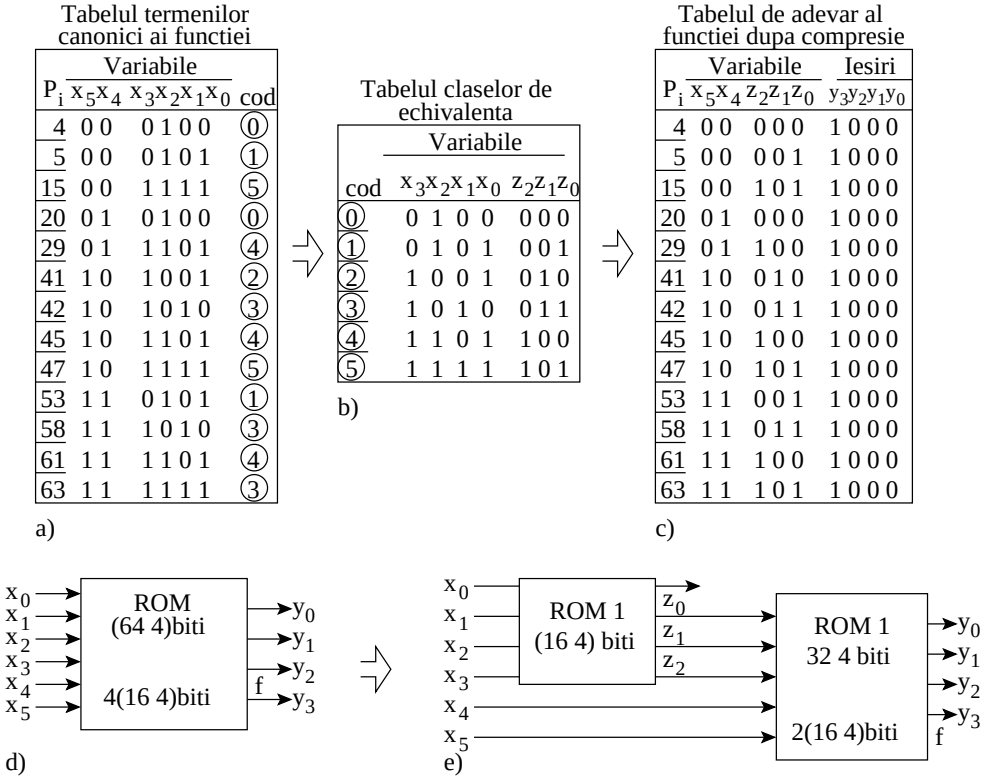


Figura 2.48 Explicativă pentru sucesiunea etapelor în compresia variabilelor unei funcţii în scopul reducerii capacităţii memoriei ROM folosită pentru implementare.

adrese sunt specificate în lista mintermilor, pe un modul memorie compus din patru circuite

de capacitate 16×4 biţi; $D_3 = f$, ceilalţi trei biţi de ieşire D_2, D_1, D_0 rămân neutilizaţi, Figura 2.48-d. Dar pentru cei 13 termeni produs ai funcţiei care au valoare logică 1, printr-o împărţire a variabilelor funcţiei în două grupe x_5x_4 şi $x_3x_2x_1x_0$, se poate realiza o compresie prin identificarea unor clase de echivalenţe. Între cele 13 cuvinte binare care sunt cuvinte de adresă ce se aplcă la intrarea memoriei ROM, pentru grupul de variabile $x_3x_2x_1x_0$ există numai şase subcuvinte binare distincte, deci s-a identificat şase clase de echivalenţă notate cu (0),(1),(2),(3),(4),(5) ca în Figura 2.48-a. Şase clase de echivalenţă pot fi codificate numai cu 3 biţi $z_2z_1z_0$ ca în tabelul din Figura 2.48-b. Transcodarea de la patru variabile x_3, x_2, x_1, x_0 la trei variabile z_2, z_1, z_0 se face cu un circuit ROM16$\times$ 4 biţi (un bit de ieşire este neutilizat). În final, funcţia se poate exprima doar cu 5 variabile x_5, x_4 şi z_2, z_1, z_0 ca în tabelul din Figura 2.48-c, cu o implementare doar pe două circuite ROM16×4 plus încă unul pentru transcodorul claselor de echivalenţă, Figura 2.48-e. Faţă de modalitatea directă de implementare cu circuite patru ROM de 16×4 biţi, Figura 2.48-d, s-a econimisit un circuit de 16×4 biţi.

Din analiza comparativă a tabelelor de adevăr ale funcţiilor booleene pentru aplicaţii logice şi a celor pentru aplicaţii aritmetice (numerice) se constată — fără a găsi o explicaţie riguroasă — că: raportul între numărul configuraţiilor variabilelor de intrare ale funcţiei care produc valoarea 1 pentru ieşire, sau numărul configuraţiilor care produc valoarea 0 pentru ieşire, supra numărul total de configuraţii ale variabilelor de intrare ale funcţiei, 2^n, este egal cu 0.5 pentru funcţiile aritmetice şi mult diferit de 0.5 pentru funcţiile logice. Se poate verifica această observaţie prin compararea tabelelor de adevăr ale operatorilor logici AND,OR,NAND,NOR de două variabile şi tabelul de adevăr al operatorului aritmetic XOR (sumă modulo doi) de două variabile (se poate extinde şi pentru $n > 2$).

Ca o consecinţă a acestei observaţii se deduce că pentru implementarea unei funcţii logice, pe bază de 1, sub formă FCD (sumă de mintermi) este necesar a se genera puţini mintermi din numărul total de 2^n ai funcţiei. De asemenea, dacă funcţia are multe valori 1 şi puţine valori 0, se poate face sinteza funcţiei negate $\bar{f}$ prin sumarea mintermilor pentru (puţinele) configuraţii la care funcţia are valoarea 0 şi apoi prin negarea lui $\bar{f}$ se obţine funcţia, deci tot un număr mic de mintermi ce trebuie generaţi în raport cu numărul total de mintermi 2^n. Dar la o implementare pe un ROM sunt generaţi toţi cei 2^n mintermi indiferent dacă sunt utilizaţi în sinteza funcţiei sau nu; pentru un număr mare de variabile de intrare costul generării mintermilor neutilizaţi devine destul de ridicat. În concluzie, pentru implementarea aplicaţiilor aritmetice (LUT-uri, programe de calculator) circuitul ROM este o soluţie recomandată dar pentru implementarea aplicaţiilor logice, mai ales pentru n ridicat, ROM-ul nu este recomandat. Vom vedea că pentru implementarea funcţiilor logice, cu n ridicat şi număr redus de mintermi necesari, este recomandat circuitul PLA.

2.4.6.1 Realizarea circuitelor şi modulelor ROM

Adresarea bidimensională. Pentru o memorie ROM de capacitate $2^n \times m$ biţi, cu n biţi de adrese şi m biţi de date pe ieşire, Figura 2.47-a, se poate calcula dimensiunea $S_{ROM(n,m)}$. Considerând fiecare linie de bit ca o poartă OR cu 2^n intrări

şi pentru fiecare buffer de ieşire TSL trei terminale rezultă:

$$\begin{aligned} S_{ROM(n,m)} &= S_{DCD(n)} + m \cdot S_{OR(2^n)} + m \cdot S_{Buffer} \\ &= n \cdot 2^n + m \cdot 2^n + m \cdot 3 \qquad \in O((n+m) \cdot 2^n) \end{aligned}$$

O astfel de dimensiune pentru n de valoare ridicată generează dificultăţi la implementarea ROM. În general, $n > m$, iar $m = 2^k$ are valori uzuale de 1,4, sau 8. Dificultăţile de implementare sunt datorită numărului mare de ieşiri de la DCDn:2^n şi valorii relativ mică a lui m. De exemplu, pentru circuitul 27C040 ROM cu capacitatea 4M ($512K \times 8 = 10^{19} \times 2^3 = 2^{22}$ biţi) ar rezulta un decodificator cu (512×1024) ieşiri, ceea ce ar fi foarte greu de realizat, iar dimensiunile matricei ar fi $2^{19} \times 8$, ori această suprafaţă "filiformă" ar fi cu totul neconvenabilă pe aria de Si. Se recomandă ca suprafeţele ocupate pe aria de Si să fie pătratice (sau dreptunghiulare) din motive tehnologice şi de împachetare (conexiuni şi pini). Aceste două dificultăţi, numărul mare de ieşiri ale DCD $n : 2^n$ cât şi suprafaţa filiformă a matricei, pot fi depăşite prin modul de adresarea bidimensională.

La adresarea bidimensională, exprimată prin Definiţia 2.9 , cuvântul de adresă $A_{n-1}\ A_{n-2}...A_1A_0$ se împarte în două subcuvinte de adresă cu lungimea n_2 biţi ($A_{n-1}A_{n-2}\ ...A_{n_1}\ A_{n_1}$) şi celălalt cu lungimea de n_1 biţi ($A_{n_1-1}A_{n_1-2}...A_1A_0$), în general n_1 şi n_2 au valori apropiate sau chiar egale, $n = n_1 + n_2$. În felul acesta se realizează două decodificatoare, unul DCDn_2:2^{n_2} pentru decodificarea liniilor matricei şi celălalt DCDn_1:2^{n_1} pentru decodificarea coloanelor, fiecare având un număr de ieşiri mult mai mic decât decodificatorul iniţial DCDn:2^n; $2^{n_1} << 2^n$, $2^{n_2} << 2^n$, $2^{n_1} \times 2^{n_2} = 2^n$. Matricea iniţială care avea 2^n linii cu m biţi pe fiecare linie se scalează la o matrice care are 2^{n_2} linii dar cu $m \cdot 2^{n_1}$ biţi pe fiecare linie, deci raportul dimensiunilor se modifică de la $2^n/m >> 1$ la $2^{n_2}/m \cdot 2^{n_1} \approx 1$. Prin trecerea de la o matrice "filiformă" la una pătratică, capacitatea memoriei, 2^{n+k} biţi, nu se modifică, se pot calcula valorile pentru n_1 şi n_2 cu următoarele relaţii: $n_2 = (n+k)/2$, $n_1 + k = (n+k)/2$; $m = 2^k$. Recalculând dimensiunea $S_{ROM(n,m)}$ pentru adresarea bidimensională rezultă:

$$\begin{aligned} S_{ROM(n,m)} &= S_{DCD(n_2)} + m \cdot 2^{n_1} \cdot S_{OR(2^{n_2})} + S_{DCD(n_1)} \\ &\quad + m \cdot S_{Buffer} \\ &= n_2 \cdot 2^{n_2} + m \cdot 2^{n_1} \cdot 2^{n_2} + n_1 \cdot 2^{n_1} + m \cdot 3 \\ &\quad \in O(m \cdot 2^n) \end{aligned} \tag{2.16}$$

Valoarea dimensiunii proporţională cu capacitatea memoriei ($m \cdot 2^n$) nu mai poate fi micşorată!

În această variantă de adresare bidimensională decodificatorul de linii, prin subcuvântul de adresă de n_2 biţi, va activa câte o linie pe care există m cuvinte de câte 2^{n_2} biţi, iar decodificatorul de pe coloane va trebui să selecteze şi să aplice la ieşire câte un bit din fiecare din cele m cuvinte ale liniei activate. Această selectare se realizează cu un grup de $m \times$MUX2^{n_1}:1, subcuvântul de adresă $A_{n_1-1}A_{n_1-2}...A_1A_0$ se aplică pe intrările de selectare ale tuturor celor m multiplexoare. Practic, pentru un subcuvânt de adresă aplicat la grupul de multiplexoare se extrag biţii din aceeaşi poziţie ale celor m cuvinte. Logic, memoria rămâne o structurare de matrice cu 2^n linii şi m coloane.

Pentru circuitul 27C040 folosit anterior, printr-o decodificare bidimensională la care dacă se alege $n_1 = 11$ şi $n_2 = 8$ se obţine o structurare cu un decodificator pentru linii cu $2^{11} = 2048$ ieşiri, pe fiecare linie sunt 2048 biţi, iar pentru selectarea

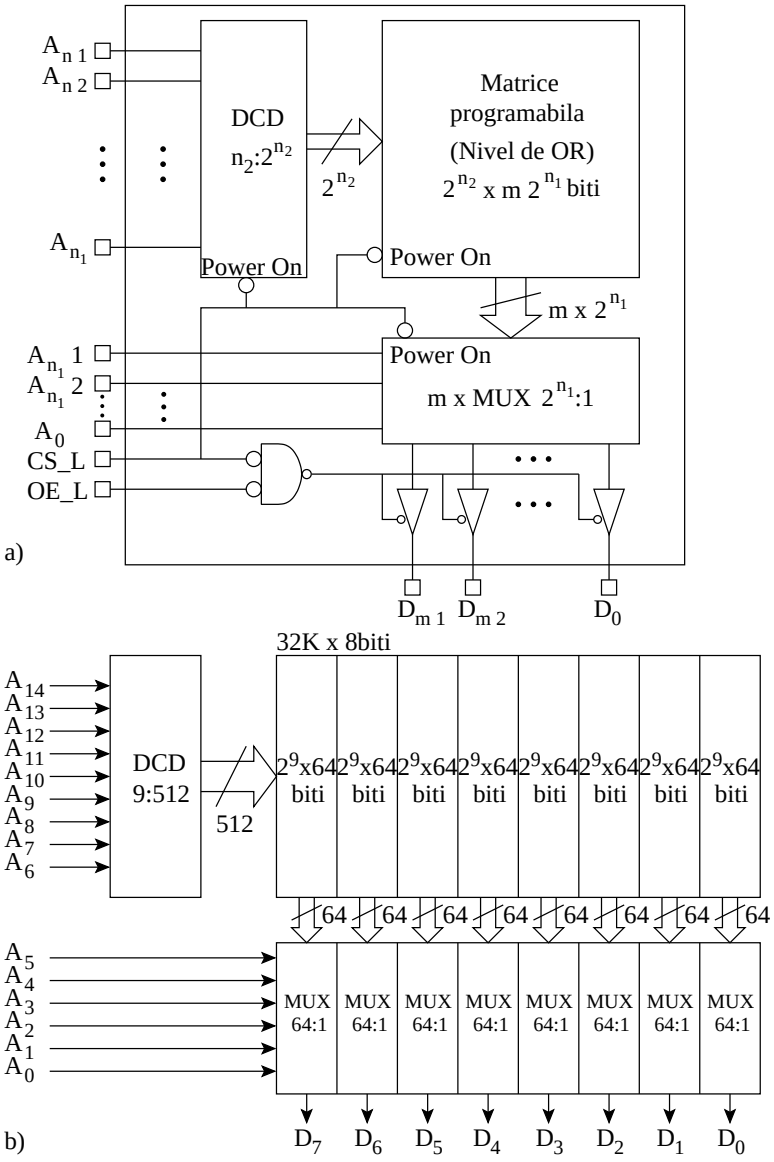


Figura 2.49 Decodificarea bidimensionaă a circuitului ROM: a) organizarea generală, de principiu, cu un DCDn_2:2^{n_2} pentru activarea liniilor şi cu un grup de $m\times$MUX2^{n_1}:1 pentru selectarea coloanelor; b) particularizarea structurării anterioare pentru o memorie ROM cu organizarea logică de $32K \times 8$ biţi.

cuvântului de ieşire de 8 biţi sunt necesare 8×MUX2^8:1; dimensiunea matricei este pătratică $2^{11}/8 \times 2^8 = 1$.

În organizarea de principiu, pentru decodificarea bidimensională a unui circuit ROM, pe lângă validarea bufferelor de ieşire TSL, prin conjuncţia dintre semnalele Chip Select şi Output Enable, există şi o comandă pentru cele trei componente: decodificatoare, matrice şi multiplexoarele pe ieşire, pentru trecerea în regimul de aşteptare (standby). Regimul de aşteptare se obţine prin neactivarea semnalului CS_L, aplicat şi ca semnal de conectare la sursa de alimentare Power On, care va deconecta alimentarea componentelor respective. Deci când circuitul *ROM* nu este selectat, automat, consumul său de putere se reduce, prin trecerea în regim de aşteptare, până la 10% din cel în regimul normal.

În Figura 2.49-b este prezentată, ca o exemplificare, o posibilă structurare de decodificare bidimensională a unui circuit ROM cu organizarea logică de $32K \cdot 8$ biţi. Decodificatorul cu 15 intrări şi $(32 \cdot 1024)$ ieşiri este substituit cu un DCD9:2^9 la care se aplică subcuvântul $A_{14} \div A_6$ şi 8×MUX64:1 (toate selectate de acelaşi subcuvânt $A_5 \div A_0$). În acest fel, matricea programabilă (nivelul OR) de la raportul dimensiunilor $2^{15}/2^3 = 2^{12}$ este scalată la raportul $2^9/(8 \cdot 2^6 = 1$ deci a devenit pătratică; pe fiecare linie, iniţial conţinând un cuvânt de date de 8 biţi, au fost plasate după scalare 8 cuvinte de date fiecare de câte 64 biţi.

Tipuri de circuite ROM. Personalizarea unui circuit ROM pentru o anumită aplicaţie este realizată prin informaţia care se înscrie pe nivelul *SAU* — matricea programabilă. Această informaţie, valorile biţilor în fiecare nod al matricei, este elaborată de către utilizator dar înscrierea/programarea în noduri se face în diferite modalităţi, fie de către producătorul circuitului, fie de către utilizator. În funcţie de modalitatea fizică de programare există mai multe tipuri de circuite ROM.

1. ROM programat prin mascare. Informaţia pentru aplicaţie, elaborată de proiectant, este trimisă la turnătoria de siliciu. Fabricantul, pe baza acestei informaţii, generează una sau două măşti şi termină fazele de fabricaţie ale circuitului ROM, adică se înscrie 1 sau 0 în nodurile matricei. Fizic, această înscriere prin mascare se reduce la prezenţa sau absenţa în fiecare nod a unei conexiuni, între linia de cuvânt şi lina/coloana de bit. Evident, odată faza de programare încheiată, prezenţa unei erori în programarea nodurilor duce la rebutarea circuitului. Datorită costului ridicat de fabricaţie şi imposibilitatea corectării unei erori, realizarea aplicaţiilor cu ROM programat prin mascare este indicată pentru generarea de funcţii standard, cum ar fi tabele de conversie foarte uzuale (LUT), funcţii specifice generate de către utilizator şi când este nevoie de o producţie de serie mare (produse auto, de larg consum etc.).

2. ROM programabil, PROM (**P**rogrammable **R**ead **O**nly **M**emory). Acest tip de circuit ROM a apărut ca un răspuns la nevoia utlilizatorului de a nu mai fi legat de turnătoria de siliciu şi de timpul lung necesar realizării aplicaţiei. Fabricantul produce circuitul care este deja înscris în toate nodurile sale fie cu bitul 1, fie cu bitul 0, depinde cum este organizat ROM-ul. Fizic, aceasta înseamnă că în fiecare nod există un fuzibil între linia de cuvânt şi lina de bit sau un tranzistor conectat cu drenul la linia de bit iar poarta este comandată de linia de cuvânt, ca în Figura 2.50-a şi a cărei sursă/emitor este legat la masă printr-un fuzibil. Într-un nod realizat cu tranzistor, al cărui fuzibil înseriat în sursă nu este ars, atunci când linia de cuvânt care comandă poarta este în stare activă (se generează mintermul respectiv) va forţa la potenţialul masei tensiunea pe linia de bit, respectiv linia de bit va fi un 1 logic atunci

când mintermul respectiv are valoarea logică 0. Rezultă că o linie de bit realizează în logică pozitivă funcţia NOR de toţi mintermii ale căror tranzistoare conectate la acea linie nu au fuzibilul ars, iar după bufferul inversor de ieşire se obţine funcţia OR.

Programarea PROM-ului este efectuată de către utilizator cu ajutorul unui programator, prin care fuzibilul este ars numai în acele noduri în care informaţia iniţială înscrisă prin fabricaţie trebuie schimbată în bitul complementar. Practic, se selectează nodul prin linia de cuvânt şi lina de bit corespunzătoare şi apoi, pe linia de bit se aplică un impuls de tensiune de valoare ridicată (10-30V) care forţează un curent prin fuzibil ce duce la arderea acestuia. Inconvenientele care pot să apară la aceste dispozitive **o singură dată programabilă, OTP** (**O**ne **T**ime **P**rogrammable) au fost prezentate la sfârşitul secţiunii 1.2. Deşi PROM-ul elimină dependenţa de turnătoria de siliciu, totuşi dezavantajul rebutării circuitului în cazul unei erori de programare nu este eliminat, acest dezavantaj este eliminat de următoarele tipuri de ROM reprogramabile.

3. ROM reprogramabil, EPROM (**E**raseble **P**rogrammable **ROM**). Facilitatea de ştergere a conţinutului unui circuit ROM şi apoi reprogramarea se bazează pe funcţionarea tranzistorului cu poartă flotantă a cărei structură este prezentată în Figura 2.50-b.

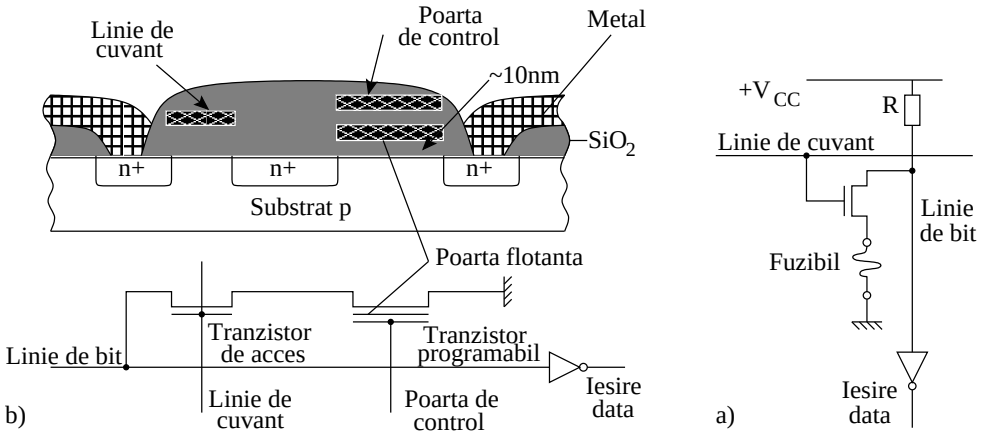


Figura 2.50 Structurarea de noduri pentru circuite ROM: a) nod de ROM programabil; b) nod de EEPROM.

Tranzistorul cu poartă flotantă, faţă de un nMOS normal, are două porţi suprapuse, poarta flotantă şi poarta de control, separate printr-un strat de SiO_2. Poarta flotantă, spre deosebire de cea de control, nu are în exterior un terminal de acces şi este izolată în masa de SiO_2 care o înconjoară, iar stratul de SiO_2 între această poartă şi substrat este foarte subţire, cel mult $10nm$. În anumite condiţii, de tensiuni aplicate pe terminalele tranzistorului, electronii cu energie ridicată din canal ("hot electron") pot străbate stratul foarte subţiere de SiO_2 pâna la poarta flotantă, iar aceasta, fiind izolată, rămâne încărcată permanent cu sarcina negativă captată (permanent înseamnă cel puţin 10 ani chiar şi la temperatura de 125 grade Celsius). Această sarcină negativă permanentă de pe poarta flotantă creşte tensiunea de prag V_{pn} a tranzistorului nMOS la aproximativ 7V, ceea ce practic înseamnă că acesta

este blocat pentru toate tensiunile normale ale circuitului $(5 \div 6)$V. Procesul poate fi reversiblil, prin aplicarea unui fascicol de radiaţie ultravioletă care anulează sarcina negativă acumulată pe poarta flotantă, deci tranzistorul devine iarăşi comandabil cu tensiunile normale ale circuitului.

Structurarea unui nod pentru un circuit EPROM este similară cu cea a unui nod dintr-un circuit PROM descrisă anterior — un tranzistor care conectează linia de bit la masă printr-un fuzibil. Numai că, la EPROM, tranzistorul de acces care leagă la masă linia de bit are în sursă înseriat un tranzistor cu poartă flotantă în loc de fuzibil. Programarea nodului, adică blocarea tranzistorului cu poarta flotantă prin colectarea unei sarcini negative pe poarta flotantă, se face în felul următor: se activează linia de cuvânt (poarta tranzistorului de acces); se aplică prin linia de bit (şi tranzistorul de acces) pe drenul tranzistorului cu poarta flotantă o tensiune în jur de 12 volţi; se aplică un impuls de tensiune de $13 \div 14$ volţi pe poarta de control care ajută la colectarea sarcinii negative (electroni) pe poarta flotantă. Rezultă că tranzistorul cu poarta flotantă, pentru care tensiunea de prag V_{pn} a devenit ridicată, întrerupe conectarea la masă în acel nod al liniei de bit chiar dacă tranzistorul de access corespunzător este comandat. Ştergerea nodului se face prin expunerea circuitului, timp de 20-30 de minute, în radiaţii ultraviolete. Circuitele EPROM au pe partea superioară o fereastră, transparentă la radiaţii ultraviolete, realizată din cuartz. Dezavantajul acestui mod de ştergere apare prin faptul că circuitul EPROM trebuie scos din soclul său de pe placa de circuit inprimat iar ştergerea sa este totală, adică se şterg toate nodurile, nu numai nodul care ar urma să fie reprogramat.

Există următoarele circuite EPROM tip Intel obtenabile comercial:

2716-16K(2Kx8biţi);	2732-32K(4Kx8biţi);
2764-64K(8Kx8biţi);	27128-128K(16Kx8biţi);
27256-256K(32Kx8biţi);	27512-512K(64Kx8biţi);
27C010-1M(128Kx8biţi);	27C020-2M(256Kx8biţi);
27C210-1M(64Kx16biţi);	7C220-2M(128Kx16biţi);
27C040-4M(512Kx8biţi);	27C240-4M(256Kx16biţi);

4. Memoria ROM cu ştergere pe cale electircă EEPROM, E^{2}PROM (**E**lectricaly **E**rasable **P**rogrammmable **ROM**). Acest tip de ROM elimină dezavantajele de la EPROM şi anume, circuitul nu mai trebuie scos din soclu pentru ştergere pentru că acesta operaţie se face pe cale electrică. Practic, ştergerea se face ca şi înscrierea prin aplicarea unei tensiuni pe poarta de control, dar de data aceasta, o tensiune de polaritate inversată, care elimină spre substrat sarcina negativă acumulată pe poarta flotantă. Ştergerea se face nu pe bit ci pe blocuri care pot ajunge până la 64Kbytes (de exemplu, la o memorie de 1Mbyte sau mai mare).

O variantă de EEPROM este **memoria flash**. Frecvent, memoriile Flash se produc sub formă de cartele astfel încât să fie utilizate în aparatura portabilă cum ar fi: camerele digitale, telefoane mobile, îmbrăcămintea electronică, transferul informaţiei între două calculatoare (simularea unei diskete). În viitor, memoria Flash poate substitui harddisk-ul oferind un timp de acces în jur de 100ns, faţă de $6 \div 10$ ms la harddisku-rile actuale. Obstacolul care există, actual, la memoriile flash constă în numărul limitat de ştergeri/înscrieri, nu cu mult peste 10.000 ori şi capacitatea de stocare care a ajuns doar la sute de Mbytes (faţă de 120 de Gbytes la harddisk-uri uzuale acum).

Tabelul 2.1 Tipuri de circuite ROM

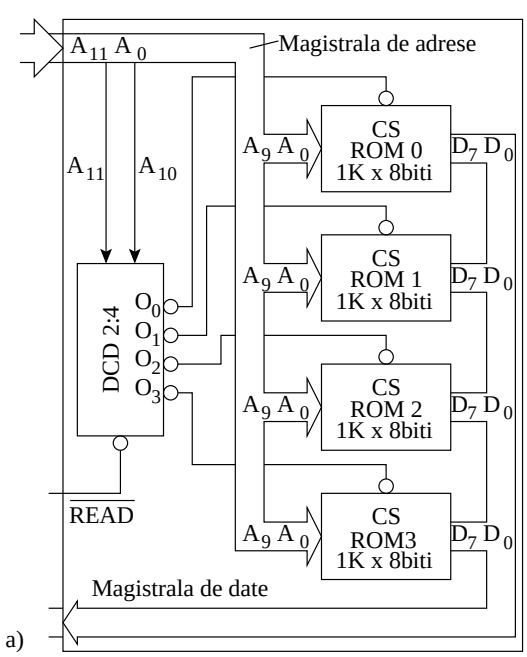
Tipul	Tehnologia	Timp de acces τ_{AA}	Timp de înscriere	Comentarii
ROM cu mascare	NMOS, CMOS	10-200ns	$> 2-3$ săpt.	OTP, consum redus de putere
ROM cu mascare	Bipolar	<100ns	$> 2-3$ săpt.	OTP, consum ridicat, densitate scazută
PROM	Bipolar	<100ns	$10-50\mu$s/byte	OTP, consum ridicat
EPROM	NMOS, CMOS	25-200ns	$10-50\mu$s/byte	Reutilizabilă, consum scăzut
EEPROM	NMOS	50-200ns	$10-50\mu$s/byte	10.000-100.000 limită înscrieri/citiri

În Tabelul 2.1 [Wakerly ′2000] sunt sintetizate caracteristicile circuitelor ROM obtenabile comercial.

2.4.6.2 Module de memorie ROM

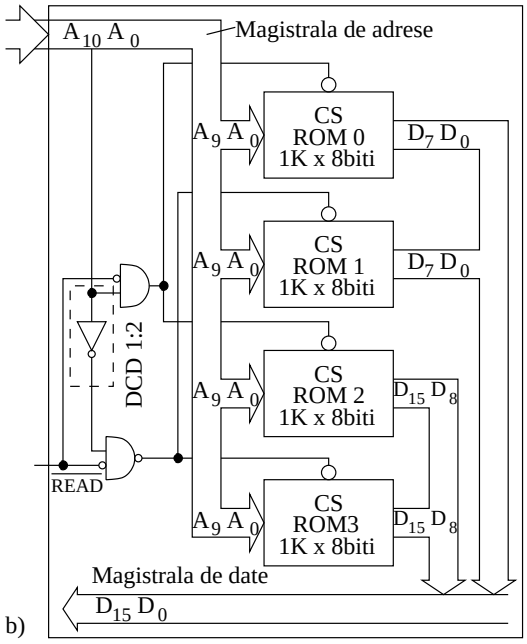
Necesităţile în aplicaţii cu memorie ROM pot depăşi capacităţile pe care le prezintă circuitele discrete obtenabile comercial. Pentru astfel de aplicaţii se realizează module de memorie ROM care extind capacitatea circuitelor ROM discrete. Pentru a putea fi integrate în module, circuitele ROM sunt prevăzute cu un semnal de control de selectare circuit, CS (în general, activ în stare L). Faţă de circuitele ROM discrete, un modul ROM realizează o extensie fie a capacităţilor de adresare, fie a lungimii cuvântului memorat sau fie a amândorura simultan.

1. Extinderea capacităţii de adresare. Extinderea apare ca o rezolvare a realizării unei memorii cu capacitate de adresare C din circuitul ROM cu număr de c adrese, ceea ce impune utilizarea a $C/c = p$ circuite componente cu c adrese. Se va exemplifica realizarea unui modul cu capacitatea de adresare 4K adrese din circuite ROM de capacitate $1K \times 8$ biţi, rezultă că sunt necesare 4K/1K = 4 circuite. Deoarece adresarea unui ROM de 1K adrese se face cu un cuvânt de adresă de 10 biţi, $A_9, A_8, ..., A_1, A_0$, iar pentru adresarea moduluilui de 4k sunt necesari 12 biţi de adresă $A_{11}, A_{10}, ..., A_1, A_0$, aceasta ar impune extinderea decodficatoarelor (interne) ale circuitelor cu încă 2 biţi A_{11} şi A_{10}. Dar, extinderea este posibilă doar în exterior cu un DCD2:4 pe ale cărui intrări sunt aplicaţi biţii de adresă A_{11} şi A_{10}, prin aceasta fiecare circuit ROM va fi selectat pe intrarea CS de către o ieşire a decodificatorului exterior, Figura 2.51-a. În spaţiul de adresare de 4K, ce poate fi acoperit cu un cuvânt de adrese cu lungimea de 12 biţi (de la 000H÷FFFH cu exprimare în hexazecimal, H), segmentul de adrese repartizat fiecărui circuit ROM se obţine prin construirea mapei adreselor memoriei. Biţii de adresă $A_9...A_0$ se aplică la toate cele patru circuite de 1K adrese, iar biţii A_{10} şi A_{11} sunt utilizaţi, prin decodificare, la selectarea a câte unui circuit de 1K adrese. Segmentele de adresă ale circuitelor, prin alăturare, formează un spaţiu de adresare continuu de 4K adrese.



Mapa adreselor memoriei

	$A_{11}A_{10}$	A_9 ... A_1A_0
ROM 3 (FFF CFF)H	1 1	1 ... 1 1 1 ... 1 0 0 ... 0 0
ROM 2 (BFF 800)H	1 0	1 ... 1 1 1 ... 1 0 0 ... 0 0
ROM 1 (7FF 400)H	0 1	1 ... 1 1 1 ... 1 0 0 ... 0 0
ROM 0 (3EF 000)H	0 0	1 ... 1 1 1 ... 1 0 0 ... 0 0



Mapa adreselor memoriei

	A_{10}	A_9A_8 ... A_1A_0
ROM 1 si ROM 3 (7FF 400)H	1	1 1 ... 1 1 1 1 ... 1 0 0 0 ... 0 1 0 0 ... 0 0
ROM 0 si ROM 2 (3EF 000)H	0	1 1 ... 1 1 1 1 ... 1 0 0 0 ... 0 1 0 0 ... 0 0

Figura 2.51 Organizarea modulelor de memorie ROM prin: a) extinderea capacităţii de adresare; b) prin extinderea simultană atât a capacităţilor de adresare cât şi a lungimii cuvântului de date.

2. Extinderea atât a capacităţii de adresare cât şi a lungimii cuântului. Folosind acelaşi circuit ROM, cu capacitatea 1Kx8biţi pentru realizarea unui modul de capacitate 2Kx16biţi, rezultă că sunt necesare $2K \times 16$biţi$/1K \times 8$biţi $= 2 \times 2$ circuite; extinderea este atât pe adrese $2 \times 1K = 2K$ adrese cât şi ca lungime de cuvânt de date 2×8biţi $= 16$ biţi Figura 2.51-b.

Extinderea de adrese este similară ca la modulul anterior cu deosebirea că de data aceasta în exterior se adaugă un DCD1:2, bitul de adresă A_{10} pentru selectarea celor două segmente de adrese. Fiecare ieşire a decodificatorului selectează simultan câte două circuite de capacitate 1Kx8biţi, ROM0 cu ROM2 şi ROM1 cu ROM3. La circuitele dintr-o pereche se aplică acelaşi cuvânt de adresă $A_9...A_0$ dar din unul se citeşte byte-ul superior iar din celălalt byte-ul inferior din care se compune cuvântul de date pe magistrala de date. În aceste exemple de extindere se consideră că ieşirile de date ale circuitelor ROM sunt de tip TSL.

Uneori, când sunt disponibili suficienţi biţi în cuvântul de adresare, se elimină DCD-ul exterior pentru extinderea capacităţii de adresare, iar pentru selectarea a câte unui circuit ROM este asignat (repartizat) un bit disponibil din cuvântul de adresa Avantajul acestui mod de adresare — adresare liniară — în raport cu adresarea cu codificarea completă este simplitatea; dar în acest caz segmentele de adrese acoperite de către fiecare circuit ROM component nu se mai alătură într-o zonă continuă din spaţiul de adresare. De exemplu, dacă sunt disponibili în cuvântul de adresare biţii $A_{13}, A_{12}, A_{11}, A_{10}$ care sunt repartizaţi pentru selectare respectiv a circuitele ROM3, ROM2, ROM1,şi ROM0, pentru modelul de 4K din Figura 2.51-a, se obţin prin construirea mapei adreselor memoriei următoarele segmente nealăturate de adrese: ROM3, 2000H-23FFH; ROM2, 1000H-13FFH; ROM1, 0800H-0BFFH; ROM0, 0400H-07FFH

2.4.7 Dispozitivele logice programabile, PLD

Dispozitivele logice programabile, **PLD** (**P**rogramming **L**ogic **D**evice) au apărut ca o necesitate pentru eliminarea inconvenientelor pe care le prezentau circuitele logice universale MUX, ROM, în implementarea funcţiilor logice sub formă de sumă de produse, şi anume:

- generarea tuturor termenilor canonici produs deşi, în general, nu sunt necesari toţi;
- formele minime/reduse ale funcţiilor nu pot fi implementate. Pentru implementarea acestora trebuie parcurs drumul invers, adică expandarea formelor reduse la formă canonică;
- inexistenţa facilităţii de implementare a funcţiei şi prin negata acesteia, când funcţia negată este mai simplă (unele circuite MUX au atât ieşirea negată cât şi nenegată).

2.4.7.1 Matricea Logică Programabilă, PLA

Prima variantă de dispozitiv logic programabil, PLD, sub formă de matrice logică progrmamabilă **PLA** (**P**rogrammable **L**ogic **A**rray), a fost introdus în anul 1975 de către firma Signetics Inc. Organizarea de principiu al unui circuit PLA, Figura 2.52-a,

poate fi privită ca fiind similară cu a memoriei ROM, numai că de data aceasta nu este programat doar nivelul de OR (codificator) ci şi nivelul AND (decodificator). Această facilitate suplimentară, de programabilitate pe matricea AND, face posibilă generarea numai a unui număr p de termeni produs ($p << 2^n$), nu neapărat canonici de variabile de intrare. În afara celor două matrice programabile, în structurarea unui PLA mai apar: 1) un nivel de bufferare pe intrare, care produce pentru fiecare dintre cele n intrări atât valoarea negată cât şi cea nenegată (acest nivel din punct de vedere logic este compus din n decodificatoare elementare); 2) un nivel de ieşire pe XOR care prin programare poate genera, fie valoarea funcţiei, fie valoarea negată a funcţiei. Deci

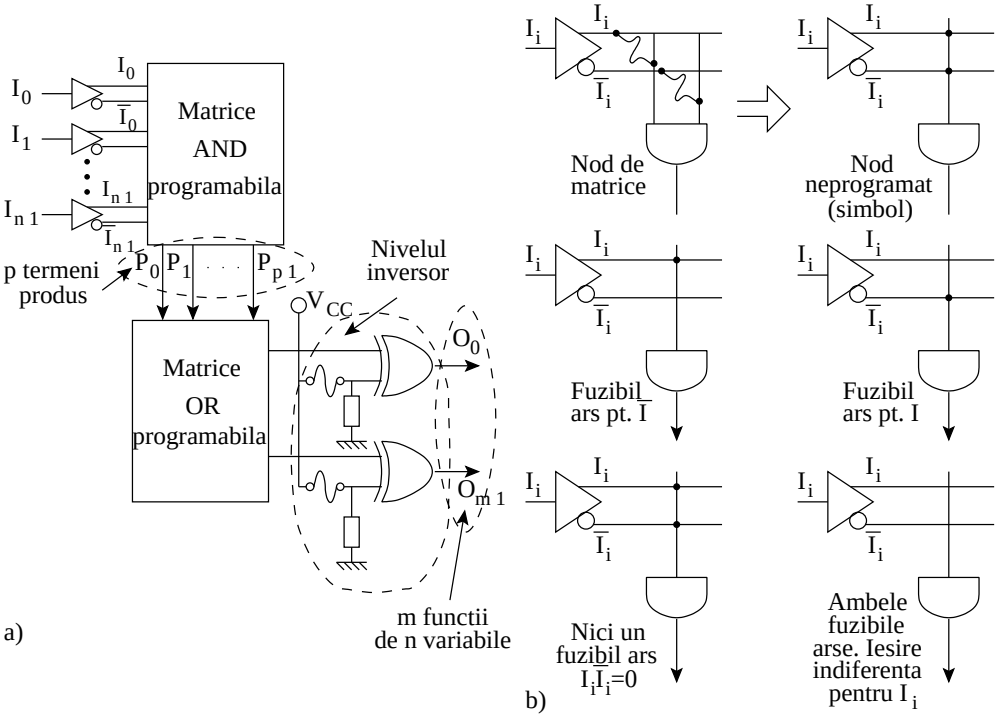


Figura 2.52 Matricea logică programabilă, PLA: a) organizarea de principiu cu evidenţierea celor două niveluri (matrice) programabile; b) simbolurile pentru reprezentarea stării nodurilor din matricea decodificatoare.

circuitul PLA este caracterizat de n intrări, de p porţi AND programabile fiecare cu $2{\times}n$ intrări (n variabile negate şi n nenegate) şi de m porţi OR programabile cu p intrări (care sunt ieşirile porţilor AND). Circuitul PLA poate fi structurat şi pe matrice programabilă numai de tip NAND deoarece implementarea pe două niveluri NAND-NAND este echivalentă cu implementarea pe 2 niveluri AND-OR.

În nodurile matricei AND, de dimensiune $2n$ linii de intrare şi $2n \times p$ coloane şi în nodurile matricei OR, de dimenisune p coloane (termeni produs obţinuţi la ieşirile porţilor AND) şi $p \times m$ linii (m porţi OR fiecare cu p intrări), modalităţile de programare pot fi cele deja descrise la memoria ROM. Se pot realiza PLA-uri, de tipul OTP, când în nod există o diodă înseriată cu un fuzibil ori un tranzistor (bipolar

sau unipolar) înseriat cu un fuzibil, ca în Figura 2.50-a, sau se pot realiza PLA-uri cu reprogramare (ştergere pe cale electrică sau cu fascicol UV), când în nod este prezent un tranzistor cu poarta flotantă, ca în Figura 2.50-b.

În aplicaţii, pentru descrierea nodurilor matricelor programabile, sunt folosite reprezentările logice simbolice din Figura 2.52-b. Un nod în care există o legătură electrică între linie şi coloană este simbolizat cu un punct (nod programat), iar acest punct lipseşte în nodul în care nu există o legătură electrică (nod neprogramat). Evident, dacă ambele noduri, atât pentru intrarea I_i cât şi pentru $\bar{I}_i$, sunt programate

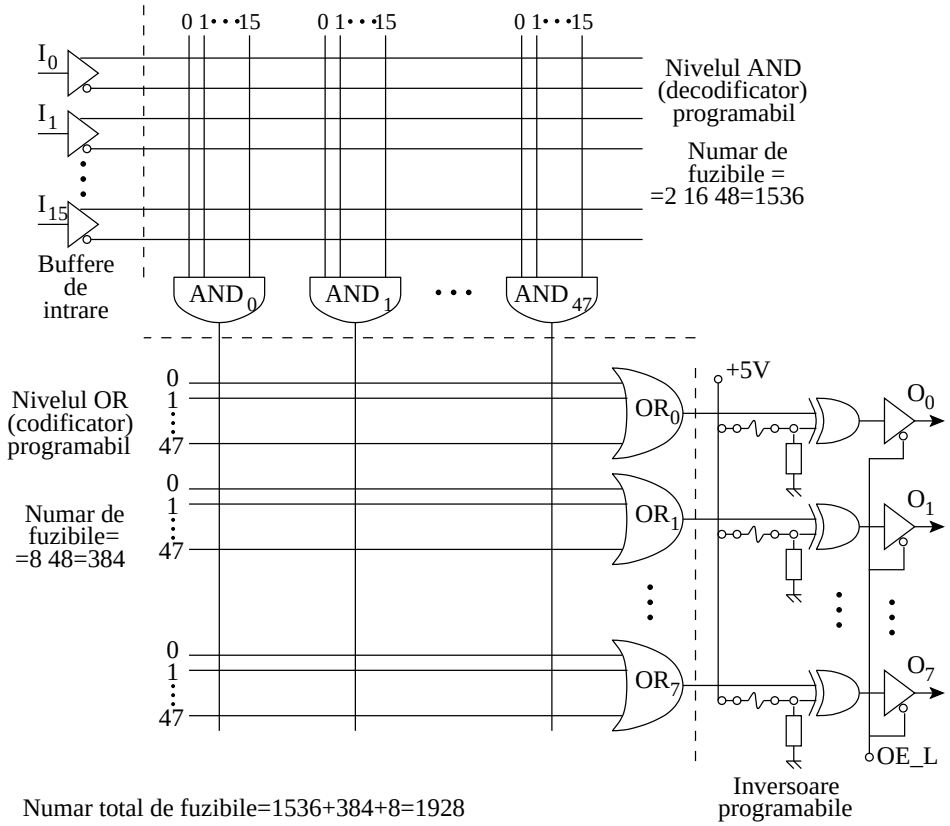


Figura 2.53 Structură tipică pentru un circuit PLA.

poarta AND, în care intră coloanele respective, va genera un termen produs a cărui valoare este permanent zero ($I_i \cdot \bar{I}_i = 0$). Poarta AND în care intră atât nodul I_i cât şi $\bar{I}_i$, dar ambele sunt neprogramate (fuzibilele au fost arse!), va genera un produs ce este indiferent în raport cu intrarea I_i (nu conţine această variabilă).

O structură tipică pentru un circuit PLA (82S100, Signetics) este cea din Figura 2.53, alte variante obtenabile comercial pot fi deduse sau se recunosc în aceasta. Circuitul 82S100 prezintă 16 intrări ($I_0, I_1, ..., I_{15}$), 48 de porţi AND fiecare având câte 32 de intrări. Se pot genera cel mult 48 de termeni produs fiecare de maxim 16 variabile. Cele 8 funcţii care se pot inplementa, ca o sumă de maxim 48 termeni produs, pot fi generate la ieşirile $O_0, O_1, ..., O_7$, fie negate, fie nenegate prin programarea porţilor

XOR. Ieşirile sunt generate prin bufferele de ieşire TSL comandate prin semnalul validare ieşire, OE_L. Numărul total de fuzibile este de 1928; o memorie PROM care poate implementa o funcţie de 16 variablie trebuie să aibă pe nivelul OR programabil $2^{16} = 65536$ fuzibile!

La o memorie ROM o configuraţie binară a cuvântului de intrare generează prin decodificare doar un singur termen canonic produs, pe când la un circuit PLA o configuraţie binară de intrare poate genera nici unul, unul sau mai mulţi termeni produs, respectiv acelaşi termen produs poate fi generat de mai multe configuraţii de intrare. Această neunivocitate rezultă din posibilitatea că unele din intrările porţilor AND să fie programate indiferent, intr-un termen produs, în raport cu anumite variabile (ambele fuzibile ale variabilei sunt arse).

Exemplul 2.19 Să se realizeze sinteza şi să se implementeze pe un circuit PLA un convertor de cod BCD-7 segmente. Notarea segmentelor LED ale afişorului cu 7 segmente corespunde celei din Figura 2.37.

Tabelul 2.2 Tabelul de adevăr pentru convertorul BCD-7 segmente

W	X	Y	Z	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	0	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1
1	0	1	0	-	-	-	-	-	-	-
1	0	1	1	-	-	-	-	-	-	-
1	1	0	0	-	-	-	-	-	-	-
1	1	0	1	-	-	-	-	-	-	-
1	1	1	0	-	-	-	-	-	-	-
1	1	1	1	-	-	-	-	-	-	-

Soluţie. Funcţiile logice a, b, c, d, e, f, g care activează segmentele LED în funcţie de cifrele zecimale exprimate în BCD sunt date în Tabelul 2.2. Pentru configuraţiile binare 1010, 1011, 1100, 1101, 1110 şi 1111, care nu apar nciodată în codul BCD, valorile funcţiilor sunt indiferente. Minimizând corelat cele 7 funcţii, pe diagramele V-K din Figura 2.54, rezultă următorii 7 implicanţi primi care sunt utilizaţi în mai mult decât într-o singură funcţie: $W, Y\bar{Z}, \bar{X}\bar{Z}, Y\bar{X}, \bar{Y}\bar{Z}, X\bar{Z}$ şi $X\bar{Y}$. Aceşti implicanţi primi se generează o singură dată pe matricea programabilă AND şi sunt utilizaţi în matricea programabilă OR, ori de câte ori este nevoie. Pentru implementare s-a structurat un circuit PLA generic cu 4 intrări, 13 porţi AND şi 8 porţi OR.

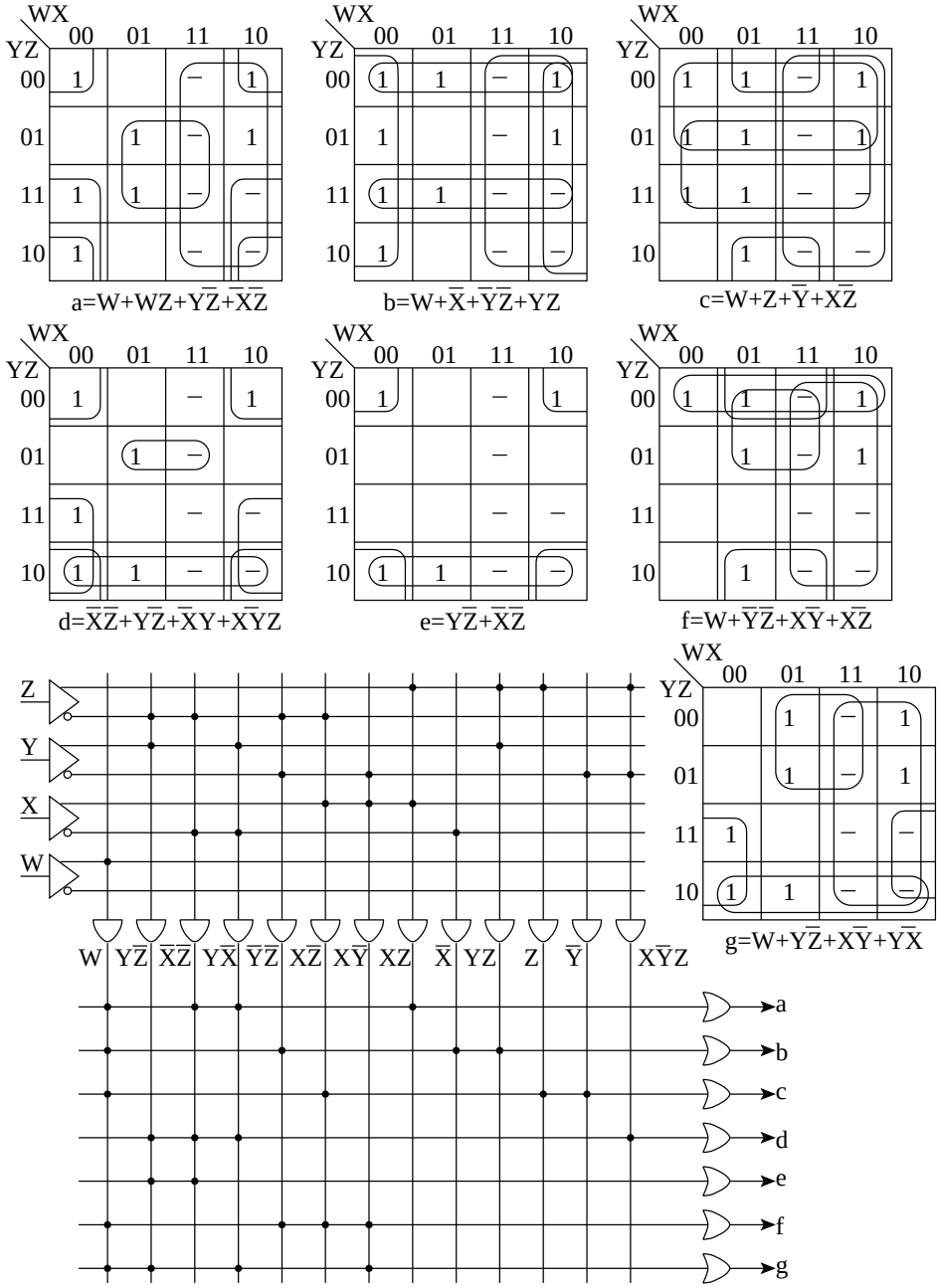


Figura 2.54 Sinteza şi implementarea convertorului de cod BCD-7 segmente pe un circuit PLA (generic).

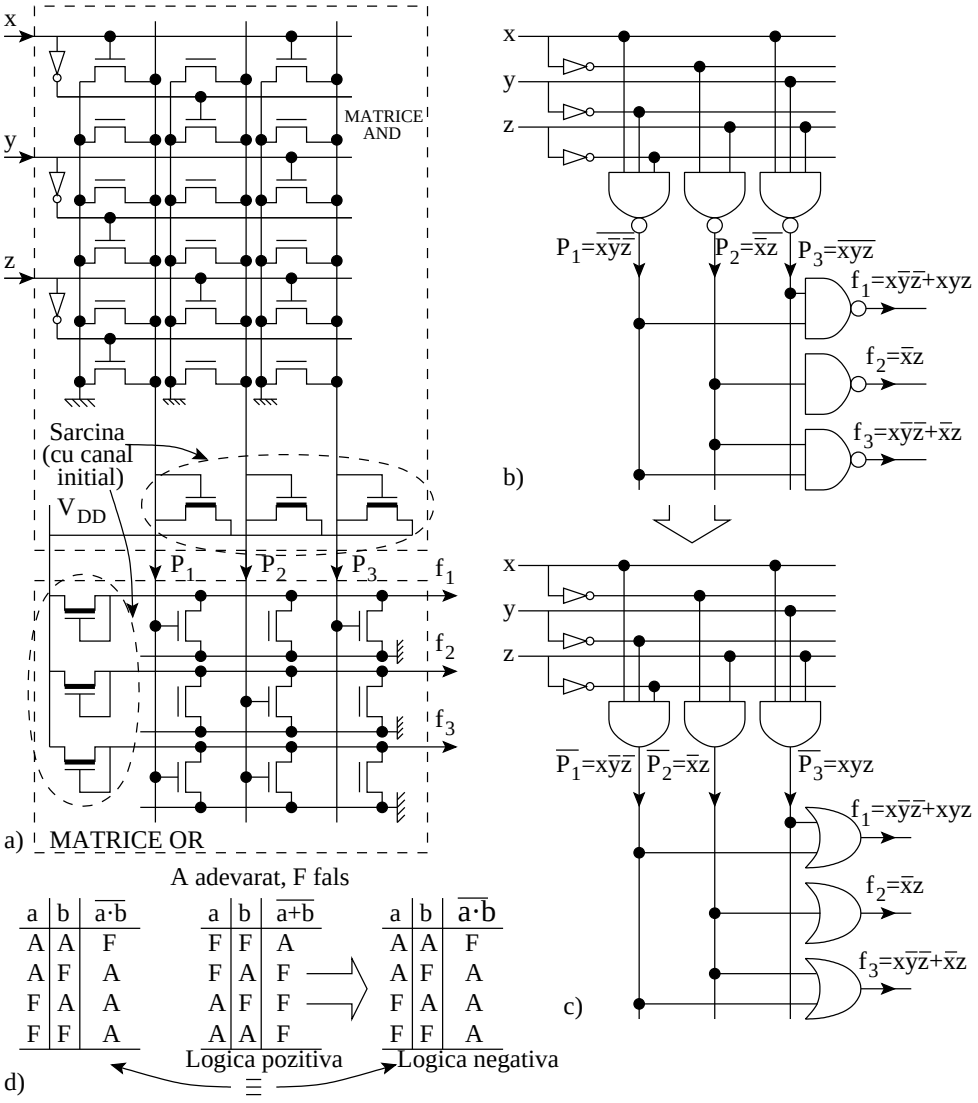


Figura 2.55 Modalitate de implementare a unei matrice PLA: a) sub forma de două matrice de porţi NOR; b,c,d) demonstrarea conversiei NOR - NOR (logică pozitivă) în NAND - NAND (logică negativă) ceea e este echivalent cu AND - OR

Din acest exemplu reiese că realizarea unui CLC pe un PLA poate fi destul de laborioasă. Spre deosebire de circuitele prezentate până acum la circuitul PLA implementarea necesită minimizarea funcţiei, iar dacă se implementează mai multe funcţii, implementarea acestora necesită o minimizare corelată. În mediile de programare automată există programe care, pornind de la funcţia logică, găseşte expresia minimă în sumă de produse atât a funcţiei cât şi funcţiei negate, iar apoi decide care din cele două este mai avantajoasă (are mai puţine produse) şi în funcţie de tipul de circuit PLA utilizat trimite la programator programul pentru programarea nodurilor. Multe circuite PLA dispun de un fuzibil de securitate prin care, după ce este ars, elimină posibilitatea de a se citi mapa nodurilor programate, prin aceasta se exclude posibilitatea de copiere a produsului.

Implementarea unui circuit PLA se realizează cu aceeaşi structură, atât pentru matricea AND cât şi pentru matricea OR, sub forma unei matrice de porţi NOR, ca în figura 2.55-a. O poartă NOR a unei astfel de matrice este compusă din tranzistoare nMOS conectate în paralel şi un tranzistor cu canal iniţial ca rezistenţă de sarcină. Dar o structură de poartă NOR în logică pozitivă va realiza în logică negativă operatorul logic NAND, ceea ce este demonstrat prin tabelele de adevăr din Figura 2.55-d. Considerând logica negativă, la ieşirea primei matrice, conform conexiunilor realizate, se obţin termenii: $P_1 = \overline{x\bar{y}z}$, $P_2 = \bar{x}z$ şi $P_3 = \overline{xyz}$. Şi a doua matrice, în logica negativă, va realiza operatorul NAND, deci la ieşire se obţin funcţiile:

$$f_1 = \overline{P_1 P_2} = \bar{P}_1 + \bar{P}_2 = x\bar{y}\bar{z} + xyz; \quad f_2 = \bar{P}_2 = \bar{x}z; \quad f_3 = \overline{P_1 P_2} = \bar{P}_1 + \bar{P}_2 = x\bar{y}\bar{z} + \bar{x}z$$

Faptul că prima matrice de NOR-uri realizează nivelul de AND, iar a doua nivelul de OR rezultă prin utilizarea cerculeţelor de negaţie ca în Figura 2.55-b şi c. Deplasând cerculeţele de negaţie de la ieşirea porţilor NAND, de la prima matrice, la intrările porţilor NAND de la a doua matrice NAND acestea se transformă în porţi OR, conform conversiei binecunoscute NAND - NAND = AND - OR (vezi secţiunea 2.3).

2.4.7.2 Matricea logică programabilă cu nivel OR fix, PAL

Circuitul **PAL** (**P**rogrammable **A**rray **L**ogic) este o variantă modificată a circuitului PLA. Modificările faţă de PLA constă în: existenţa numai a matricei AND programabilă, matricea OR fixă (neprogramabilă), invers ca la circuitul ROM, şi o facilitate ca unele dintre terminalele circuitului să poată fi utilizate atât ca intrări cât şi ca ieşiri. Aceste modificări reduc flexibilitatea generării funcţiilor logice dar simplifică programarea şi eficientizează utilizarea terminalelor circuitului.

O structură tipică de circuit PAL (PAL16L10) este prezentată în Figura 2.56, care, de fapt, este reprezentarea puţin simplificată a circuitului PAL16L8. În compunerea codului de denumire al circuitului primul număr specifică numărul de terminale de intrare, în cazul acesta 16, iar al doilea este numărul terminalelor de ieşire, aici 8. Circuitul PAL16L8 are 20 de pini (sunt incluşi si cel de masă şi de alimentare), ceea ce înseamnă că numărul total de terminale de intrare şi de ieşire necesar (16 + 8 + 2 = 26) este mai mare decât numărul de pini existenţi (20). Această diferenţă rezultă din posibilitatea de utilzarea a unor pini atât ca terminale de ieşire cât şi ca terminale de intrare.

Există grupuri de câte 8 porţi logice AND programabile, fiecare poartă din grup are 32 de intrări pentru 16 variabile de intrare (negate şi nenegate). Din fiecare

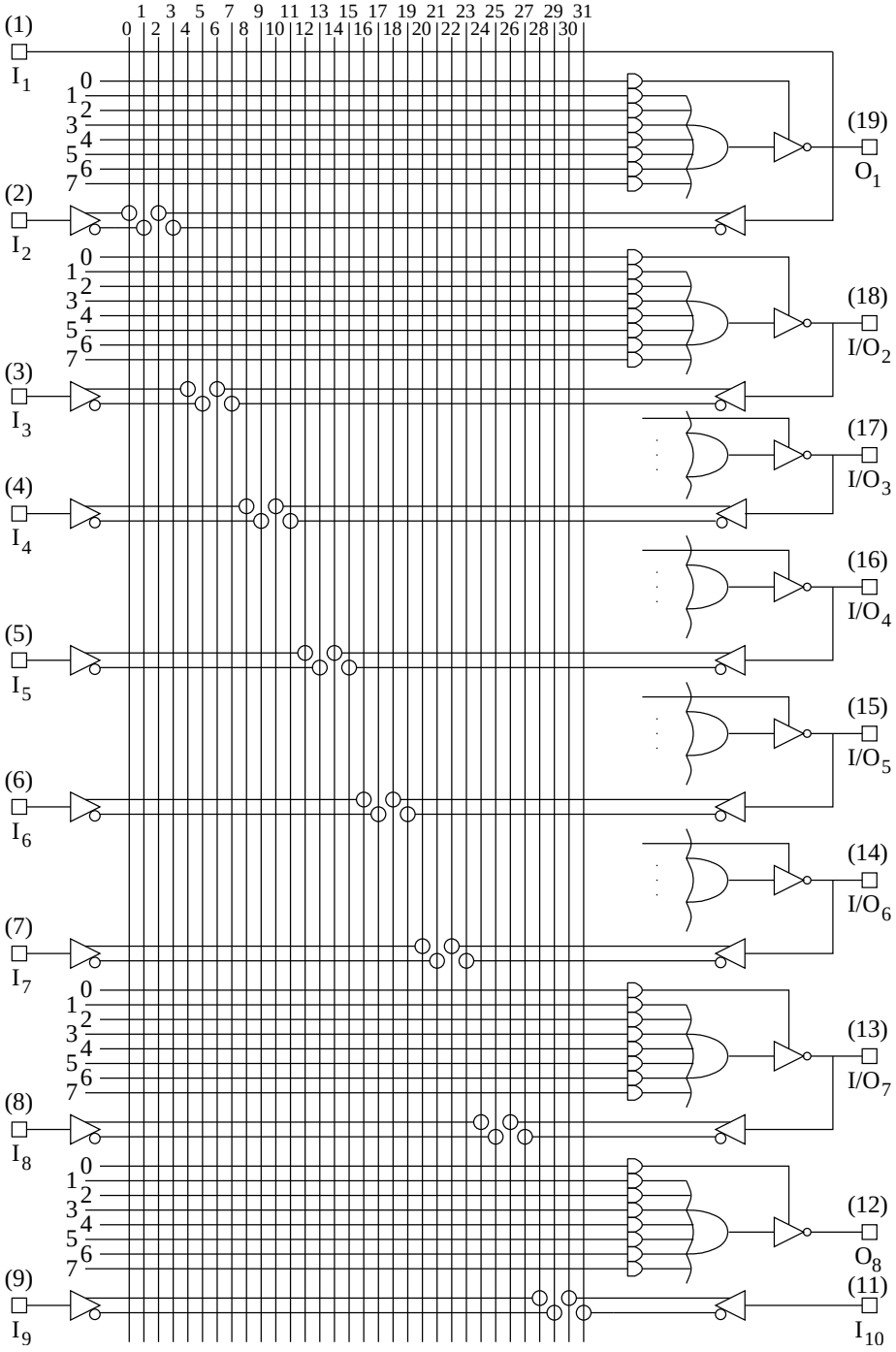


Figura 2.56 Structura tipică de circuit PAL (PAL16L10).

grup, câte 7 porţi AND au ieşirile cablate la intrările unei porţi OR iar a opta poartă AND din grup, poarta de validare a ieşirii, comandă prin ieşirea sa starea de funcţionare a unui buffer inversor TSL, care este conectat la ieşirea porţii OR. Un număr de 10 terminale, $I_1, I_2, ..., I_{10}$, sunt utilizate numai pentru aplicarea variabilelor de intrare, două terminale O_1 şi O_8 sunt numai terminale de ieşire, iar 6 terminale, $I/O_2, I/O_3, I/O_4, I/O_5 I/O_6$ şi I/O_7, pot fi programate atât ca terminale de intrare cât şi ca terminale de ieşire. De pe cele 6 terminale bidirecţionale semnalul de ieşire este introdus şi în reţeaua programabilă AND printr-un buffer de intrare, similar ca de la oricare terminal de intrare. Numărul de fuzibile de pe nivelul AND programabil este 32 intrări în poartă $\times 8$ porţi $\times 8$ grupuri $= 2048$.

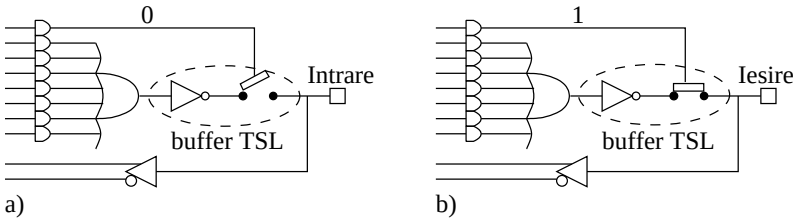


Figura 2.57 Explicativă pentru posibilitatea de transfer bidirecţional la un terminal I/O: a) utilizarea ca terminal de intrare; b) utilizarea ca terminal de ieşire.

Valoarea logică generată de poarta de validare, a opta poartă AND din grupurile care au un terminal de tipul I/O, determină dacă pinul terminalului respectiv este utilizat pentru intrare sau pentru ieşire. Poarta AND de validare a ieşirii poate fi programată să genereze permanent la ieşirea sa valoarea 0 ceea ce înseamnă că bufferul inversor TSL este în starea de înaltă impedanţă (HZ), deci terminalul este un terminal de intrare, Figura 2.57-a, sau poate fi programată să genereze permanent 1 ceea ce înseamnă că bufferul de ieşire are o funcţionare normală de poartă inversor, deci terminalul este de ieşire, Figura 2.57-b. Sau, poarta de validare poate avea o valoare pe ieşirea sa, care se calculează în funcţie de configuraţiile binare aplicate pe intrarea sa, caz în care terminalul în timp îşi modifică starea corespunzător (alternând starea de terminal de intrare cu cea de terminal de ieşire).

Când bufferul inversor are ieşirea sa în starea HZ spre pinul I/O corespunzător, acesta este un terminal de intrare deci se aplică o variabilă de intrare. Deci în total, pot fi aplicate maximum $16 (= 10 + 6)$ variabile de intrare.

Dacă bufferul inversor TSL are funcţionare normală (I/O este terminal de ieşire) este posibil ca o funcţie de maximum 7 termeni produs, obţinută la ieşirea porţii OR, să fie generată la pinul corespunzător (terminal de ieşire) sau să fie aplicată înapoi în matricea AND programabilă, ca variabilă de intrare. Această facilitate de aplicare înapoi apare ca o soluţie pentru situaţiile când funcţia de implementat este o sumă de mai mult de 7 termeni produs. În astfel de cazuri, funcţia se partajează într-un grup de 7 termeni produs şi alte grupuri de maximum 6 termeni produs; la prima trecere se calculează pe un grup de 7 porţi AND şi o poartă OR (cu conexiunea fixă între acestea) suma de 7 termeni produs care apoi se reintroduce în reţeaua AND. Pe un alt grup la această sumă de şapte termeni se mai adaugă, prin sumare, alţi 6 termeni produs, iar rezultatul se introduce iarăşi în reţeaua AND; reintroducerile pot continua până la sumarea tuturor termenilor produs ai funcţiei. Aceste treceri

repetate prin reţea măreşte timpul de calcul pentru funcţie. Valori curente pentru timpul de propagare, de la oricare intrare la oricare ieşire, sunt sub 10ns. De asemenea, această facilitate de reintroducere (feedback) a unei valori calculate dă posibilitatea implementării circuitelor secvenţiale.

2.4.7.3 Circuitul de tip GAL

Circuitul **GAL** (**G**eneric **A**rray **L**ogic) poate fi privit ca un circuit PAL la care s-au introdus anumite facilităţi pentru o extindere a posibilităţilor de utilizare (a fost introdus de Lattice Semiconductor). Codul de denumire este similar cu cel PAL, de exemplu GAL20V8 indică 20 de intrări şi 8 ieşiri.

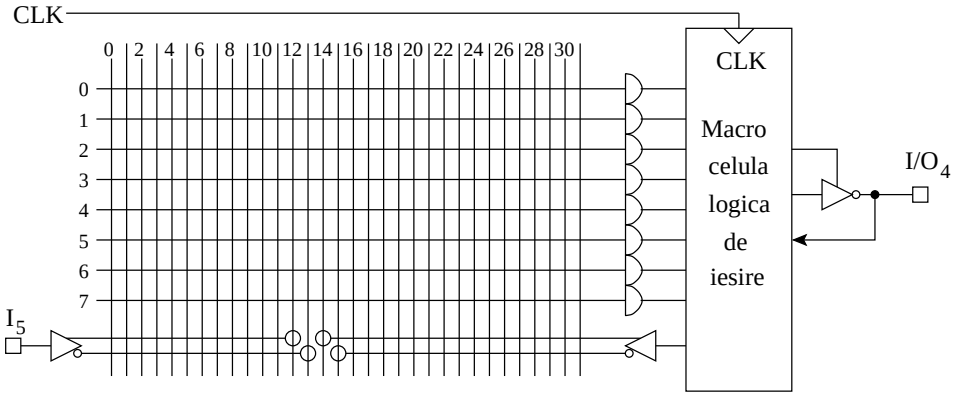


Figura 2.58 Mărirea posibilităţilor de procesare/utilizare a semnalului asignat unui terminal I/O prin introducerea unei macrocelule de ieşire.

Astfel de facilităţi apar pe terminalele bidirecţionale I/O prin îmbogăţirea circuisticii respective – denumirea pentru această circuistică este de macrocelulă de ieşire. În Figura 2.58 este desenat un terminal I/O de la un circuit GAL, care este, de fapt, grupul de celule AND corespunzătoare terminalului I/O_4 de la structurarea tipică de PAL din Figura 2.56, dar acum are pe ieşire o **macrocelulă de ieşire**. Un circuit GAL are atâtea astfel de grupuri, care au ieşirea pe o macrocelulă, câte terminale I/O prezintă. Din această figură se observă că în macrocelulă s-a inclus şi poarta OR, colectoare de termeni produs, care uzual are la ieşirea sa un XOR pentru a putea selecta polaritatea, adică: fie funcţia, fie funcţia negată. În circuistica macrocelulei sunt incluse: buffer de ieşire TSL; celulă pentru validarea ieşirii (bufferului); multiplexoare pentru diferite selectări de semnale; latch-uri pentru memorarea de semnale şi evident, cale de reintroducere a semnalului, obţinut la inşirea porţii OR (sumă de produse), în matricea programabilă AND (vezi secţiunea 4.5).

Circuitul GAL poate implementa o gamă mare de circuite combinaţionale şi secvenţiale.

2.5 CIRCUITE COMBINAŢIONALE PENTRU FUNCŢII NUMERICE

Procesarea informaţiei, în sistemele digitale, se bazează pe algoritmi ce utilizează într-o pondere ridicată operaţii aritmetice. În consecinţă, o mare parte din circuistica sistemelor digitale este construită din circuite cu funcţii numerice. În raport cu circuitele logice, în general, la circuite numerice gradul de replicare este mai mare. Această replicare apare în urma faptului că operaţiile aritmetice se realizează asupra cuvintelor cu lungime relativ mare şi, mai mult, operaţia respectivă se aplică identic asupra fiecărei pereche de biţi din cuvintele procesate. În consecinţă, la circuitele cu funcţii numerice metoda generală de sinteză se bazaeză pe identificarea unui circuit elementar/celulă care procesează o pereche de biţi, urmând apoi replicarea până la nivel de cuvânt (vezi secţiunea 2.2.5). Prezentarea, în continuare, a circuitelor cu funcţii numerice se va face prin identificarea circuitului elementar şi, apoi extensia lui prin replicare.

2.5.1 Comparatorul

Circuitul comparator determină identitatea între biţii de acelaşi rang a două cuvinte. Pe lângă relaţia de egalitatea a celor două cuvinte comparate, se poate determina şi o relaţie de mai mare, mai mic. În general, prin comparator — sub formă de circuit integrat discret — se înţelege circuitul care determină pentru două numere, exprimate în binar, relaţiile de ordine $=, <, >$; deci calculează funcţia de egalitate F_e, de inferioritate F_i şi de superioritate F_s. Sinteza unui comparator, de exemplu, pentru două cuvinte de patru biţi, $A = A_3A_2A_1A_0$ şi $B = B_3B_2B_1B_0$, după metoda normală de sinteză pornind de la tabelul de adevăr ar fi destul de greoaie. O astfel de sinteză ar necesita pentru fiecare dintre funcţiile F_e, F_s şi F_i câte un tabel cu 256 linii (configuraţii de intrare).

O altă modalitate se sinteză a comparartorului utilizează ideea expusă în secţiunea 2.2.5, identificarea unei celule repetitive în structurarea circuitului. O astfel de celulă replicabilă, ce se poate identifica, este celula comparator pentru doua cuvinte A şi B de câte un singur bit, care generează cele trei funcţii f_e, f_i şi f_s. Sinteza comparatorului de doi biţi este simplă pentru ca se pleacă de la un tabel de adevăr cu numai patru configuraţii de intrare, Figura 2.59-a. Funcţiile obţinute $f_e = \bar{A} \cdot \bar{B} + A \cdot B = \overline{A \oplus B}$, $f_i = \bar{A} \cdot B$ şi $f_s = A \cdot \bar{B}$ sunt implementate fiecare pe câte o poartă apoi acestea sunt reunite într-un singur circuit — comparatorul pentru două cuvinte de un bit.

Bazându-se pe celula comparator pentru cuvinte de câte un singur bit se poate face sinteza unui comparator pentru cuvinte de n biţi, ca exemplificare se va face sinteza pentru cuvinte de patru biţi; comparatorul de patru biţi este uzual în aplicaţii ca circuit integrat discret. Relaţiile de ordine F se determină din relaţiile de ordine f pentru fiecare pereche de biţi, dar pornind de la perechea cu rangul cel mai ridicat în felul următor:

- relaţia de egalitate F_e, $A = B$ a cuvintelor de patru biţi există când: $A_3 = B_3$ şi $A_2 = B_2$ şi $A_1 = B_1$ şi $A_0 = B_0$ ceea ce formal se exprimă prin:

$$F_e = f_{e3} \cdot f_{e2} \cdot f_{e1} \cdot f_{e0}$$

- relaţia de superioritate F_s, $A > B$ există când: $A_3 > B_3$ sau $A_3 = B_3$ şi $A_2 > B_2$ sau $A_3 = B_3$ şi $A_2 = B_2$ şi $A_1 > B_1$ sau $A_3 = B_3$ şi $A_2 = B_2$ şi $A_1 = B_1$ şi $A_0 > B_0$ ceea ce duce la următoarea expresie logică

$$F_s = f_{s3} + f_{e3} \cdot f_{s2} + f_{e3} \cdot f_{e2} \cdot f_{s1} + f_{e3} \cdot f_{e2} \cdot f_{e1} \cdot f_{s0}$$

- relaţia de inferioritate F_i, $A < B$, se deduce printr-un raţionament asemănător şi are forma

$$F_i = f_{i3} + f_{e3} \cdot f_{i2} + f_{e3} \cdot f_{e2} \cdot f_{i1} + f_{e3} \cdot f_{e2} \cdot f_{e1} \cdot f_{i0}$$

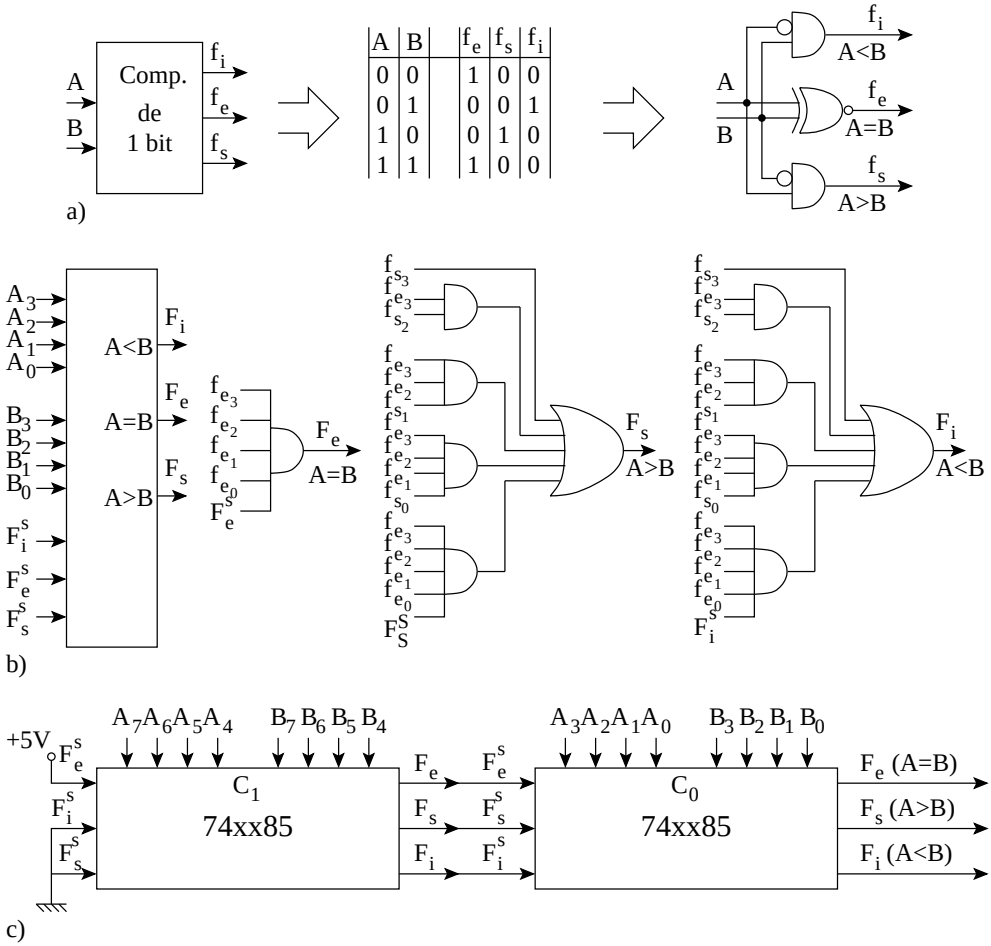


Figura 2.59 Comparatorul: a) circuitul comparator pentru două cuvinte de un bit; b) circuitul comparator pentru cuvinte de patru biţi; c) comparator pentru cuvinte de un byte realizat pe baza circuitelor 74xx85.

Evident că pot fi calculate numai două din cele trei funcţii deoarece fiecare dintre acestea se poate deduce din conjuncţia negatelor celorlalte două funcţii.

$$F_e = \bar{F}_i \cdot \bar{F}_s \quad F_i = \bar{F}_s \cdot \bar{F}_e \quad F_s = \bar{F}_e \cdot \bar{F}_i$$

Totuşi din raţionamente de încărcare şi de a uniformiza timpii de propagare fiecare dintre aceste funcţii se implementează separat ca în Figura 2.59-b. Circuitul comparator de patru biţi trebuie să aibă posibilitatea de a fi utilizat şi ca o componentă în realizarea comparatoarelor pentru cuvinte cu lungime multiplu de patru biţi. Pentru calculul fiecărei dintre cele trei relaţii, de ordine între două numere, se porneşte de la rangurile superioare. Chiar dacă pe un interval continuu de biţi, al celor două cuvinte, relaţia de ordine este îndeplinită evident că relaţia pe cuvântul întreg nu va fi îndeplinită dacă pe intervalul de biţi superior acestui interval relaţia respectivă nu este îndeplinită. În consecinţă, fiecare circuit de patru biţi pentru calculul uneia dintre cele trei relaţii F_e, F_i şi F_s trebuie să primească, ca o validare, de la circuitul intervalului superior de patru biţi, semnalul respectiv al relaţiei de ordine ${F_e}^s$, ${F_i}^s$ şi ${F_s}^s$. Ţinând cont şi de aceasta, relaţiile anterioare de ordine se modifică în felul următor:

$$\begin{aligned} F_e &= f_{e3} \cdot f_{e2} \cdot f_{e1} \cdot f_{e0} \cdot {F_e}^s \\ F_s &= f_{s3} + f_{e3} \cdot f_{s2} + f_{e3} \cdot f_{e2} \cdot f_{s1} + f_{e3} \cdot f_{e2} \cdot f_{e1} \cdot f_{s0} + \\ &\quad + f_{e3} \cdot f_{e2} \cdot f_{e1} \cdot f_{e0} \cdot {F_s}^s \\ F_i &= f_{i3} + f_{e3} \cdot f_{i2} + f_{e3} \cdot f_{e2} \cdot f_{i1} + f_{e3} \cdot f_{e2} \cdot f_{e1} \cdot f_{i0} + \\ &\quad + f_{e3} \cdot f_{e2} \cdot f_{e1} \cdot f_{e0} \cdot {F_i}^s \end{aligned}$$

Circuitul 74xx85 este un comparator care modelează aceste relaţii. În Figura 2.59-c este prezentată o structură pe bază de circuite 74xx85 pentru determinarea relaţiilor de ordine între două cuvinte de un byte. Se observă că semnalele relaţiei de ordine F_e, F_i şi F_s calculate pe circuitul comparator C_1 al intervalului de biţi $7 \div 4$ se aplică sub forma semnalelor ${F_e}^s$, ${F_i}^s$, ${F_s}^s$ la circuitul comparator următor C_0, al intervalului de biţi $3 \div 0$.

2.5.2 Sumatorul

Adunarea este operaţia aritmetică cu frecvenţa cea mai ridicată care se realizează pe echipamentele de calcul electronic, se consideră că şi incrementarea este tot o adunare, când un operand este unu. În consecinţă, există tendinţa justificată ca timpul de realizare al operaţiei de sumare T_Σ, luat foarte adesea ca reper de comparaţie pentru performanţe de viteză ale sistemelor de calcul, să fie cât mai mic. De exemplu, dacă timpul operaţiei de sumare, pe un procesor, este redus de la $3ns$ la $2,5ns$ aceasta înseamnă o creştere de la $3,33 \cdot 10^6$ adunări/s cu încă 670.000 adunări/s. De aceea, structurile de circuite sumator şi celulele componente ale acestora continuă, încă, să fie intens studiate în funcţie de tehnologia de implementare. În această secţiune se vor expune doar structurile fundamentale de circuite sumatoare la care pot fi reduse multe din sumatoarele existente. Pentru o abordare exhaustivă a circuitelor aritmetice recomandăm [Omondi '94].

2.5.2.1 Sumatorul cu Transport Progresiv, STP

Sumarea a două numere binare exprimate sub forma a două cuvinte A şi B cu lungimea de n biţi se realizează prin adunarea fiecărei perechi de biţi A_i şi B_i începând cu perechea A_0, B_0, de rang zero (2^0), până la perechea A_{n-1}, B_{n-1}, de rang $n-1$ (2^{n-1}). Pe fiecare rang adunarea se realizează cu un circuit denumit **celulă sumator**

complet, notată simbolic $\mathbf{\Sigma(3,2)}$. Celula sumator $\Sigma(3,2)$, de exemplu, pentru rangul i (corespunde ponderii 2^i în valoarea numărului) are trei intrări (A_i,B_i şi transportul anterior, C_{i-1}, generat de celula de rang $i-1$) şi două ieşiri (s_i suma, şi C_i, transportul următor, care se aplică la celula următoare, de rang $i+1$, ca transport anterior). În Figura 2.60-a este reprezentată numai celula de rang 2^i a unui sumator la care, de pe cele două magistrale pentru cuvintele de însumat A şi B, se aplică biţii A_i şi B_i şi se generează bitul sumă s_i pe linia i a magistralei S pentru cuvântul sumă.

Există şi **celulă semi-sumator** $\mathbf{\Sigma(2,2)}$, care sumează doar cele două intrări A_i şi B_i, fară transport anterior, şi generează s_i şi C_i. Tabelul de adevăr al celulei $\Sigma(3,2)$ este prezentat în Tabelul 1.6. Expresiile logice deduse, relaţiile 1.15 şi 1.16, în secţiunea 1.14 pentru s_i şi C_i arată că suma s_i este funcţia $PARITATE(A_i, B_i, C_{i-1})$ (are valoarea 1 când un număr impar de intrări sunt 1, Figura 2.19-b) iar transportul următor C_i este funcţia logică $MAJORITAR(A_i, B_i, C_{i-1})$ (are valoarea 1 când cel puţin două din cele trei intrări sunt 1). Rescriem aceste relaţii logice:

$$\begin{aligned} s_i &= PARITATE(A_i, B_i, C_{i-1}) = A_i \oplus B_i \oplus C_{i-1} \\ C_i &= MAJORITAR(A_i, B_i, C_{i-1}) = A_iB_i + A_iC_{i-1} + B_iC_{i-1} = \\ &= \overline{\overline{C_{i-1}(A_i \oplus B_i)} \cdot \overline{(A_i \cdot B_i)}} \end{aligned} \tag{2.17}$$

Ultima formă a relaţiei pentru C_i a fost adusă la o exprimare numai cu operatori NAND pentru a putea fi implementată NAND-NAND, care duce la o structură de circuite mai rapide decât circuitele implementate pe două niveluri neinversoare AND-OR. Conform exprimărilor prin relaţiile 2.17 rezultă pentru celula $\Sigma(3,2)$ implementarea din Figura 2.60-b.

Relaţiile anterioare pentru s_i şi C_i pot fi exprimate şi în felul următor, utile pentru implementare în tehnologie CMOS:

$$\begin{aligned} s_i &= A_i \oplus B_i \oplus C_{i-1} \\ C_i &= A_i \cdot B_i + C_{i-1}(A_i + B_i) \end{aligned} \tag{2.18}$$

Se observă că spre deosebire de relaţiile 2.17, când componenta $(A_i \oplus B_i)$ calculată în expresia lui s_i era utilizată şi în expresia lui C_i, acum fiecare funcţie este calculată independent, ceea ce crează posibilitatea realizării unui lanţ al tuturor circuitelor generatoare de C_i independent de porţi ale generatoarelor de sumă s_i. Implementarea celulei $\Sigma(3,2)$, conform relaţiilor 2.18, este prezentată în Figura 2.60-c; este desenat circuitul electric numai pentru C_i, cel pentru s_i este desenat în Figura 1.64-c. Realizarea celulei sumator complet necesită 32 de tranzistoare.

O altă formă de exprimare pentru s_i şi C_i este următoarea:

$$\begin{aligned} s_i &= A_i \cdot B_i \cdot C_{i-1} + (A_i + B_i + C_{i-1}) \cdot \overline{(A_i \cdot B_i + C_{i-1}(A_i + B_i))} = \\ &= A_i \cdot B_i \cdot C_{i-1} + (A_i + B_i + C_{i-1}) \cdot \overline{C}_{i-1} \\ C_i &= A_i \cdot B_i + C_{i-1}(A_i + B_i) \end{aligned} \tag{2.19}$$

De data aceasta generatorul de sumă utilizează expresia lui $\overline{C}_{i-1}$, calculată de generatorul de transport următor. Structurarea corespunzătoare a celulei $\Sigma(3,2)$ este prezentată în Figura 2.60-d. Realizarea celulei necesită necesită 28 de tranzistoare deoarece în anumite organizări de sumatoare sunt necesare semnalele $\bar{s}_i$ şi $\bar{C}_i$, deci

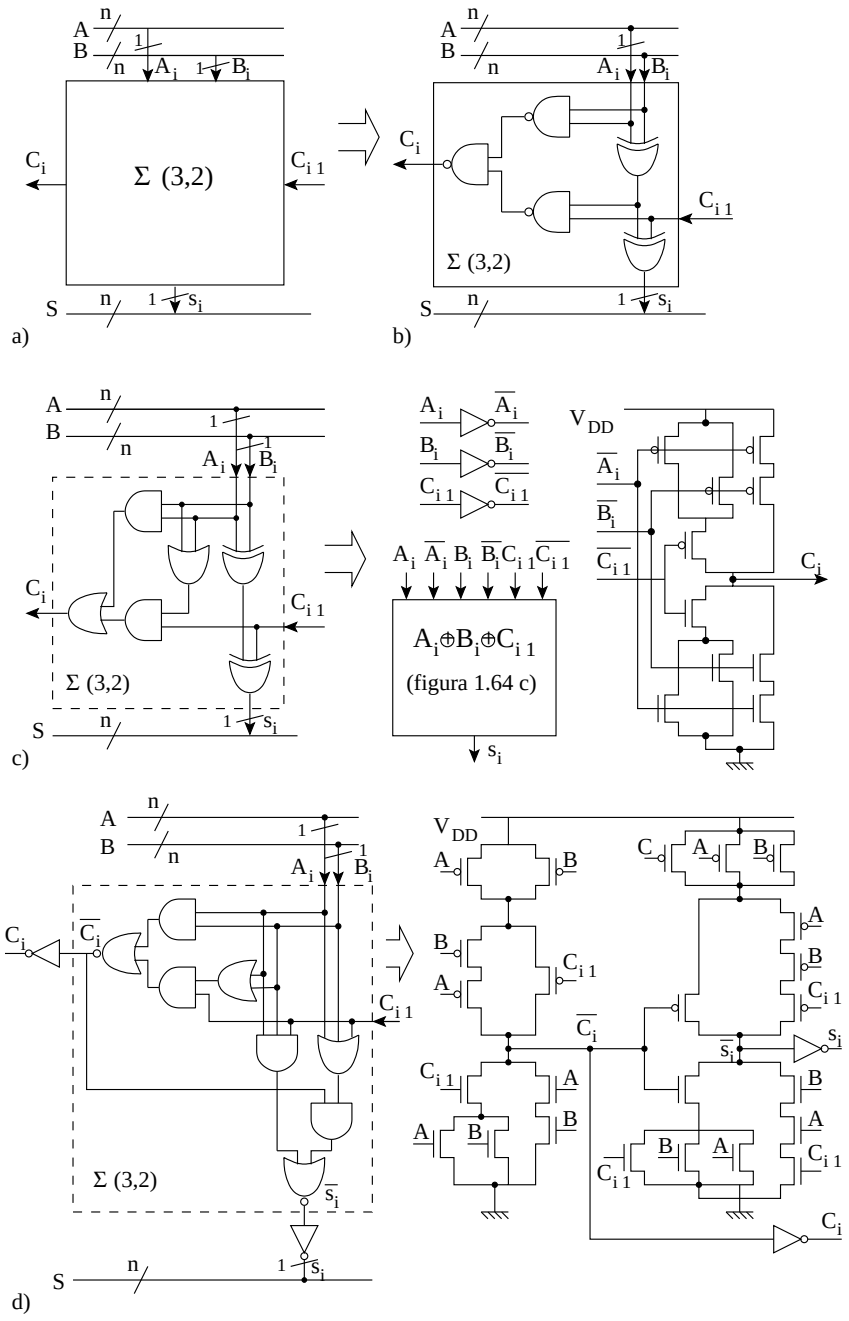


Figura 2.60 Celule sumator complet, $\Sigma(3,2)$: a) modul de conectare a celulei de rang i a unui sumator, la magistralele cuvintelor de însumat (A,B) şi la magistrale cuvântului sumă S; b) structura logică a celulei conform relaţiei 2.17; c,d) structura logică şi implementarea în tehnologie CMOS conform relaţiilor 2.18 şi 2.19.

amplificatoarele de ieşire pentru aceste două semnale pot fi eliminate; structura din figură corespunde cazului de utilizare s_i şi C_i.

Sumatorul cu transport progresiv pentru sumarea a două numere de n biţi, se obţine prin punerea în paralel a n celule sumator complet, dar porţile generatoare de transport următor se înseriază începând de la celula de rang zero până la celula de rang $(n-1)$; transportul următor C_{i-1}, generat de celula de rang $(i-1)$, este aplicat ca transport anterior la celula următoare de rang i, Figura 2.61-a. La aplicarea simultană pe cele două magistrale A şi B a numerelor de însumat şi a transportului de intrare în sumator C_{-1} (la celula de rang zero, în general, se consideră $C_{-1} = 0$) se generează, pe magistrala S, cuvântul sumă rezultat şi la ieşire transportul următor C_{n-1}, de la celula corespunzătoare perechei de biţi cei mai semnificativi. Dimensiunea sumatorului cu transport progresiv este $S_{SUM}(n) = 5 \times n \in O(n)$. Cu o astfel de dimensiune şi o structură, prin replicarea celulei $\Sigma(3,2)$, sumatorul cu transport progresiv este uşor realizabil deoarece rezultă o geometrie pe siliciu simplă şi repetitivă.

În ceea ce priveşte **timpul de sumare $\mathbf{T_\Sigma}$** acesta este tot în $O(n)$ ceea ce pentru n de valori mari, de exemplu pentru lungimile de cuvânt de 64 sau 128 biţi la procesoarele actuale, determină ca acest tip de sumator să nu fie aplicabil. Timpul de sumare T_Σ trebuie să fie mai mare sau egal cu cel mai lung **timp de propagare al transportului τ_Σ** care apare când transportul $C_0 = 1$ generat în celula de rang zero (se consideră $C_{-1} = 0$) se propagă progresiv din celulă în celulă până la celula de rang $n-1$ unde se generează transportul C_{n-1}.Timpul cel mai lung de propagare se obţine, de exemplu, când se adună operanzii $A = 11\ldots11, B = 00\ldots01$ şi se calculează cu relaţia (se consideră că toţi biţii celor două cuvinte se aplică simultan)

$$\tau_\Sigma = \tau_{A_0B_0-C_0} + (n-2)\tau_{C_{(i-1)}-C_i} + \tau_{C_{(n-2)}s_{n-1}} \leq T_\Sigma \tag{2.20}$$

în care:

- $\tau_{A_0B_0-C_0}$ este intervalul de timp din momentul aplicării biţilor A_0, B_0, pe prima celulă de rang zero, până la generarea transportului C_0;
- $\tau_{C_{(i-1)}-C_i}$ este timpul de propagare al transportului pe o celulă, din momentul aplicării semnalului C_{i-1} până la generarea lui C_i;
- $\tau_{C_{(n-2)}s_{n-1}}$ este întârzierea, la celula de rang $n-1$, din momentul aplicării transportului anterior C_{n-2} până la generarea bitului de sumă $s_{(n-1)}$.

Din relaţia 2.20 se deduce că pentru sumatoarele cu n mare reducerea timpului total de propagare este sensibilă la micşorarea propagării transportului pe celula sumatoare, $\tau_{C_{(i-1)}-C_i}$. La un sumator implementat cu celule cu structura din Figura 2.60-d micşorarea componentei $\tau_{C_{(i-1)}-C_i}$ se poate obţine prin eliminarea inversorului de ieşire din partea de generare de transport a celulei, deci se va utiliza numai $\bar{C}_i$ în loc de C_i. Utilizarea lui $\bar{C}_i$ în loc de C_i, pentru micşorarea lui τ_Σ, va impune în organizarea sumatorului ca la celulele din poziţiile pare (şi nu rangurile pare!) să se aplice intrările negate $\overline{A_i}$,$\overline{B_i}$ în loc de A_i şi B_i; dar această alternare, între A_i, B_i şi $\overline{A_i}$,$\overline{B_i}$ când se trece de la poziţii impare la poziţii pare atrage după sine ca şi inversorul de pe ieşirea s_i, al celulelor din poziţii pare, să fie eliminat.

Evident, un sumator nu poate fi comandat pentru o nouă operaţie de sumare decât numai după un interval de timp egal cu timpul de sumare $T_\Sigma \geq \tau_\Sigma$, rezultă că

timpul de propagare al transportului este un parametru limitativ în viteza de lucru a sumatoarelor.

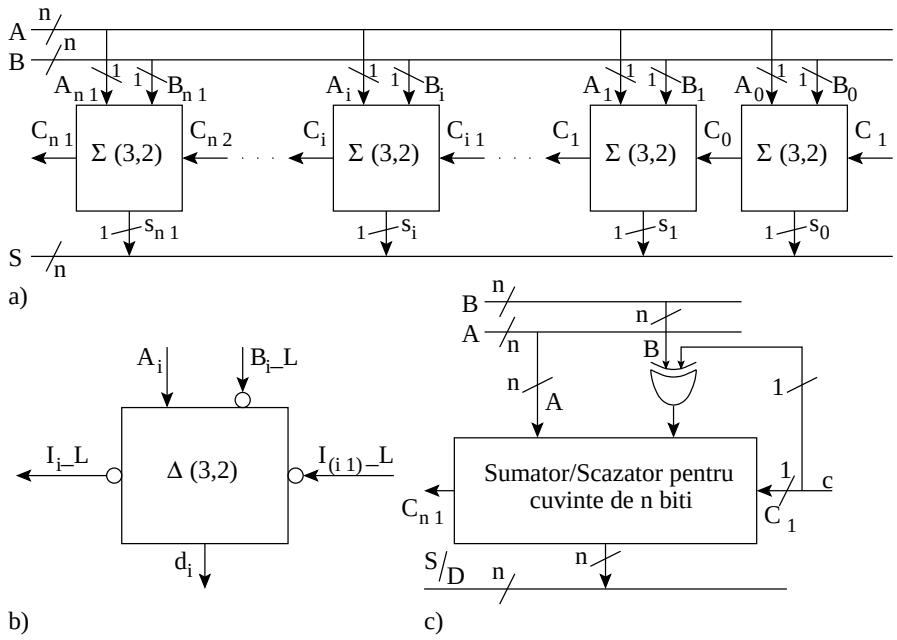


Figura 2.61 Sumatorul cu transport progresiv, STP: a) organizare de principiu pentru un sumator STP de n biţi; b) reprezentarea (schemă bloc) celulei scăzător complet $\Delta(3,2)$; c) organizare de principiu pentru un circuit sumator/scăzător comandat care realizează scăderea pentru $c = 1$ şi adunarea pentru $c = 1$.

În aceeaşi modalitate în care s-a structurat sumatorul se poate face şi structurarea unui scăzător care realizează scăderea numărului B (scăzător) din numărul A (descăzut), $A - B$. Tabelul de adevăr pentru o **celulă scăzător complet, $\boldsymbol{\Delta(3,2)}$**, este prezentat în Tabelul 1.6, are trei intrări (A_i, bitul descăzut; B_i, bitul scăzător; I_{i-1}, împrumutul anterior) şi două ieşiri (d_i, diferenţa rezultată; I_i, împrumutul următor). Prin sinteza pe bază de 1 se obţin relaţiile:

$$\begin{aligned} d_i &= A_i \oplus B_i \oplus I_{i-1} \\ I_i &= \bar{A}_i \cdot B_i + \bar{A}_i \cdot I_{i-1} + B_i \cdot I_{i-1} \end{aligned}$$

pentru care, la prima relaţie, utilizând identitatea $B_i \oplus I_{i-1} = \bar{B}_i \oplus \bar{I}_{i-1}$, iar la a doua aplicând teorema lui De Morgan, se obţin exprimările:

$$\begin{aligned} d_i &= A_i \oplus \bar{B}_i \oplus \bar{I}_{i-1} \\ \bar{I}_i &= A_i \cdot \bar{B}_i + A_i \cdot \bar{I}_{i-1} + \bar{B}_i \cdot \bar{I}_{i-1} \end{aligned} \tag{2.21}$$

Comparând relaţiile 2.17 cu relaţiile 2.21 se poate face o echivalenţă între celula sumator complet şi celula scăzător complet. Această echivalenţă determină următoarea afirmaţie: o celulă sumator complet devine o celulă scăzător complet dacă bitul B_i se consideră activ în starea low, B_i_L (devine scăzătorul), transportul anterior

C_{i-1} se consideră activ în starea low, $I_{(i-1)}\_L$ (devine împrumutul anterior) şi transportul următor C_i se consideră activ în starea low, I_i_L (devine împrumutul următor). Reprezentarea pentru o celulă $\Delta(3,2)$ este dată în Figura 2.61-b. Rezultă că organizarea de sumator cu transport progresiv poate fi transformată într-o organizare de scăzător cu împrumut progresiv dacă: biţii cuvântului B, înainte de aplicare la sumator, sunt complementaţi printr-un inversor (devine scăzător) iar transporturile sunt considerate active în starea low (devin împrumuturile) şi, evident, împrumutul iniţial $I_{-1}\_L = 1$ (invers ca la funcţionarea de sumator $C_{-1} = 0$) este inactiv în H (transportul iniţial C_{-1} era inactiv în starea L).

Aceeaşi transformare a sumatorului în scăzător poate fi realizată pornind de la faptul că o scădere $A-B$ poate fi privită ca o adunare a numărului A cu complementul faţă de doi a numărului scăzător $-B$, $(-B = [B]_2)$, deci $A + [B]_2$. Complementul faţă de doi se obţine din complementul faţă de unu $[B]_1$, care se realizează prin complementarea lui B, la care apoi se adaugă 1, adică $[B]_2 = [B]_1 + 1$. Un inversor comandat de o variabilă de control c se obţine cu o poartă XOR, $B \oplus c$. Se poate genera atât $A + B$ cât şi $A - B$, în funcţie de valoarea variabilei de control, cu următoarea expresie:

$$A + B \oplus c + c = \begin{cases} A + [B]_1 + 1 = A + [B]_2 = A - B & pentru\ c = 1 \\ A + B + 0 = A + B & pentru\ c = 0 \end{cases} \tag{2.22}$$

Implementarea corespunzătoare este reprezentată în Figura 2.61-c, pentru $c = 0$ se realizează adunarea $A = B$, iar pentru $c = 1$ se realizează scăderea $A - B$, deci o structurare de sumator/scăzător, S/D, comandat.

2.5.2.2 Sumatoare de performanţă ridicată

La sumatorul cu transport succesiv, deoarece timpul de sumare T_Σ nu poate fi mai mic decât timpul de propagare a transportului τ_Σ, relaţia 2.20, în întregul lanţ de celule, deci performanţa de viteză este scăzută. De fapt, la orice tip de sumator adunarea obţinută nu poate fi considerată efectuată până nu se calculează corect atât bitul sumă s_{n-1} cât şi bitul de transport următor C_{n-1}. Dar acestea nu pot fi calculate corect până nu se primeşte transportul anterior C_{n-2}, care la rândul său depinde de sosirea transportului C_{n-3} şi aşa mai departe până se ajunge la generarea lui C_0. Toate tipurile de sumatoare, şi există foarte multe, pentru a obţine performanţe de viteză superioare celui cu transport progresiv, prin diferite artificii logice sau de organizare, calculează valoarea C_{n-1} într-un timp mai mic decât cel necesar pentru transportul progresiv din celulă în celulă, relaţia 2.20. În acest sens, se vor prezenta diferite modalităţi de reducere a timpului de calcul pentru determinarea propagării transportului aplicate la trei tipuri de circuite sumatoare denumite: a) sumator cu transport anticipat, b) sumator cu lanţ Manchester şi c) sumator cu selectarea transportului.

a). Sumatorul cu transport anticipat, STA. Ideea transportului anticipat constă în calculul transportului C_{i-1}, pentru obţinerea sumei $s_i = A_i \oplus B_i \oplus C_{i-1}$ la celula sumatoare de rang i, nu în funcţie de valorile anterioare ale transporturilor $C_0, C_1, \ldots, C_{i-2}$ (care necesită timp de propagare) ci în funcţie numai de valorile care se aplică în primul moment la intrările sumatorului adică C_{-1}, A_0 şi B_0, A_1 şi

B_1, ..., A_{i-1} şi B_{i-1}, cum este sugerat din Figura 2.62-a. Pentru acest calcul în avans/anticipat al valorii lui C_i, generat de celula de rang i, se vor utiliza variabilele intermediare p_i – propagare şi g_i – generare introduse pentru o celulă $\Sigma(3,2)$ în Tabelul 1.6 şi care acum se vor redefini în contextul unui sumator.

- Pentru o configuraţie a valorilor biţilor A_i, B_i, aplicată la celula de rang i, există o generare de transport următor $C_i = 1$ atunci când variabila generare are valoarea 1 indiferent de intrările anterioare C_{-1}, $A_{i-1} \div A_0$, $B_{i-1} \div B_0$. Evident, configuraţia de intrare, pentru care $C_i = 1$, este $A_i = 1$ şi $B_i = 1$, deci **variabila intermediară generare** este obţinută prin produsul logic $\mathbf{g_i = A_i \cdot B_i}$.

- Variabila intermediară propagare va avea valoarea $p_i = 1$ pentru acele configuraţii ale valorilor biţilor A_i, B_i pentru care se produce $C_i = 1$ în prezenţa unui transport anterior $C_{i-1} = 1$ (generat de intrările anterioare C_0, $A_{i-1} - A_0$, $B_{i-1} - B_0$). Această propagare prin celulă se realizează când cel puţin un bit de intrare pe culala i are valoarea 1, deci **variabila intermediară propagare** este o sumă logică de intrări $\mathbf{p_i = A_i + B_i}$.

Pe celula i se generează transport următor atunci când $g_i = 1$ SAU atunci când $C_{i-1} = 1$ şi există propagare ($p_i = 1$), rezultă relaţia:

$$C_i = g_i + p_i \cdot C_{i-1} = A_i \cdot B_i + (A_i + B_i) \cdot C_{i-1} \tag{2.23}$$

Dar pentru propagare în loc de relaţia sumă logică se poate utiliza operatorul XOR, $p_i = A_i \oplus B_i$ care acoperă numai configuraţiile $A_i = 1$ şi $B_i = 0$ sau $A_i = 0$ şi $B_i = 1$ nu şi configuraţia $A_i = 1$ şi $B_i = 1$. Oricum, pentru configuraţia $A_i = 1$ şi $B_i = 1$ există transport următor, $C_i = 1$, în relaţia 2.23, chiar dacă acesta nu este obţinut prin propagare ci prin generare $g_i = A_i \cdot B_i = 1$; ceea ce rezultă şi din faptul că următoarea relaţie este o identitate $A_i \cdot B_i + (A_i + B_i) \cdot C_{i-1} = A_i \cdot B_i + (A_i \oplus B_i) \cdot C_{i-1}$. Deci pentru transportul următor este corectă şi relaţia următoare:

$$C_i = g_i + p_i C_{i-1} = A_i \cdot B_i + (A_i \oplus B_i) \cdot C_{i-1} \tag{2.24}$$

care are avantajul că propagarea p_i odată calculată (ca sumă modulo doi) poate fi utilizată şi pentru calculul lui s_i după cum se observă din următoarele două grupuri de relaţii aplicabile unei celule $\Sigma(3,2)$:

grup 1	grup 2
$g_i = A_i \cdot B_i$	$g_i = A_i \cdot B_i$
$p_i = A_i \oplus B_i$	$p_i = A_i + B_i$
$C_i = g_i + p_i \cdot C_{i-1}$	$C_i = g_i + p_i \cdot C_{i-1}$
$s_i = p_i \oplus C_{i-1}$	$s_i = A_i \oplus B_i \oplus C_{i-1}$

(2.25)

Pornind de la relaţia 2.24, pentru un sumator de n biţi, exprimând transportul următor al unei celule sumator în funcţie de transportul iniţial de intrare C_{-1} şi de toate variabilele de propagare şi generare ale celulelor anterioare se obţin expresiile:

$$\begin{aligned} C_0 &= g_0 + p_0 C_{-1} \\ C_1 &= g_1 + p_1 C_0 = g_1 + p_1 g_0 + p_1 p_0 C_{-1} \\ C_2 &= g_2 + p_2 C_1 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 C_{-1} \\ &\cdots \\ C_i &= g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + \cdots + p_i p_{i-1} p_{i-2} \ldots p_3 p_2 p_1 p_0 C_{-1} \end{aligned} \tag{2.26}$$

din care rezultă posibilitatea ca, în paralel, să se calculeze anticipat toate transporturile următoare fără a mai aştepta transportul anterior. Fiecare transport următor, C_i, în aceste relaţii se calculează pe două niveluri logice AND-OR plus încă un nivel logic pentru calculul variabilelor intermediare g_i, p_i (respectiv pe o poartă AND sau o poartă OR/XOR), deci în total trei niveluri logice. Pentru timpul de calcul al sumei s_i se mai adaugă încă un nivel corespunzător porţii XOR, Figura 2.62-a. Rezultă că timpul de sumare T_Σ, egal cu cel al fiecărei celule, este constant şi corespunde parcurgerii a patru niveluri logice indiferent de numărul de biţi ai sumatorului.

Această performanţă de viteză atrăgătoare la STA, timp de sumare constant egal cu patru niveluri logice, este mult diminuată la implementări pentru n de valori mari datorită creşterii dimensiunii, fan-out şi fan-in, capacităţilor parazite şi iregularităţii geometrice pe siliciu (layout). Considerând pentru o poartă XOR o dimensiune dublă faţă de AND sau OR rezultă dimensiunea unui STA:

$$S_{STA}(n) = (n^3 + 9n^2 + 74n)/6 \in O(n^3)$$

Circuitul pentru generarea lui C_{n-1} necesită n porţi AND din care una cu n intrări plus o poartă OR cu $n+1$ intrări (porţile AND sau OR cu $n > 4$ prin asociativitate se realizează pe mai multe niveluri, vezi Exemplul 2.12). De asemenea, fiecare semnal g_i trebuie să comande $(n-i)$ intrări iar p_i trebuie să comande (i+1)(n-1) intrări. În consecinţă, STA-urile sunt limitate, în general, la $n = 4$.

Pentru un STA cu $n = 4$ un circuit pentru generarea lui C_3 este prezentat în Figura 2.62-b. Organizarea circuitului se bazează pe rescrierea expresiei lui C_3 în felul următor:

$$C_3 = g_3 + p_3(g_2 + p_2(g_1 + p_1(g_0 + p_0 C_{-1})))$$

cu o implementare de tip dinamic pe o poarta CMOS (nMOS) domino. Circuite generatoare doar pentru C_2, C_1 sau C_0 se pot obţine din structura circuitului pentru C_3 prin eliminarea succesivă respectiv a perechilor g_2, p_2; g_1,p_1 şi g_0,p_0.

Organizarea de principiu a unui STA este prezentată în Figura 2.62-c, în care sunt indicate cele trei generatoare componente: generatorul g_i, p_i, generatorul de transport următor C_i şi generatorul de sumă s_i. Aceste trei generatoare pentru un rang $i(= 0, 1, 2, 3)$ se obţin prin particularizare în circuitele din figurile 2.62-a şi 2.62-b.

Obtenabile, comercial ca circuite integrate discrete, există circuitele sumatoare 74xx283 şi 74xx83, care sunt STA de patru biţi.

Deoarece este dificil de realizat STA cu n ridicat se poate utiliza avantajul metodei transportului anticipat prin organizarea sumatorului prin înserierea a n/m blocuri cu transportul anticipat, fiecare bloc fiind de m biţi, un astfel de sumator este referit ca **sumator cu transport anticipat pe blocuri, STAB**, cu reprezentarea din Figura 2.62-d. STAB poate fi privit ca un sumator cu transport progresiv care are drept celule module de m biţi cu transport anticipat. Adâncimea pentru această organizare

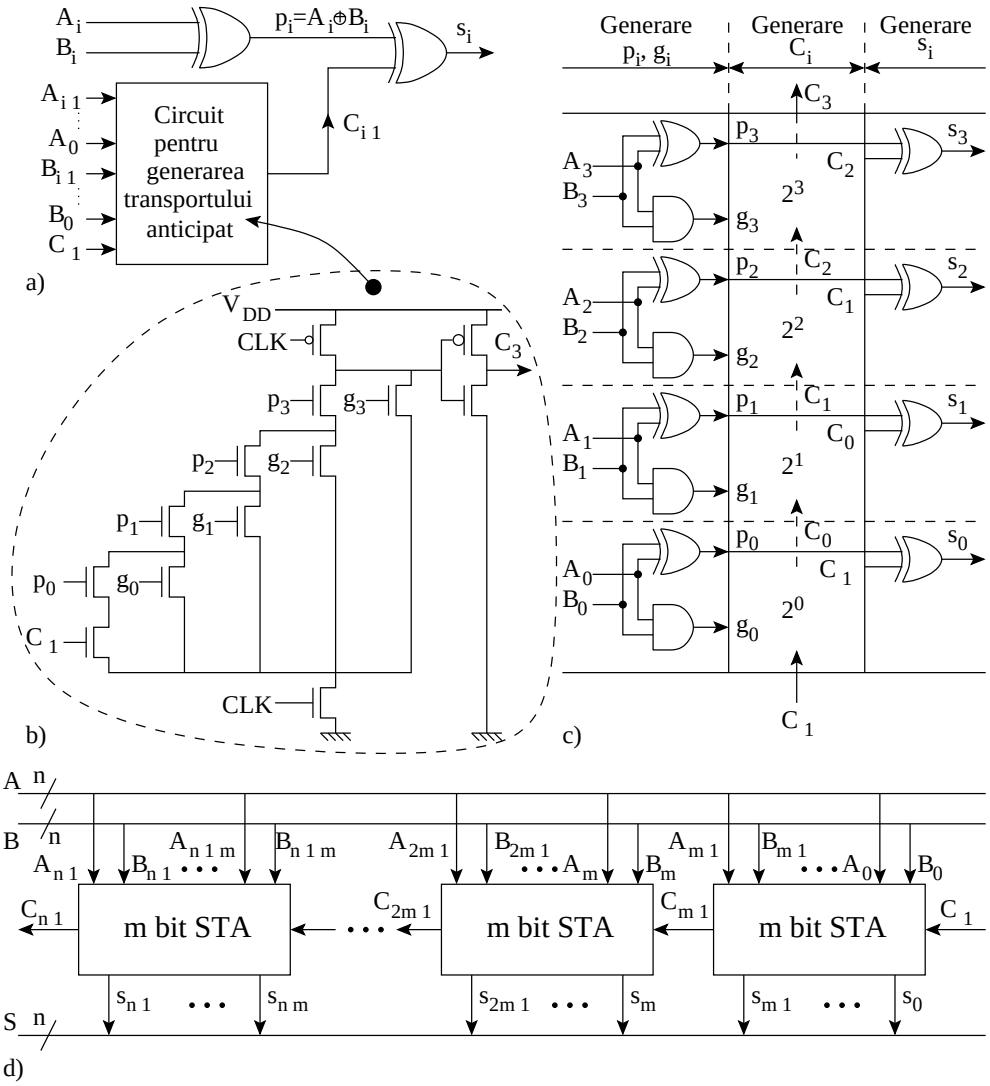


Figura 2.62 Sumatorul cu transport anticipat, STA: a) schema de principiu pentru celula de rang i a unui STA; b) circuitul dinamic CMOS (nMOS) pentru generarea transportului anticipat C_3 peste patru ranguri de sumator; c) organizarea unui STA de patru biţi cu identificare pentru fiecare rang al celor trei generatoare componente (pentru p_i şi g_i, C_i, s_i, iar calculul se face după grup 1 din relaţiile 2.25); d) organizarea unui sumator cu transport progresiv, dar pe bază de blocuri cu transport anticipat, STAB.

este $2(n/m+1)$ niveluri logice, deci intermediar între STP şi STA, iar dimensiunea este n/m ori a unui STA de m biţi.

Se pot concepe şi alte organizări de sumatoare pe bază de blocuri componente STA dar la care se elimină transportul progresiv dintre blocuri şi se realizează pentru câte un grup de blocuri, în exteriorul fiecărui grup, un circuit pentru calculul transportului anticipat. De exemplu pentru numere de 64 biţi, dacă se utilizează module STA de 4 biţi rezultă patru grupuri, fiecare grup de câte patru blocuri, deci patru circuite exterioare, fiecare circuit exterior calculează transportul anticipat pentru câte un grup (pe patru STA de patru biţi). Apoi, peste cele patru circuite de calcul de transport anticipat se poate conecta în exteriorul lor un al cincelea asemenea circuit care, calculează transportul anticipat pe întreg sumatorul de 64 biţi [Wakerly ′2000][Omondi ′94]. Pentru calculul în exterior al transportului anticipat există circuitul 74xx182, Figura 2.76-c.

b). Sumatorul cu lanţ Manchester, SM. Acest tip de sumator, unul din primele utilizate, realizează un traseu separat pentru propagarea transportului. De fapt, şi în organizarea unui sumator cu transport progresiv cu celule ca cea din Figura 2.60-b se realizează o cale continuă (un lanţ) de la C_{-1} până la C_{n-1} pentru propagarea transportului. Rezultă că, un sumator cu lanţ Manchester este un sumator cu transport progresiv cu particularizarea că pe traseul de propagare nu sunt porţi ci comutatoare comandate. În Figura 2.63 sunt prezentate segmentele corespunzătoare celulelor de rang $(i-1)$ şi i din lanţul unui sumator Manchester.

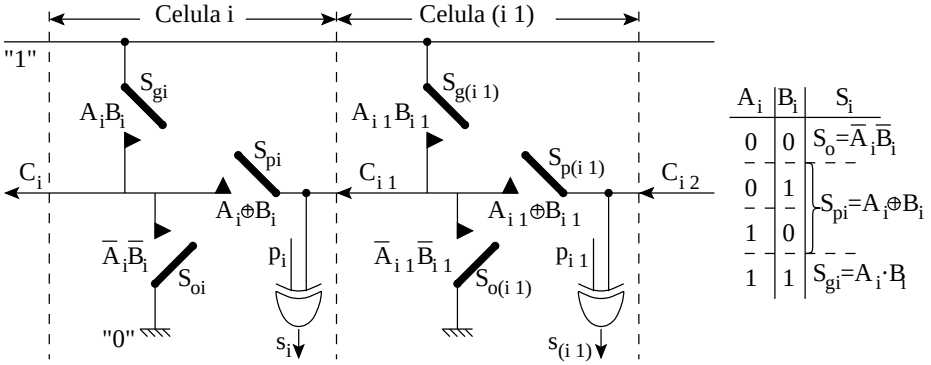


Figura 2.63 Structurarea de principiu a unui sumator cu lanţ Manchester.

Din tabelul de adevăr din această figură se deduce: pentru configuraţia biţilor $A_iB_i = 01$ sau 10 comutatorul S_{pi} (de propagare) conectează C_{i-1} la C_i, iar pentru configuraţia $A_iB_i = 11$ comutatorul S_{gi} (de generare) determină valoarea 1 pentru transportul următor, C_i. Pentru $A_iB_i = 00$ comutatorul S_{0i} aplică valoarea 0 pentru C_i. Evident, comenzile celor trei comutatoare, calculate cu porţi AND, XOR şi NAND cu cei doi biţi A_i, B_i sunt exclusive. Ca elemente comutatoare pot fi utilizate tranzistoare de trecere sau porţi de transmisie CMOS astfel că timpul de propagare este foarte redus.

În anumite implementări, pentru creşterea vitezei, transportul se consideră activ în stare L, aceasta permite ca în intervalul dintre două operaţii de sumare lanţul (fizic) de transfer (de exemplu, din trazistoare de trecere) să fie încărcat la potenţial H,

iar când se realizează sumarea celula, care produce un transfer următor, va descărca lanţul la potenţialul masei. Se alege această variantă deoarece, lanţul fiind echivalent cu o sarcină capacitivă, descărcarea la potenţialul masei se realizează cu o constantă de timp mai mică decât constanta de timp de încărcare la potenţialul H şi prin aceasta rezultă o valoare mai mică pentru timpul de propagare al transportului τ_Σ ($T_\Sigma \geq \tau_\Sigma$). Această schimbare a polarităţii de activare pentru transport determină pentru comutatorul $S_{gi} = A_i \cdot B_i$ să fie conectat la masă, iar comutatorul $S_{0i} = \overline{A_i + B_i}$ să fie conectat la 1 (potenţialul H), în raport cu organizarea din Figura 2.63. Structura de sumator cu lanţ Manchester este simplă şi ordonată ceea ce îl recomandă pentru implementări VLSI.

Ideea de a realiza o cale separată pentru propagarea transportului este utilizată şi la **sumatoarele cu saltul transportului**. La variantele de sumatoare cu saltul transportului se realizează în exterior, peste câte un grup de celule, un traseu de grup separat pentru propagarea transportului care intră în acel grup. Când propagarea pe acel grup de celule (egal cu produsul logic al variabilelor propagare ale tuturor celulelor din grup) are valoarea 1, traseul de grup este comandat în conducţie realizând o cale directă (în exteriorul grupului) pentru transportul de intrare în grup până la ieşirea din grup, eliminându-se astfel timpul lung de propagare al transportului din celulă în celulă. Ideea poate fi dezvoltată în sensul că se poate realiza de asemenea încă un traseu, peste alte câteva trasee de grup, care va fi comandat în conducţie (va scurtcircuita traseele de grup) când sunt în conducţie toate traseele de grup (produsul logic al propagărilor de pe traseele de grup este 1) [Omondi'94].

c). Sumatorul cu selectarea transportului, SST. Atracţia şi performanţele sumatorului cu selectarea transportului sunt datorate ideii simple pe care se bazează funcţionarea sa. Într-un sumator, divizat în blocuri, suma calculată corect pe un bloc nu este realizată până nu soseşte transportul anticipat de la blocul anterior. Dar, acest transport, când soseşte, nu poate fi decât 1 sau 0, deci se pot realiza două blocuri sumator, în paralel, care sumează aceleaşi numere cu deosebirea că unui bloc i se aplică permanent 1 ca transport anterior iar celuilalt 0 ca transport anterior. În momentul când transportul anterior soseşte se va selecta suma deja calculată de la blocul care a avut aplicat transportul anterior egal cu valoarea transportului sosit de la blocul anterior.

Organizarea de principiu a SST este reprezentată în Figura 2.64-a. Blocurile componente pot fi oricare tip de sumator (STP, STA, cu lanţ Manchester, etc.). Primul bloc sumează biţii $A_{m-1} \ldots A_0$ şi $B_{m-1} \ldots B_0$ ai cuvintelor de sumat, iar, în acelaşi timp, următoarele două blocuri sumează tot m biţi din subintervalele $A_{2m-1} \ldots A_m$ şi $B_{2m-1} \ldots B_m$ ale cuvintelor de sumat, dar la unul se aplică $C_{in} = 1$ iar la celălalt $C_{in} = 0$. Aplicând simultan, pe intrările celor trei blocuri, biţii corespunzători ai numerelor de sumat, cu o anumită întârziere (care depinde de tipul de organizare al blocurilor sumatoare), se generează transporturile următoare C_m, ${C_{2m-1}}^0$, ${C_{2m-1}}^1$ şi cele trei sume. Cu ajutorul transportului următor C_m, generat de primul bloc, prin intermediul unui grup de $m\times$MUX2 : 1 se va selecta doar suma de la acel bloc următor care a realizat sumarea pentru $C_{in} = C_m$. În acelaşi timp cu selectarea sumei, se selectează, tot în funcţie de C_m, care din transporturile următoare C^0_{2m-1} şi C^1_{2m-1}, generate respectiv pentru $C_{in} = 0$ şi $C_{in} = 1$, se va aplica la următoarele două blocuri. Acest transport următor selectat va realiza aceleaşi operaţii de selectare pentru următoarele două blocuri sumator ca şi transportul C_m pentru cele două blocuri

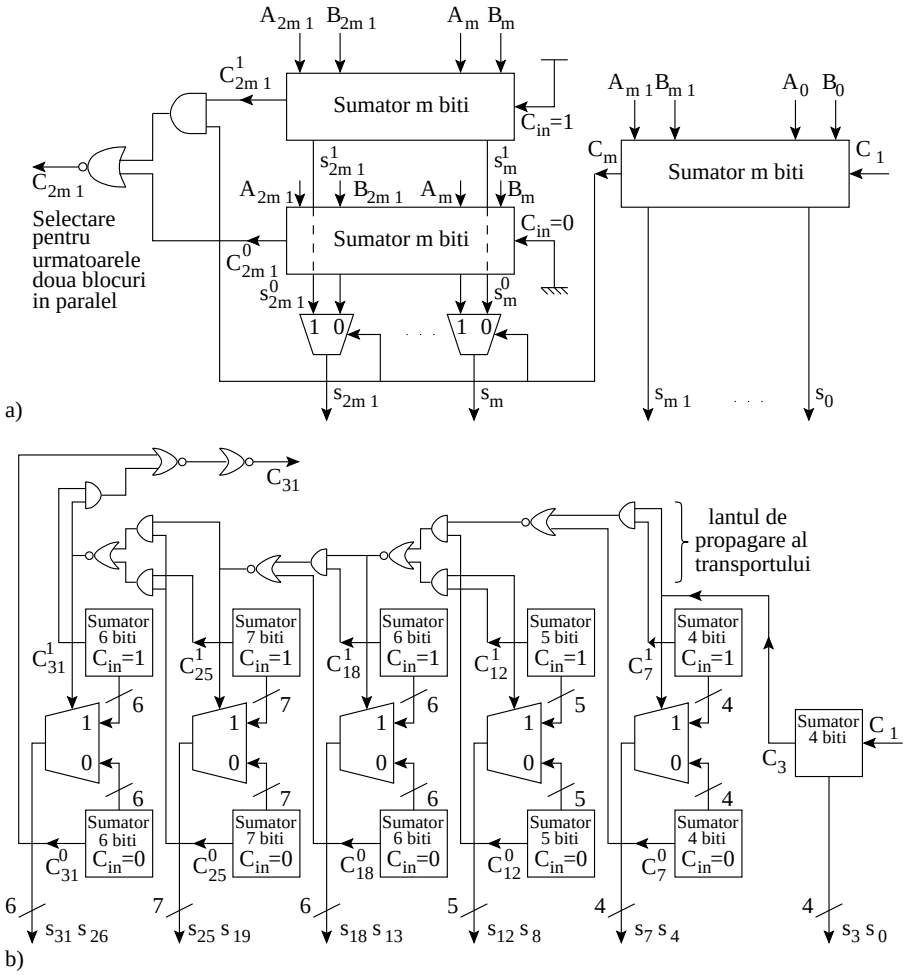


Figura 2.64 Sumatorul cu selectarea transportului, SST: a) la cele două module care funcţionează în paralel, unul cu $C_{in} = 1$ altul cu $C_{in} = 0$, se alege suma, deja calculată, care corespunde valorii transportului C_{in} sosit de la modulul anterior; b) organizarea unui SST pentru cuvinte de 32 biţi.

în paralel din figură.

Pentru a optimiza timpii de propagare, blocurile nu sunt de lungimi egale. Semnalul de selectare pentru două blocuri care funcţionează în paralel, se calculează, în lanţul de propagare al transportului pe două niveluri de porţi, în funcţie de transporturile următoare C^0 şi C^1 de la cele două blocuri anterioare. Aceasta înseamnă, în cazul în care toate blocurile ar avea aceeaşi lungime, că semnalul de selectare ajunge la următoarele două blocuri în paralel cu o întârziere de două nivele logice după ce sumarea pe aceste blocuri s-a efectuat, se presupune că la toate blocurile intrările se aplică simultan. Deoarece sumarea a doi biţi se realizează pe două niveluri logice, rezultă că fiecare pereche de blocuri următoare poate avea o lungime cu un bit în plus faţă de perechea de blocuri anterioare. Aplicând acest mod de optimizare rezultă pentru un SST de 32 biţi următoarele lungimi de module 4-4-5-6-7-6, Figura 2.64-b, ceea ce determină un timp de sumare egal cu întârzierea prin 18 niveluri de porţi, $(4 + 1 + 1 + 1 + 1 + 1) \cdot 2 = 18$. Trebuie observat că pentru semnalul de selectare creşte fun-out-ul pe măsură ce creşte lungimea modulelor componente ceea ce devine un impediment la implementare cu n ridicat, la fel ca la STA.

Tabelul 2.3 Caracteristicile asimptotice ale unor tipuri de sumatoare

Tipul de sumator	Timpul sumare T_Σ	Aria consumată pe siliciu A_{Si}
STP	O(n)	O(n)
STA	O(log n)	O(n log n)
SM	O(n)	O(n)
SST	$O(\sqrt{n})$	O(n)

Pentru sumatoarele prezentate, valorile asimptotice, la creşterea numărului de biţi n a cuvintelor sumate, ale caracteristicilor timp de sumare T_Σ şi aria ocupată la implementarea în siliciu, A_{Si}, sunt prezentate în Tabelul 2.3. Cunoaşterea comportării asimptotice a sumatoarelor este utilă în înţelegerea lor dar, luarea unor decizii de organizare ţinând cont numai de valorile asimptotice, poate ascunde o capcană. În general, organizarea unui sumator se face pe bază de blocuri, ca în Figura 2.62-a sau Figura 2.64-b, a căror lungime nu depăşeşte ordinul unităţilor. Dar pentru blocuri sumatoare cu lungimi de ordinul unităţilor, de exemplu $n = 4$, diferenţele de performanţă sau de implementare între diferitele tipuri de structuri sumatoare, ca cele din Tabelul 2.3, nu sunt aşa de evidente. Rezultă că, aproape indiferent de tipul de bloc folosit, important devine modul cum se organizează/conectează aceste blocuri într-un sumator de lungime ridicată (32,64,128 biţi).

2.5.3 Multiplicatorul

Echipamentele digitale sunt înzestrate tot mai frecvent cu circuite specializate pentru multiplicare; unităţile aritmetice specializate (coprocesoare) sau procesoarele digitale de semnal se numără printre acestea. Procesarea digitală a semnalelor (corelaţia,

convoluţia, filtrare, analiza frecvenţială) este un domeniu care îşi bazează performanţele pe puternice circuite de multiplicare. Circuitul de multiplicare poate fi realizat cu un CLC deoarece rezultatul înmulţirii depinde exclusiv doar de cei doi operanzi. Se vor prezenta trei organizări de circuite de multiplicare (multiplicatorul matriceal, multiplicatorul tip arbore Wallace şi multiplicatorul tabelar). S-a limitat prezentarea numai la aceste trei organizări, pe care le considerăm de bază, deoarece multe circuite multiplicator utilizate pot fi recunoscute ca variante ale uneia din aceste trei.

2.5.3.1 Multiplicatorul matriceal

Metoda de înmulţire a două numere de cinci biţi, $A = A_4A_3A_2A_1A_0$, $B = B_4B_3B_2B_1B_0$, cu creionul pe hârtie după regula comună, învăţată în şcoala primară, este prezentată în Figura 2.65-a. Produsul rezultat cu lungimea de zece biţi, $P = p_9p_8p_7p_6p_5p_4p_3p_2p_1p_0$, se obţine prin adunarea succesivă a produselor parţiale AB_0, AB_1, AB_2, AB_3 şi AB_4, fiecare produs parţial, înainte de adunare, fiind deplasat la stânga cu o poziţie (adică înmulţit cu 2^1). Pentru realizarea unui circuit de multiplicare, conform metodei de înmulţire prezentate, este necesar a se identifica o structură de celulă elementară componentă a acestui circuit. Să analizăm cum este realizat şi utilizat în operaţia de înmulţire, din această matrice, un produs de doi biţi dintr-un produs parţial, de exemplu produsul A_2B_2. Acest produs se realizează simplu printr-o poartă AND, deoarece există identitate între operatorii produs aritmetic şi produs logic.

Acest produs A_2B_2 se însumează cu rezultatul adunării s_{31} de pe aceeaşi coloană, adică suma anterioară între A_4B_0 şi A_3B_1, se însumează cu transportul anterior, C_{12}, provenit de la termenul produs din dreapta de pe aceeaşi linie A_1B_2 şi, în urma acestor însumări, se generează un bit sumă, s_{22}, care se va însuma cu termenul produs următor A_1B_3, de pe aceeaşi coloană, şi se generează un bit de transport următor, C_{22}, care este aplicat la termenul produs din stânga, A_3B_2, de pe aceeaşi linie. Toate acestea corespund cu operaţiile realizate de o celulă sumator $\Sigma(3,2)$, Figura 2.65-b. Rezultă că circuitul multiplicator poate fi compus din $4 \times 4 = 16$ celule elementare, sumator $\Sigma(3,2)$, câte una pentru fiecare termen produs A_iB_j, iar trecerea de la matricea operaţiei de multiplicare la topologia circuitului de multiplicare se face printr-o mapare 1:1, înlocuind fiecare produs de doi biţi cu o celulă elementară, Figura 2.65-c. Se observă că în structura matriceală a circuitului multiplicator la toate celulele de pe o linie se aplică acelaşi bit B_j, $j = 0,1,2,3,4$ al inmulţitorului B şi la toate celulele de pe aceeaşi diagonală se aplică acelaşi bit A_i, $i = 0,1,2,3,4$ al deînmulţitorului A. Această structurare a multiplicatorului, sub formă de paralelogram, poate fi uşor desenată şi sub formă de matrice pătrată pentru a fi potrivită unui layout pe siliciu.

Din organizarea circuitului multiplicator se poate deduce uşor că dimensiunea sa este în $O(n^2)$. Fiecare linie a multiplicatorului este de fapt un sumator cu transport progresiv. Timpul cel mai lung de înmulţire se obţine când o intrare la prima celula A_0B_0 afectează transportul următor de la ultima celula A_4B_4, adică bitul produs p_9, iar pentru această propagare traseul cel mai lung este în lungul sumatorului corespunzător primei linii, prin coloana cu celulele A_3B_1, A_2B_2, A_1B_3 şi apoi prin celulele A_0B_4, A_1B_4, A_2B_4, A_3B_4, A_4B_4 corespunzătoare sumatorului de pe ultima linie (produsul parţial AB_4). Dacă se consideră, pentru simplitate, că întârzierea pe o celulă de la oricare intrare a sa la oricare ieşire este τ_m atunci timpul de multiplicare

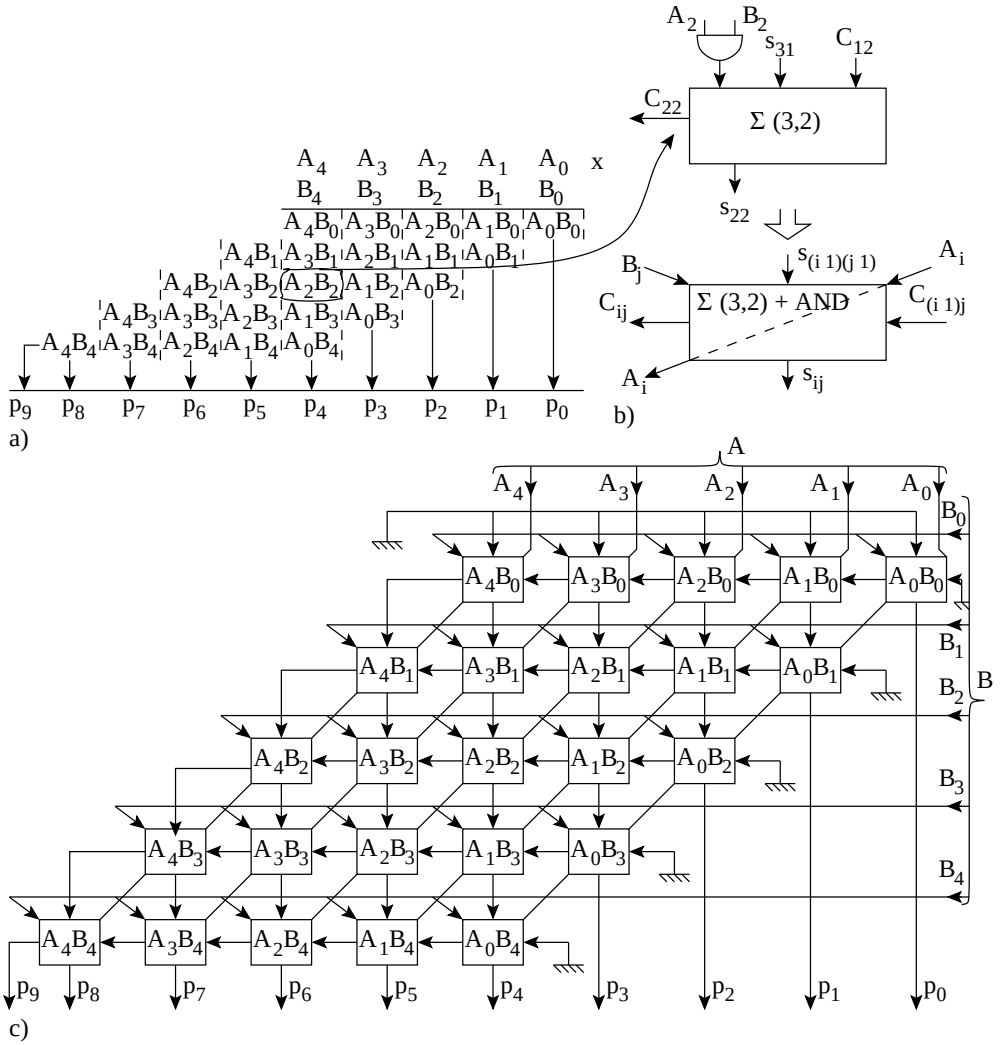


Figura 2.65 Circuitul multiplicator matriceal: a) matricea produselor parţiale obţinută la inmulţirea a două cuvinte A şi B de cinci biţi; b) celula elementară ($\Sigma(3,2)$) care modelează un termen dintr-un produs parţial al inmulţirii; c) structurarea unui circuit multiplicator matriceal pentru $n = 5$.

T_m cel mai mic este de $13 \cdot \tau_m$. Iar în cazul unui înmulţitor pentru două cuvinte de n biţi se obţine $(3n-2)\tau_m \leq T_m$, adică în $O(n)$ ceea ce pentru lungimea actuală de 64 biţi, a numerelor reprezentate în virgulă fixă, determină viteze destul de scăzute.

Modalităţi de a îmbunătăţi performanţa de viteză a multiplicatorului matriceal constă, fie în micşorarea numărului de produse parţiale, deci de adunări ale acestora, fie prin efectuarea a cât mai multe adunări de produse parţiale în paralel sau fie printr-o structurare fără transport progresiv a sumatoarelor.

Din organizarea anterioară de multiplicator matriceal, în tendinţa de a creşte performanţa de viteză, se poate obţine cu mici modificări un multiplicator reprezentat în Figura 2.66 denumit **multiplicator matriceal cu salvarea transportului, MMST**. Mărirea performanţei de viteză rezultă prin însumări de produse parţiale în paralel şi prin utilizarea de sumatoare cu salvarea transportului, SSLT.

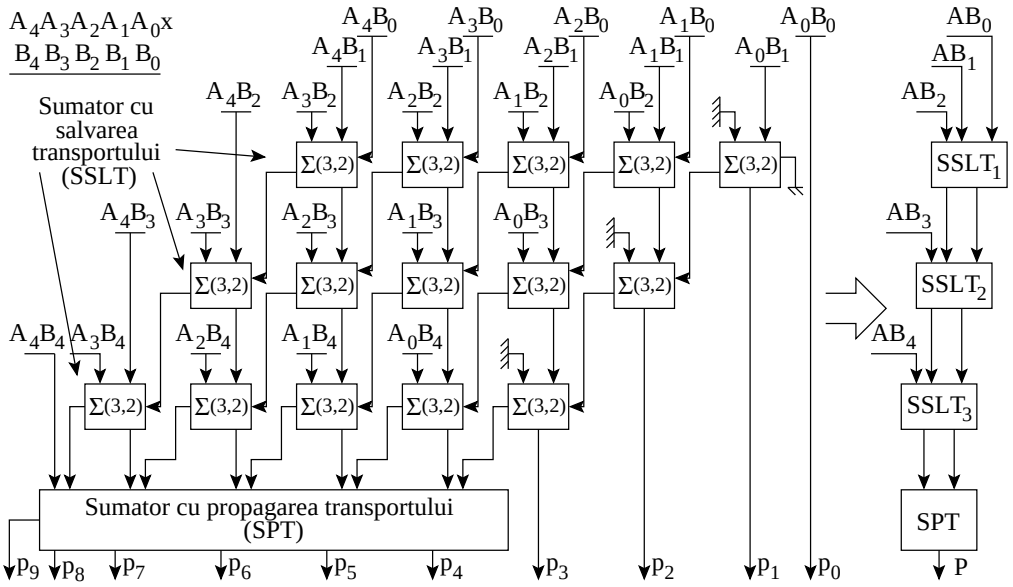


Figura 2.66 Organizarea multiplicatorului matriceal cu salvarea transportului, MMST.

La organizarea matriceală anterioară a multiplicatorului suma parţială după a i-a linie, a produsului parţial AB_{i-1}, poate fi calculată corect numai după ce s-a primit transportul următor de la sumarea parţială de la linia anterioară, adică de la linia produsului parţial AB_{i-2}. Aceste întârzieri în determinarea corectă a sumelor parţiale se datorează faptului că sumatorul de pe fiecare linie este un sumator cu transport progresiv. Se poate ca transportul următor de la celula A_iB_j să nu mai fie aplicat la celula următoare $A_{i+1}B_j$, de pe aceeaşi linie, ci la celula următoare A_iB_{j+1} de pe linia următoare. În felul acesta se obţine, în acelaşi timp, pe fiecare linie a matricei la fiecare celulă o informaţie corectă a celulei, dar exprimată prin perechea: sumă s şi transport C. Structura aceasta de celule sumatoare, în paralel, între care nu există transport progresiv, de pe o linie a matricei care generează perechile s, C, este referită ca sumator cu salvarea transportului. Fiecare linie din matricea

multiplicatoare formează un **sumator cu salvarea transportului, SSLT** de n biţi. Dacă se consideră, pentru simplitate, că întârzierea pe o celulă de la oricare intrare a sa la oricare ieşire este τ_m, înseamnă că din momentul aplicării pe intrări a operanzilor A şi B, după o întârziere egală cu $(n-1)\cdot\tau_m$ se obţin perechile de valori s, C la ieşirea sumatorului cu salvarea transportului de pe penultima linie a matricei. Apoi, introducând aceste perechi, s, C, într-un sumator cu propagarea transportului, cu timpul de sumare T_Σ, se obţine în ultima linie cuvântul produs corect. Sumatorul cu propagarea transportului poate fi oricare tip de sumator prezentat în secţiunea 2.5.2. Timpul de multiplicare T_m respectă relaţia $T_m \geq (n-1)\tau_m + T_\Sigma$ şi este mai mic cam de trei ori comparativ cu cel al multiplicatorului matriceal prezentat anterior, dar tot în $O(n)$ (evident T_Σ depinde de tipul de sumator cu propagarea transportului folosit). Dimensiunea multiplicatorului se calculează pe matricea celor $n(n-1)$ celule plus pe sumatorul cu propagarea transportului, dar se situează în $O(n^2)$.

Această organizare de multiplicator cu salvarea transportului mai poate fi îmbunătăţită prin utilizarea facilităţilor de sumare paralelă. Se observă că celulelor sumatoare din prima linie li se pot aplica pe intrări trei produse parţiale AB_0, AB_1 şi AB_2 (ca în Figura 2.66) eliminând astfel din organizarea matriceală anterioară două linii. Pentru multiplicatorul de $n = 5$ biţi din figură rezultă $T_m \geq 3\tau_m + T_\Sigma$, iar pentru cazul general $T_m = (n-2)\tau_m + T_\Sigma$. Multiplicatorul matriceal cu salvarea transportului duce la o regularitate a geometriei de realizare, deci este indicat pentru implementările VLSI.

2.5.3.2 Multiplicatorul tip arbore Wallace

Performanţa de viteză, la multiplicatorul tip arbore Wallace, se obţine printr-o sumare paralelă de produse parţiale într-o structură de arbore care realizează o **compresie în raportul 3:2**. Pe un sumator cu salvarea transportului, SSLT, se pot aplica simultan trei produse parţiale şi se pot obţine două cuvinte: cuvântul sumă şi cuvântul transport; un SSLT este format din celule $\Sigma(3,2)$ neînseriate prin intermediul transportului anterior. De fapt celula $\Sigma(3,2)$ poate fi privită ca un numărător de biţi 1 conţinuţi în cuvântul de intrare, cum este prezentat în ultimele trei coloane din Tabelul 1.6. De exemplu, dacă cele trei intrări la celulă sunt $A = 1$, $B = 0$, $C = 1$ numărul de biţi 1 ai cuvântului 101 este exprimat codificat în binar de perechea obţinută a ieşirilor C, s, care este pentru acest caz egală cu 10 adică 2; sau pentru $A = 1$, $B = 0$, $C = 0$, care formează cuvântul de intrare 100 în celula sumator complet, perechea obţinută C, s, are valoarea 01, adică 1 în binar natural.

În Figura 2.67-a este prezentat, pentru înmulţirea a două cuvinte A, B cu lungimea de şase biţi, modul cum se pot grupa, succesiv, câte trei cuvinte care aplicate la un SSLT va genera numai două cuvinte.

Structura arborelui de tip Wallace, pentru $n = 6$, este dată în Figura 2.67-b. Fiecare din cele şase produse parţiale se obţine uşor pe câte un grup de şase porţi AND2, deci în total 36 de porţi AND2. Structurarea arborelui urmăreşte succesiunea aplicării tripletelor la sumatoarele cu salvarea transportului. Pe primele două sumatoare cu salvarea transportului SSLT1 şi SSLT2 se aplică simultan Triplet1 şi Triplet2. Pe nivelul doi există un singur sumator SSLT3 care din Sumă2, Transport1 şi Sumă1 generează Transport3 şi Sumă3. Sumatorul SSLT4, de pe nivelul trei, realizează conversia de la Transport2, Transport3 şi Sumă3 la Transport4 şi Sumă4. Ultimul nivel care sumează Transport4 cu Sumă4 trebuie să fie un sumator cu propagarea

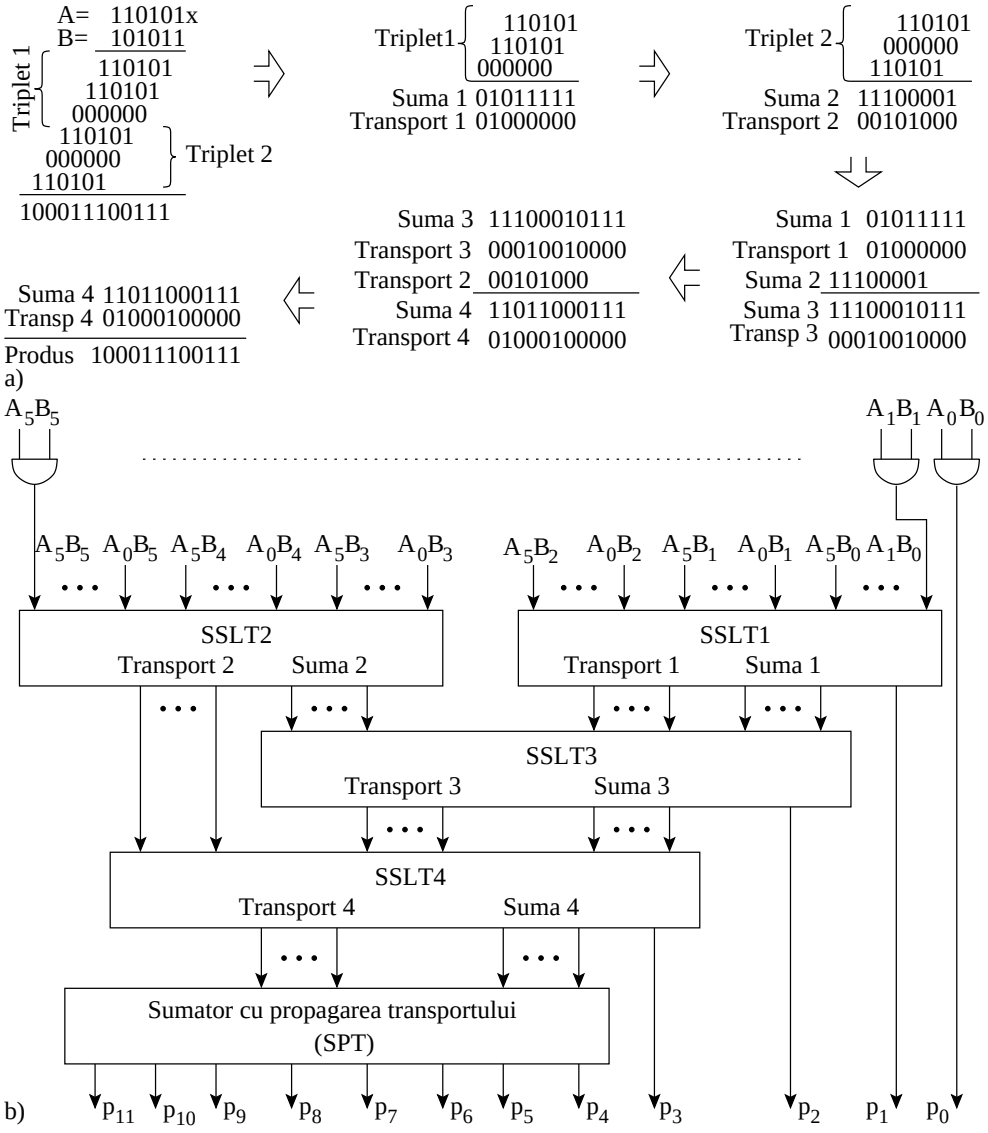


Figura 2.67 Multiplicatorul arbore tip Wallace pentru $n = 5$**:** a) modalitatea de formare a tripleţilor pentru fiecare nivel al arborelui; b) structurarea arborelui pe bază de sumatoare su salvarea transportului, SSLT, şi un sumator cu propagarea transportului, SPT.

transportului, SPT, de orice tip prezentat în secţiunea 2.5.2.

În cazul general, pentru un multiplicator Wallace de n biţi cele n produse parţiale, fiecare de n biţi, se grupează în k_0 tripleţi, $n = 3k_0 + l_0$ unde $0 \leq l_0 \leq 2$, se aplică primului nivel compus din k_0 sumatoare cu salvarea transportului şi se obţin k_0 perechi sumă şi transport. Pentru al doilea nivel de sumatoare cu salvarea transportului se grupează în continuare în k_1 tripleţi conform relaţiei $2k_0+l_0 = 3k_1+l_1$ unde $0 \leq l_1 \leq 2$ şi se generează k_1 perechi sumă şi transport. Acest proces se continua în compresii consecutive de 3:2 pe un număr de $\lceil log_{3/2}n \rceil$ nivele până când se obţine o singură pereche sumă şi transport care sunt sumate în final pe un sumator cu propagarea transportului. Deci timpul minim de multiplicare T_m este egal cu întârzierea pe o poartă AND plus timpul de propagare prin $\lceil \log_{3/2}n \rceil$ niveluri de SSLT la care se adaugă timpul de sumare T_Σ, pe sumatorul cu propagarea transportului. Dimensiunea multiplicatorului este în $O(n^2)$.

Multiplicatorul Wallace are performanţă de viteză mai bună decât multiplicatorul matriceal cu salvarea transportului dar, totuşi, din cauza neregularităţii layout-ului, ultimul este preferat în implementările VLSI.

Ideea de structurare a multiplicatorului sub formă de arbore a generat celule cu compresia diferită de 3:2. Pentru o compresie 2:1 rezultă o structurare de arbore binar. Alte organizări pe bază de celule cu compresia $(5:3), (7:3)$ şi $(15:4)$ reduc adâncimea arborelui dar nu neapărat reduc şi timpul de multiplicare deoarece odată cu creşterea raportului de compresie creşte şi întârzierea pe celulă [Omandi ′94][Petterson ′96][Smith ′97].

2.5.3.3 Multiplicatorul tabelar

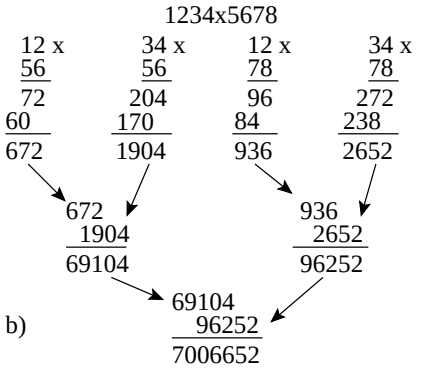
După cum reiese şi din denumire, multiplicatorul tabelar este o tabelă (LUT) care conţine toate produsele între două cuvinte de lungime de n biţi. Teoretic, acest tip de multiplicator ar trebui să fie cel mai simplu şi mai rapid. Tabelul de adevăr pentru produsul a două cuvinte, A_1A_0 şi B_1B_0 de doi biţi, este prezentat în Figura 2.68-a. Se pot deduce expresiile logice pentru fiecare din cei patru biţi $p_3p_2p_1p_0$ ai cuvântului produs:

$$\begin{aligned}
p_0 &= \bar{A}_1A_0\bar{B}_1B_0 + \bar{A}_1A_0B_1B_0 + A_1A_0\bar{B}_1B_0 + A_1A_0B_1B_0 = A_0B_0\\
p_1 &= \bar{A}_1A_0B_1\bar{B}_0 + \bar{A}_1A_0B_1B_0 + A_1\bar{A}_0\bar{B}_1B_0 + A_1\bar{A}_0B_1B_0 + A_1A_0\bar{B}_1B_0\\
&\quad + A_1A_0B_1\bar{B}_0 = \bar{A}_1A_0B_1 + A_1A_0\bar{B}_0 + A_1\bar{A}_0B_0 + A_1\bar{B}_1B_0\\
p_2 &= A_1\bar{A}_0B_1\bar{B}_0 + A_1\bar{A}_0B_1B_0 + A_1A_0B_1\bar{B}_0 = A_1\bar{A}_0B_1 + A_1B_1\bar{B}_0\\
p_3 &= A_1A_0B_1B_0.
\end{aligned}$$

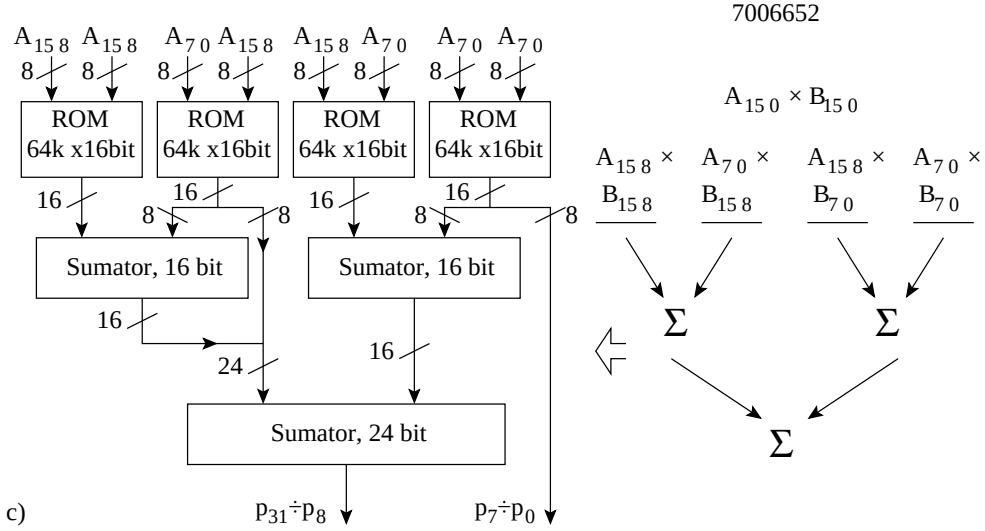
care pot fi implemetate cu porţi logice, cu DCD + OR sau cu un circuit ROM. Implementarea cea mai recomandată pentru astfel de tabele este cu circuite ROM; cele două cuvinte de n biţi formează, prin alăturare, adresa de $2n$ biţi a locaţiei unde este stocat cuvântul produs, al celor două cuvinte, cu lungimea de $2n$ biţi. Limitarea, fie de capacitata de adresare, fie de lungime de cuvânt stocat, impusă de un anumit circuit ROM la implementarea unei tabele de înmulţire, poate fi depăşită prin segmentarea lungimii operanzilor. În acest sens, în Figura 2.68-b, este prezentată înmulţirea segmentată a operanzilor 1234×5678. Fiecare operand s-a segmentat în două numere 12 cu 34 şi 56 cu 78 şi s-au efectuat înmulţirile fiecare cu fiecare, apoi s-au sumat produsele segmentate obţinute pe două niveluri de sumator, evident la

Intrari (Adresa) A_1A_0	B_1B_0	Iesiri $p_3p_2p_1p_0$	Intrari (Adresa) A_1A_0	B_1B_0	Iesiri $p_3p_2p_1p_0$
0 0	0 0	0 0 0 0	1 0	0 0	0 0 0 0
0 0	0 1	0 0 0 0	1 0	0 1	0 0 1 0
0 0	1 0	0 0 0 0	1 0	1 0	0 1 0 0
0 0	1 1	0 0 0 0	1 0	1 1	0 1 1 0
0 1	0 0	0 0 0 0	1 1	0 0	0 0 0 0
0 1	0 1	0 0 0 1	1 1	0 1	0 0 1 1
0 1	1 0	0 0 1 0	1 1	1 0	0 1 1 0
0 1	1 1	0 0 1 1	1 1	1 1	1 1 1 1

a)



b)



c)

Figura 2.68 Multiplicatorul tabelar: a) tabelul de adevăr pentru înmulţirea a două cuvinte de doi biţi, $A_1A_0 \times B_1B_0$; b) modalităţi de substituire a înmulţirii numerelor cu multe cifre, prin segmentare, cu mai multe înmuţiri de numere cu cifre mai puţine; c) structurarea circuitului pentru înmulţirea a două cuvinte de 16 biţi, prin segmentare, pe bază de circuite ROM şi sumatoare.

sumare s-a ţinut cont de deplasările necesare. Pentru segmentarea a două numere binare cu lungimea de 16 biţi $A = A_{15}...A_1A_0$ în A_{15-8} cu A_{7-0} şi $B_{15}...B_1B_0$ în B_{15-8} cu B_{7-0}, în Figura 2.68-c se prezintă o variantă posibilă pentru implementarea tabelelor de înmulţire pe circuite ROM cu capacitatea de adresare de 64K. Tabelul înmulţirii a două cuvinte binare de 8 biţi începe într-o capacitate de $2^{16} \times (2 \times 8)$ biţi, adică două circuite ROM de 64x8biţi conectate în paralel, în total sunt necesare 8 astfel de circuite ROM.

Timpul de multiplicare T_m se compune din timpul de acces la memorie τ_{AA} plus întârzierea pe calea cea mai lungă determinată de nivelurile de sumatoare. Dimensiunea multiplicatorului este dimensiunea/capacitatea memoriei $2^{2n} \times 2n$ deci în $O(n \times 2^{2n})$ plus cea a sumatoarelor utilizate. Această valoare ridicată a dimensiunii limitează încă implementarea multiplicatorului tabelar. Probabil, cu creşterea densităţii de integrare simultan cu reducerea costului şi acest tip de multiplicator poate deveni atractiv.

O modalitate care poate duce la micşorarea capacităţii ROM necesară pentru tabele constă în efectuarea indirect a multiplicării prin logaritmare şi antilogaritmare conform relaţiei:

$$A \cdot B = antilog(\log A + \log B)$$

Pentru această implementare sunt necesare două tabele (una pentru stocarea logaritmilor şi una pentru antilogaritmi) şi un sumator. Aceste tabele au evident mai puţine intrări decât o tabelă care realizează direct înmulţirea ($2 \times n$ intrări) iar lungimea cuvântului stocat în aceste tabele depinde de nivelul de eroare care se acceptă în rezultatul înmulţirii (indicat pentru înmulţiri de numere reprezentate în virgulă flotantă, unde produsul obţinut de $2n$ biţi se reduce doar la n biţi). Timpul de multiplicare este determinat de accesul la cele două memorii (tabele) plus cel necesar sumării.

2.5.4 Circuite de deplasare

Circuitele de deplasare (shift-are) transferă fiecare bit de intrare x_i, $i = 0, 1, 2, \ldots, n-1$, al unui cuvânt de n biţi, într-un bit de aceeaşi valoare în cuvântul de ieşire dar deplasat faţa de poziţia i cu $\pm D$ poziţii, conform relaţiei:

$$y_{i \pm D} = x_i$$

Mai exact, acest proces de deplasare al cuvântului de intrare X, într-un cuvânt de ieşire Y este caracterizat de următoarele mărimi:

- direcţia de deplasare (stânga/dreapta);
- distanţa de deplasare D (exprimată în binar prin numărul de poziţii);
- condiţiile de capăt.

Prin condiţiii de capăt se înţelege modul cum se procedează cu biţii care ies din intervalul de index al cuvântului $[n-1, n-2, \ldots, 1, 0]$, respectiv cu ce se completează poziţiile rămase libere la celălalt capăt al cuvântului.

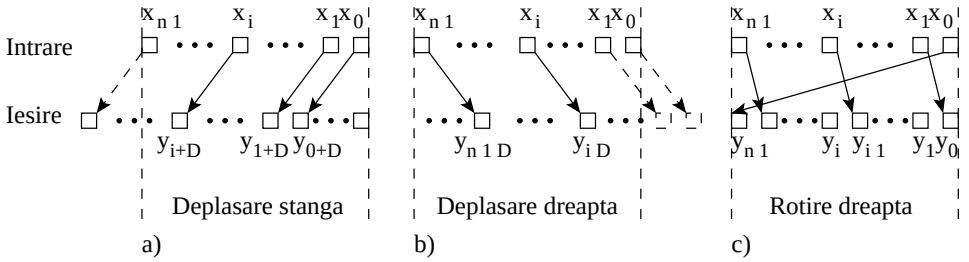


Figura 2.69 Modalităţi de realizare a operaţiei de deplasare/rotire: a,b) deplasare stânga/dreapta; c) rotire dreapta.

Cuvântul căruia i se aplică deplasarea poate avea semnificaţia de număr (în care fiecare poziţie corespunde unui rang în exprimarea ponderată în baza 2 a valorii numărului), sau are semnificaţia doar a unui şir de biţi (când poziţia nu determină o pondere binară). Corespunzător acestor două semnificaţii, care le poate avea cuvântul deplasat, rezultă respectiv o deplasare aritmetică sau o deplasare logică.

Printr-o **deplasare aritmetică** cu D poziţii (spre stânga) numărul respectiv este înmulţit cu 2^D iar printr-o deplasare aritmetică cu $-D$ poziţii (dreapta) numărul este împărţit cu 2^D. Dacă numărul reprezentat binar este fără semn atunci atât la deplasarea stânga, Figura 2.69-a, cât şi deplasarea dreapta, Figura 2.69-b, biţii care ies din intervalul poziţiilor $[n-1, n-2, \ldots, 1, 0]$ se pierd în numărul reprezentat la ieşirea circuitului de deplsare iar poziţiile rămase goale în acest interval se completează cu zero. La **deplasarea logică** se procedează la fel dar cuvântul supus deplasării stânga/dreapta nu mai este interpretat ca un număr ci doar ca un şir de biţi.

La deplasarea numerelor cu semn, care au bitul de semn în poziţia $(n-1)$, acesta trebuie păstrat (**extensia de semn**) indiferent că numărul este multiplicat cu 2^{-D} sau 2^D. De exemplu, numărul $20|_{10} = 10100|_2$ iar -20 în complement de doi se obţine $10100 \rightarrow 01011 + 1 \rightarrow 01100 \rightarrow 1\ 01100 = [20]_2$. La deplasarea spre dreapta a numărului $[20]_2$ cu două poziţii (împărţire cu 2^2) trebuie completat bitul de semn în cele două poziţii din stânga, se obţine: $111011 \rightarrow (-1)2^5+1\cdot2^4+1\cdot2^3+0\cdot2^2+1\cdot2^1+1\cdot2^0 = -5$. La deplasarea spre stânga cu o poziţie (inmulţire cu 2), bitul de semn nu iese în afara intervalului de şase biţi (semnul trebuie păstrat), intervalul trebuie extins la şapte biţi, se obţine: $1011000 \rightarrow (-1)2^6+0\cdot2^5+1\cdot2^4+1\cdot2^3+0\cdot2^2+0\cdot2^1+0\cdot2^0 = -40$.

Există şi cazul particular de deplasare stânga/dreapta când poziţiile $(n-1)$ şi 0 se consideră vecine, circuitul de deplasare în acest caz este referit ca **circuit rotitor**, Figura 2.69-c. Prin rotire, deoarece biţii care ies din intervalul de index la un capăt sunt introduşi la celălalt capăt, nu rămân poziţii libere deci nu se fac completări cu zero. Exemplificăm rotirea cu şirul format din primele 8 litere din alfabet (s-au substituit biţii prin litere) $ABCDEFGH$ care printr-o rotire la dreapta cu şase poziţii ($D = -6$) se obţine $CDEFGHAB$ iar printr-o rotire la stânga cu două poziţii ($D = 2 = 8 - 6$) se obţine $CDEFGHAB$, adică acelaşi cuvânt în ambele cazuri. Lungimea cuvântului de n biţi, în general, este un număr putere a lui 2, deci $n - D$ este complementul faţă de doi $[D]_2$ al lui D. Din această observaţie rezultă că o structură de rotitor poate realiza atât rotirea spre stânga cât şi rotirea spre dreapta, deoarece o rotire cu $-D$ (dreapta) este echivalentă cu o rotire spre stânga exprimată de un număr de poziţii egal cu $[D]_2$. În exemplul dat, comanda de deplasare dreapta

cu şase poziţii $-6 = -110|_2$ se substituie cu comanda spre stânga cu două poziţii spre stânga $[6]_2 = 8 - 6 = 1000 - 110 = 010 = 2$.

Analizând pe Figura 2.69, în ce constă operaţia de deplasare, se constată că este o selectare a unui bit de intrare şi transferul acestuia într-o altă poziţie la ieşire fără a-i modifica valoarea. Ori, această operaţie de selectare, de la intrare la ieşire, este proprie circuitului multiplexor. Dacă se extinde aceasta de la bit la nivel de cuvânt se constată că operaţia de deplasare poate fi realizată cu o structură de selectare secvenţială a datelor cu multiplexoare ca în Figura 2.36-b. Un circuit de deplasare, pentru un cuvânt de n biţi, structurat ca un circuit de selectare a datelor, pe bază de multiplexoare, este format din n multiplexoare în paralel, fiecare multiplexor cu un număr de intrări de date $\geq n$. Fiecare din cele n cuvinte obţinut, de exemplu, prin deplasarea stânga cu D poziţii ($D \in [n-1, 0]$) este stocat într-un port iar biţii acestui port sunt aplicaţi la intrarea cu acelaşi număr a tuturor celor n multiplexoare în paralel. La aplicarea cuvântului de selectare (deplasare) D, comun pentru toate intrările de selectare ale multiplexoarelor, este selectat portul de intrare în care este înscris cuvântul deja deplasat cu D poziţii spre stânga şi aplicat la portul de ieşire.

Structurarea anterioară a circuitului de deplasare, ca selector de cuvinte (înscrise deja în configuraţii deplasate) din n porturi de intrare, poate fi restrânsă la un singur port de intrare ca în Figura 2.70-a, unde portul de intrare este o magistrală de patru biţi pe care se aplică cuvântul de deplasat $X = x_3x_2x_1x_0$. La ieşirea circuitului de deplasare stânga de patru biţi pentru deplasările de $0, 1, 2, 3$ poziţii se obţin respectiv următoarele cuvinte $x_3x_2x_1x_0$, $x_2x_1x_0$0, $x_1x_0$00 şi $x_0$000. La o astfel de structurare, cu un singur port de intrare, bitul j (linia j a magistralei) nu se mai aplică doar la intrarea j de date a unui singur multiplexor, ca pentru structurarea anterioară, ci se aplică la fiecare din multiplexoare dar pe intrări de date diferite (a se vedea, de exemplu, linia de magistrală x_0). La deplasarea spre stânga a cuvântului de patru biţi în poziţiile rămase libere, unde în cazul general ar intra biţii din poziţiile $-1, -2, -3, \ldots$ dar care în cazul acesta nu există, se introduc zero-uri prin conectarea la masă a intrărilor corespunzătoare ale multiplexoarelor.

Varianta de circuit rotitor de patru biţi este reprezentată în Figura 2.70-b. Aici, se observă clar că fiecare bit de intrare, linie de magistrală, se aplică la fiecare multiplexor dar pe intrări diferite. La fel ca şi la circuitul de deplasare, scriind cele patru cuvinte care se pot obţine la ieşire $x_3x_2x_1x_0$, $x_2x_1x_0x_3$, $x_1x_0x_3x_2$ şi $x_0x_3x_2x_1$ se deduce foarte uşor la care linie de magistrală se conectează fiecare intrare a multiplexoarelor.

Caracteristicile de adâncime şi dimensiune ale circuitului de deplasare dreapta/ stânga, CDD/S, sunt determinate de cele n multiplexoare în felul următor:

$$D_{CDD/S}(n) \in O(1)$$
$$S_{CDD/S}(n) = n \times S_{MUX(n)} \in O(n^2 log\ n)$$

rezultă şi produsul dimensiune-adâncime:

$$S_{CDD/S}(n) \times D_{CDD/S}(n) \in O(n^2 log\ n)$$

($S_{MUX(n)} \in O(n \cdot 2^n)$) această relaţie dedusă în secţiunea 2.4.4 corespunde cazului când n este numărul de intrări de selectare la MUX2^n:1, dar în cazul CDD/S n este numărul de intrări de date la MUXn:1 deci relaţia pentru dimensiune devine $S_{MUX(n)} = O(n \cdot \log n)$).

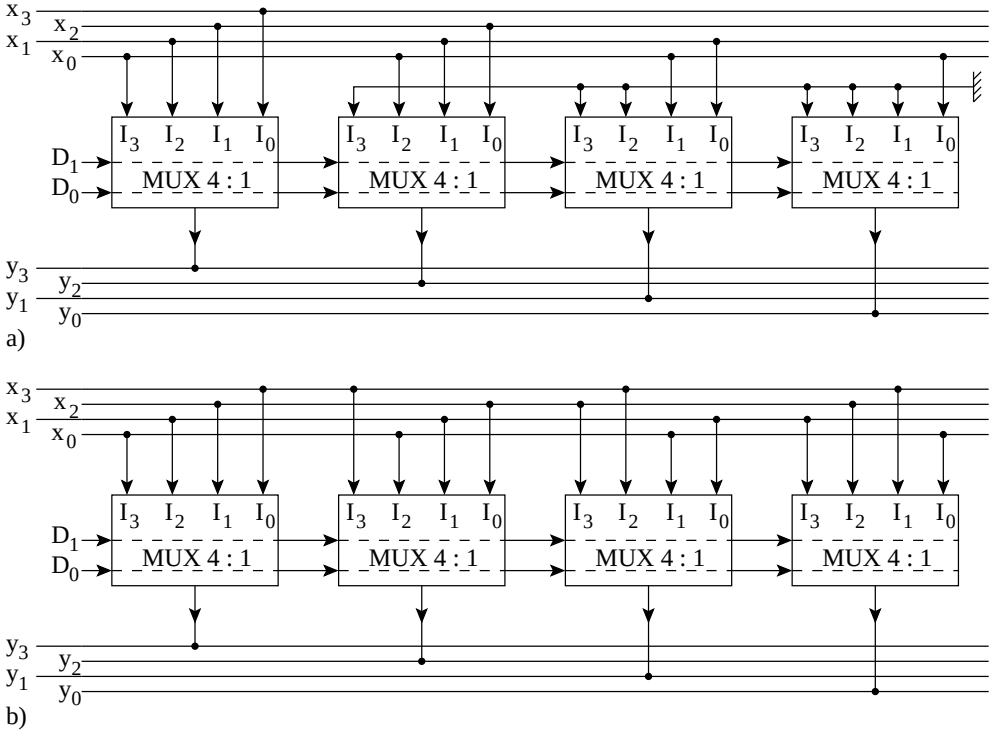


Figura 2.70 Exemplu de structurare a circuitelor de deplasare/rotire pe bază de multiplexoare: a) circuit de deplasare stânga pentru cuvinte de patru biţi; b) circuit rotitor de patru biţi.

Dimensiunea mare pe care o implică astfel de structurare a circuitului de deplasare (cea a circuitului rotor este la fel) poate ridica dificultăţi de implementare. În plus, adâncimea atractivă în $O(1)$ poate fi mult înrăutăţită de fan-out-ul ridicat. Fiecare bit al cuvântului de intrare comandă n intrări de date ale multiplexoarelor şi la fel, fiecare intrare de selectare se aplică la n multiplexoare.

Cauza acestor neatrăgătoare performanţe ale circuitului de deplasare cu structurarea anterioară rezidă în realizarea pe un singur nivel, format din $n\times$MUXn:1, a tuturor celor n deplasări comandabile prin cuvântul $D = D_{k-1}D_{k-2}...D_1D_0$, unde $k = \lceil \log_2 n \rceil$. Dar valoarea numărului de poziţii de deplasat rezultă, ca la oricare număr binar, printr-o sumă ponderată după puterile lui 2 din cuvântul de control D. De exemplu, pentru un cuvânt de deplasat cu lungimea de 8 biţi o comandă de deplasare cu 6 poziţii $D = D_2D_1D_0 = 110$, $6 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$ poate fi realzată prin două deplasări succesive întâi o deplasare de două poziţii ($D_1 = 1$) pe un nivel de multiplexoare, apoi cuvântul rezultat se aplică pe următorul nivel de multiplexoare unde se realizează o deplasare de patru poziţii ($D_2 = 1$), deci la ieşire se obţine o deplasare cu 6 poziţii. În plus, un nivel numai cu o singură valoare de deplasare, din cele n posibile se poate realiza numai cu multiplexoare $n\times$MUX2:1, iar aplicarea biţilor cuvântului de intrare la intrările multiplexoarelor MUX2:1 se face după o regulă foarte simplă. Pentru nivelul format din $n\times$MUX2:1, comandat de bitul de control

D_2 (unde se realizează deplasarea cu 4 poziţii, $D_2 \cdot 2^2$), bitul i din cuvântul de intrare se aplică la intrarea de date 0 a multiplexorului i precum şi la intrarea 1 de date a multiplexorului $i+4$ dacă deplasarea este stânga, ori la intrarea 1 de la multiplexorul $i-4$ dacă deplasarea este la dreapta. La acest nivel de deplasare comandat de D_2, pentru $D_2 = 0$ cuvântul de intrare va fi obţinut la ieşire fără deplasare (deplasare zero, $0 \cdot 2^2$), iar pentru $D_2 = 1$ la ieşire se obţine cuvântul de intrare deplasat cu patru poziţii, $1 \cdot 2^2 = 4$. O structurare pe trei niveluri succesive de câte 8 $MUX2:1$ pentru un cuvânt de 8 biţi este prezentată în Figura 2.71-a. Pentru un cuvânt de control $D = 101$ primul nivel va efectua o deplasare cu $D_0 \cdot 2^0 = 1$ poziţie, al doilea nivel o deplasare cu $D_1 \cdot 2^1 = 0$ poziţii, iar ultimul nivel o deplasare cu $D_2 \cdot 2^2 = 4$ poziţii. În total 1+0+4=5 poziţii.

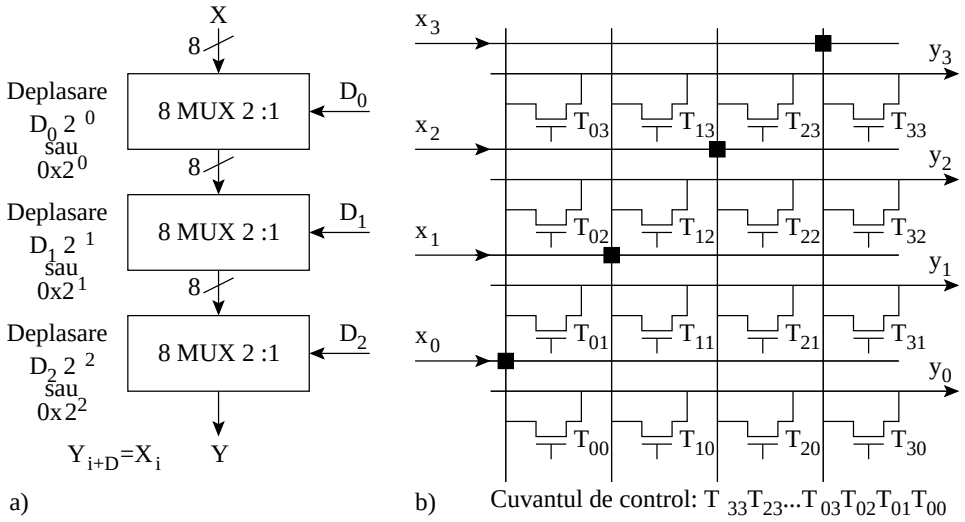


Figura 2.71 Structuri de circuite de deplasare/rotire: a) pe bază de niveluri succesive de nMUX2:1; b) pe bază de matrice de comutaţie.

Adâncimea circuitului de deplasare $D'_{CDD/D}(n)$ cu niveluri succesive este egală cu numărul de niveluri de multiplexoare $\log_2 n$, $D'_{CDD/S}(n) \in O(\log n)$ şi este mai mare decât la structurarea anterioară unde era constantă şi egală cu $D_{CDD/S}(n) = 4$, iar fan-out-ul unui semnal de bit, ce se aplică la intrarea unui nivel format din nMUX2:1, este egal cu doi indiferent de lungimea cuvântului de deplasat, în schimb, fan-out-ul unui bit D_i, din cuvântul de control, rămâne tot n. Dimensiunea circuitului este:

$$S'_{CDD/S}(n) = log_2 n \cdot n \cdot S_{MUX2:1} = 4 \cdot n \cdot log_2 n \in O(n \cdot log\ n)$$

rezultând produsul dimensiune-adâncime în $O(n \cdot \log^2 n)$

$$S'_{CDD/S}(n) \cdot D'_{CDD/S}(n) = 4 \cdot n \cdot log_2 n \cdot log_2 n \in O(n \cdot log^2 n)$$

produs care este mai mic decăt $O(n^2 \log n)$ al variantei anterioare de structurare pe un singur nivel a circuitului de deplasare:

$$S_{CDD/S}(n) \cdot D_{CDD/S}(n) > S'_{CDD/D}(n) \cdot D'_{CDD/S}(n)$$

Această relaţie admite aceeaşi interpretare, care s-a dat la analiza relaţiei 2.8 în ceea ce priveşte corelarea între adâncime şi dimensiune când se caută un circuit de viteză mai ridicată (sporul de viteză este mai mic decât creşterea în dimensiune în circuistică).

Similar, se poate structura şi un circuit rotitor din niveluri succesive de multiplexoare, fiecare nivel următor realizând rotaţii crescătoare după puterile lui doi, caracteristicile de adâncime şi dimensiunea sunt aceleaşi cu cele deduse anterior.

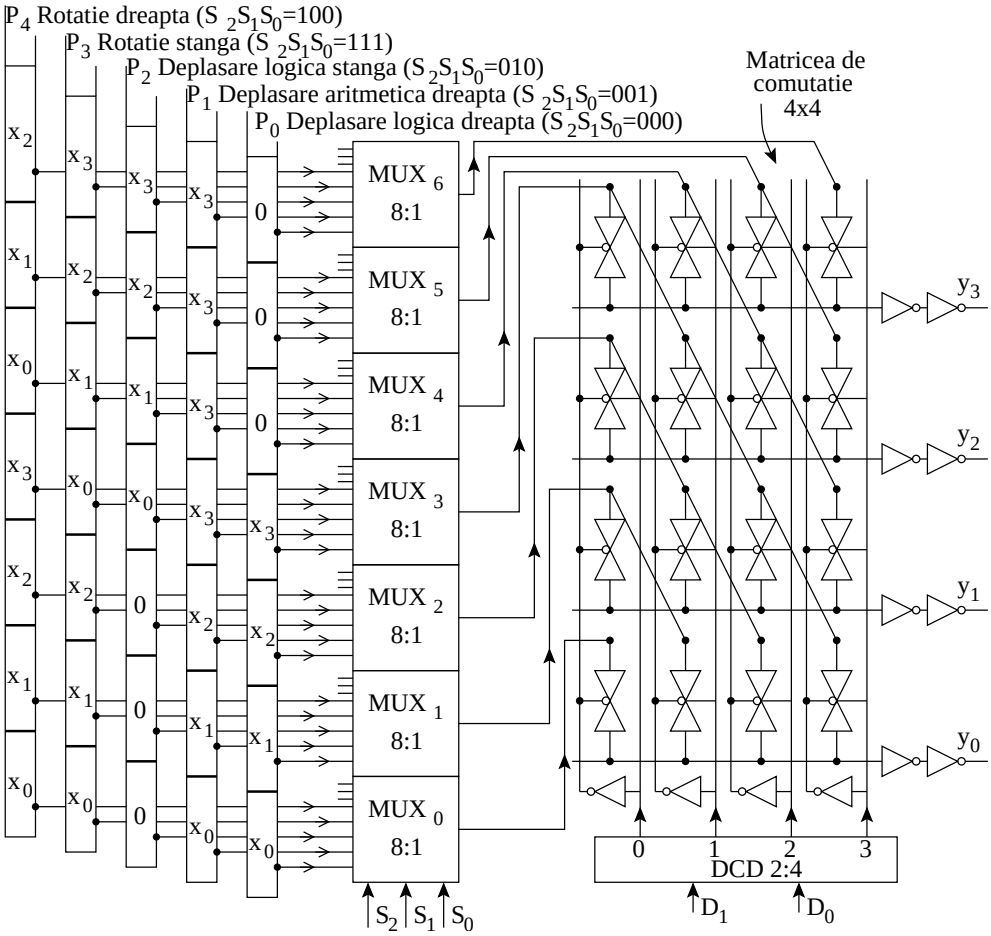


Figura 2.72 Structurarea unui circuit shifter de 4 biţi pe baza unei matrice de comutaţie 4×4 **şi a unui sistem de selectare secvenţială 7×MUX8:1**

Circuitul general care poate implementa cele mai multe procesări asupra unui şir de biţi (deplasare logică dreapta/stânga, rotaţie stânga/dreapta, deplasarea aritmetică stânga/dreapta, inversarea ordinii, amestecare, extragere de biţi etc.) este **matricea de comutaţie**, Figura 2.71-b. În fiecare nod ij al unei matrice de comutaţie $n \times n$ există un element de comutaţie care conectează coloana i la linia j, deci fiecare coloană i (linia de intrare x_i) poate fi conectată la oricare linie de ieşire y_j. Ca element de comutaţie poate fi utilizată poarta CMOS de transmisie sau un tranzistor de trecere (uzual nMOS). Dezavantajul principal pentru o astfel de matrice de comutaţie este

numărul mare, n^2, de conexiuni individuale de comandă care trebuie aplicate la elementele de comutaţie din cele n^2 noduri (un cuvânt de control cu lungimea de n^2 biţi).

În cazurile particulare, când se implementează pe matricea de comutaţie doar un singur tip de procesare (o singură operaţie), numărul de comenzi individuale se restrânge. Pentru o matrice 32×32 utilizată, de exemplu, doar pentru rotire stânga vor fi necesare numai 32 de comenzi individuale corespunzătoare distanţelor de deplasare cu $0, 1, 2, ..., 30, 31$ poziţii. Pentru o rotire stânga cu D poziţii vor fi comandate elementele de comutaţie din n noduri astfel alese încât un bit de pe coloana i să fie mapat pe linia $j = i + D$; rezultă că pentru toate tranzistoarele din nodurile $i, i + D,\ i = 0, 1, ..., 31$ porţile vor fi cablate împreună şi comandate cu semnalul obţinut la ieşirea unui decodificator când pe intrarea acestuia se aplică mumărul D ($D = 0, 1, ..., 30, 31$) în binar. Restrângerea doar la un singur tip de operaţie pe matricea de comutaţie reduce numărul de comenzi la n, dar semnalul cablat pentru fiecare dintre aceste comenzi trebuie să asigure un fan-out egal cu n.

Totuşi, există posibilitatea ca pe o matrice de comutaţie, deşi cablată doar pentru o anumită operaţie, să poată fi realizate şi alte operaţii prin modificare deja în exterior a cuvântului care se aplică la intrare, ca în Figura 2.72, un astfel de circuit este referit ca **shifter (Barrel Shifter)**. Acest circuit shifter conţine o matrice de comutaţie compusă din 4×4 porţi CMOS de transmisie care este cablată doar pentru realizarea deplasărilor dreapta cu $0, 1, 2, 3$ poziţii, prin valorile cuvântului de control al deplasării, D_1D_0, egale respectiv cu: $00, 01, 10, 11$ (aplicate la intrarea unui DCD2:4). Dar, la intrarea matricei se aplică pentru fiecare tip de operaţie, realizată de shitfter, un cuvânt obţinut deja prin modificarea cuvântului de intrare $x_3x_2x_1x_0$. Fiecare cuvânt, modificat şi înscris în unul din porturile P_0, P_1, P_2, P_3, P_4, printr-un sistem de selectare compus din 7×MUX8:1, este selectat prin cuvântul $S_2S_1S_0$ şi aplicat la intrarea matricei de comutaţie. Deci, cuvintele de control $S_2S_1S_0$ şi D_1D_0 determină, primul, tipul de operaţie, iar al doilea, numărul de poziţii de deplasare. Acest shifter poate realiza următoarele operaţii: deplasare logică dreapta/ stânga, rotaţie dreapta/stânga şi deplasare aritmetică dreapta. Deoarece la fiecare MUX8:1 mai există trei intrări de date neutilizate se mai pot selecta încă trei porturi (nedesenate în figură) în care se poate introduce cuvântul $x_3x_2x_1x_0$ modificat pentru implementarea a încă trei operaţii.

2.5.5 Unitatea Aritmetică şi Logică, ALU

2.5.5.1 Calea de date

Sistemele digitale de calcul, atât din punct de vedere al proiectării şi realizării lor cât şi din punct de vedere al tratării didactice, pot fi organizate/compuse din două părţi, denumite foarte general, **calea de date şi calea de control** (vezi Figura 3.14). În calea de date se realizează operaţii de tip logic sau aritmetic asupra cuvintelor binare (operanzi), iar calea de control selectează operaţia care se efectuează în calea de date şi pentru operaţia selectată se comandă etapele de realizare.

Organizarea unei posibile căi de date este prezentată în Figura 2.73-a. Această cale de date conţine, pentru realizarea conexiunilor în vederea transferurilor, trei magistrale, două pentru operanzii sursă A, B şi una pentru operandul rezultat R. Operanzii sursă şi operandul rezultat (destinaţie) sunt stocaţi într-un bloc de registre (echiva-

lentul fizic al noţiunii de port). Selectarea unui port (operand) pentru înscrierea conţinutului său pe o magistrală sursă se realizează cu un cuvânt de selectare S_A sau S_B aplicat unui grup de multiplexoare, care compun un bloc de selectare a datelor cu o structură ca în Figura 2.36-b. În această cale de date fiecare bloc de selectare este compus din $n\times$MUX16:1, deoarece se consideră că blocul de registre conţine 16 porturi cu lungimea de n biţi, iar selectarea acestora se realizează prin cuvintele de selectare de patru biţi $S_A = S_{A3}S_{A2}S_{A1}S_{A0}$, $S_B = S_{B3}S_{B2}S_{B1}S_{B0}$. Înscrierea operandului rezultat, de pe magistrala rezultat R, într-un port destinaţie se realizează prin intermediul unui bloc secvenţial de distribuţie selectat prin cuvântul de selectare $S_D = S_{D3}S_{D2}S_{D1}S_{D0}$ (în acest caz blocul de distribuţie este constituit din $n\times$DMUX1:16).

Într-o cale de date este absolut necesară o **unitate logico-aritmetică, ALU** (**A**rithmetic **L**ogic **U**nit) şi, uneori, aceasta este inseriată cu un circuit de deplasare (**barrel shifter**). Unitatea logico-aritmetică efectuează o operaţie asupra celor doi operanzi A, B aplicaţi, prin cele două magistrale sursă, la intrările sale şi generează operandul rezultat O care, înainte de a fi aplicat pe magistrala rezultat R, poate fi modificat în circuitul de deplasare prin: deplasări aritmetice/logice, stânga dreapta, rotiri, extrageri de biţi sau gupuri de biţi. Valoarea de deplasare în shifter este prescrisă prin cuvântul de control D care în acest caz are 5 biţi D_4, D_3, D_2, D_1, D_0, deoarece se consideră lungimea de cuvânt procesat este egală cu $n = 32$.

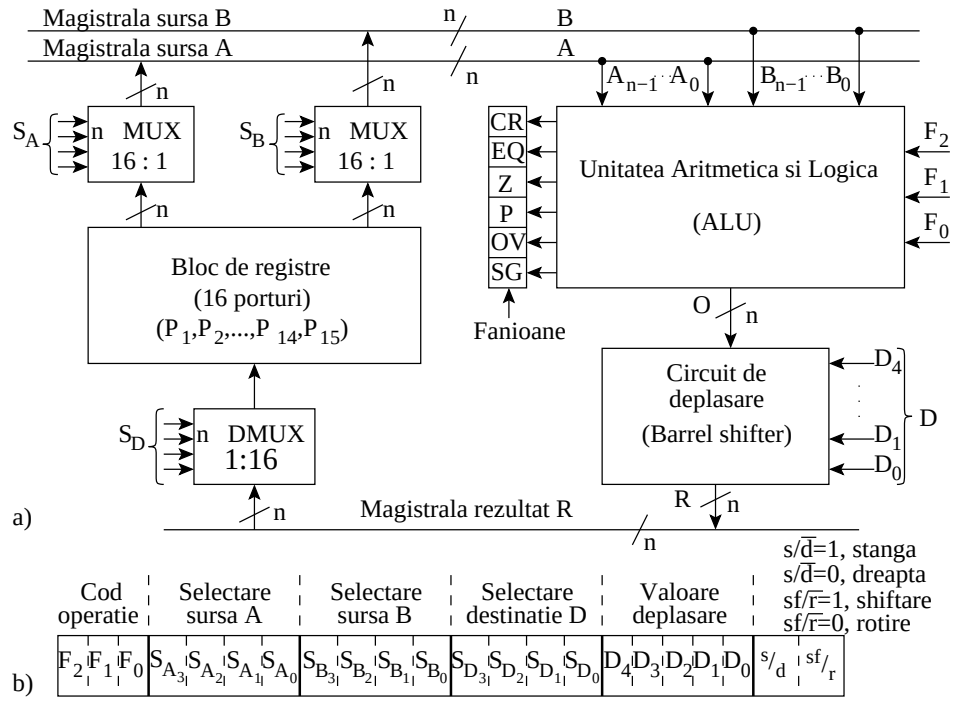


Figura 2.73 Calea de date: a) organizare posibilă a unei căi de date cu trei magistrale; b) formatul binar al cuvântului de comandă a unei operaţii în calea de date (instrucţiune în cod maşină).

Proiectarea unei ALU porneşte de la nivelul arhitectural al acesteia, adică de la ceea ce "vede" un utilizator/programator. Se "vede" interfaţarea/conectarea cu celelalte elemente din calea de date prin intrări şi ieşiri. Intrări la calea de date sunt cele două cuvinte de pe magistrala A şi B şi biţii de control pentru operaţia de efectuat. Numărul operaţiilor pe care le poate realiza o unitate logico-aritmetică este de la câteva până la zeci; în organizarea din figură se pot realiza doar opt operaţii deoarece cuvântul de control are numai 3 biţi F_2, F_1, F_0.

Ieşirile din ALU sunt: cuvântul rezultat O cu lungimea de n biţi şi semnalele **indicatorii de condiţii** (care înscriu anumite fanioane/flag-uri, biţii de condiţii). Indicatorii de condiţii specifică anumite relaţii între cei doi operanzi sau anumite caracteristici valorice ale operandului rezultat. Se poate introduce un indicator pentru oricare condiţie dacă este necesară a fi utilizată în evaluarea corectitudinii operaţiei efectuate sau pentru condiţionarea operaţiilor următoare. Ca exemplificare, enumerăm şase din biţii de condiţii cei mai des utilizaţi:

CARRY(CR) este bitul care indică depăşirea de capacitate la reprezentarea numerelor întregi pozitive. Într-un sistem de calcul există o lungime maximă n a unui cuvânt cu care se poate opera. Pentru o lungime mai mare nu există suportul fizic de a fi reprezentată. Rezultă că se poate opera numai cu numere care pot fi reprezentate cu cei n biţi. O operaţie sau un număr care necesită mai mult decât n biţi generează o depăşire de capacitate. Bitul CR are valoarea transportului următor C_{n-1} din cuvântul de ieşire, adică de la rangul $n-1$ la rangul n, de la poziţia n la $n+1$ (care nu există).

EQUAL(EQ) este bitul care indică identitatea celor două cuvinte A şi B. Valoarea sa se generează prin porţi NXOR, $\overline{A_i \oplus B_i}$, asupra fiecărei pereche de biţi care apoi se colectează într-o poartă AND.

ZERO(Z) este bitul care indică faptul că rezultatul operaţiei este un cuvânt compus din n zerouri. Valoarea sa se generează prin colectarea tuturor biţilor cuvântului rezultat într-o poartă OR.

PARITY(P) este bitul care indică paritatea sau imparitatea cuvântului rezultat dacă este interpretat ca număr. Valoarea sa este identică cu valoarea bitului de rang zero al cuvântului O.

OVERFLOW(OV) este bitul de depăşire de capacitate la reprezentare numerelor cu semn. În reprezentarea numerelor cu semn (în complement faţă de 1, în complement faţă de 2, în mărime şi semn) bitul de semn (bitul cu indice $n-1$) are valoarea 1 pentru un număr negativ şi valoarea 0 pentru un număr pozitiv. Depăşirea de capacitate, C_{n-2}, apare ca un transport de la bitul cel mai semnificativ al numărului (bitul cu indicele $n-2$) la bitul de semn şi dacă bitul de semn indică o operaţie eronată (de exemplu adunarea a două numere pozitive generează un număr negativ). Un algoritm simplu, care determină existenţa depăşirii la numere cu semn reprezentate în complement faţă de doi, este: neidentitatea dintre valoarea transportului C_{n-2} de la O_{n-2} la O_{n-1} şi valoarea transportului C_{n-1} generat de la bitul de semn O_{n-1} ($C_{n-1} \neq C_{n-2}$). În consecinţă, valoarea sa se calculează simplu printr-o poartă XOR, $OV = C_{n-1} \oplus C_{n-2}$.

SIGN(SG) este bitul care indică semnul cuvântului rezultat, O, când este interpretat ca număr cu semn, $SG \equiv O_{n-1}$.

Se fixează pentru ALU din figură codurile de selectare şi mnemonicile operaţiilor corespunzătoare în felul următor:

- $F_2F_1F_0 = 000$, **AND** (produsul logic); $O_i = A_i \cdot B_i, i = 0, 1, ..., n-1$
- $F_2F_1F_0 = 001$, **OR** (sumă logică); $O_i = A_i + B_i, i = 0, 1, ..., n-1$
- $F_2F_1F_0 = 010$, **TFM** (trecere fără modificări); $O = A$
- $F_2F_1F_0 = 011$, **DCR** (decrementare); $O = A - 1$
- $F_2F_1F_0 = 100$, **XOR** (sumă modulo 2); $O_i = A_i \oplus B_i, i = 0, 1, ..., n-1$
- $F_2F_1F_0 = 101$, **ADD** (adunare modulo 2^n, operanzii sunt numere întregi); $O = A + B$
- $F_2F_1F_0 = 110$, **SUB** (scădere modulo 2^n, operanzii sunt numere întregi); $O = A - B$
- $F_2F_1F_0 = 111$, **INC** (incrementare); $O = A + 1$

Operanzii sursă sunt referiţi prin numărul de port din care se citesc şi sunt selectăţi prin cuvintele S_A şi S_B, iar operandul rezultat este referit prin numărul portului destinaţie în care se înscrie, selectabil prin cuvântul S_D (cele 16 porturi sunt notate cu $P_1, P_1, ..., P_{14}, P_{15}$).

Comanda căii de date pentru realizarea unie operaţii este descrisă de o instrucţiune care are următoarea formă (în limbaj de asamblare):

$$\text{MNEMONIC OPERATIE} \quad P_D, P_{S1}, P_{S2} \qquad ; P_D \leftarrow (P_{S1} \ \ \text{OPERATIE} \ \ P_{S2})$$

cu următoarea semantica: asupra operanzilor sursă din porturile sursă P_{S1} şi P_{S2} se efectuează operaţia OPERATIE iar rezultatul se înscrie în portul destinaţie P_D. De exemplu, pentru efectuarea operaţiei de adunare între operanzii din porturile P_{13} şi P_7, iar operandul rezultat să fie înscris în portul P_5, se scrie următoarea instrucţiune în limbaj de asamblare:

$$\text{ADD} \quad P_5, P_{13}, P_7 \tag{2.27}$$

Instrucţiunea din limbaj de asamblare este convertită (asamblată), folosind codurile operaţiilor şi codurile cuvintelor de selectare din calea de date, într-un cuvânt binar care comanda efectuarea operaţiei respective în calea de date. Formatul acestui cuvânt reprezentat în Figura 2.73-b este compus din următoarele 7 subcâmpuri:

1. codul operaţiei $F_2F_1F_0$
2. codul portului sursă A: $S_{A3}S_{A2}S_{A1}S_{A0}$
3. codul portului sursă B: $S_{B3}S_{B2}S_{B1}S_{B0}$
4. codul portului destinaţie D : $S_{D3}S_{D2}S_{D1}S_{D0}$

5. valoarea deplasării $D_3D_2D_1D_0$
6. $s/\bar{d}$, deplasarea stânga $s = 1$, deplasarea drepata $\bar{d} = 0$
7. $sf/\bar{r}$, shiftare $sf = 1$, rotaţie $\bar{r} = 0$.

Pentru instrucţiunea anterioară (ADD P_5, P_{13}, P_7),introducând în fiecare câmp codul corespunzător, rezultă următorul cuvânt binar de comandă (instrucţiune în limbaj/cod maşină), scris cu spaţii între subcâmpuri:

101 1101 0111 0101 0000 0 0

Acest cuvânt de comandă, prin biţii săi — fiecare subcâmp de biţi aplicat pentru comandă la elementul corespunzător din calea de date — realizează în calea de date procesarea conţinută în instrucţiunea (dată în limbaj de asamblare) din relaţia 2.27.

2.5.5.2 Organizarea şi implementarea unei unităţi aritmetică şi logică

După definirea arhitecturii ALU, în contextul unei căi de date, se trece la stabilirea organizării interne a acesteia, adică la alegerea acelor părţi componente care pot realiza toate funcţiile definite prin arhitectură. Există diferite părţi componente/blocuri care pot realiza aceeaşi funcţie, ceea ce înseamnă că pot fi mai multe variante de realizare internă, deci pentru aceeaşi arhitectură pot exista mai multe organizări. Elaborarea unei organizări, care să asigure suport pentru toate operaţiile efectuate de ALU poate fi gândită în două variante.

În prima variantă se concepe ALU ca o asamblare de blocuri/circuite specializate, Figura 2.74-a. Fiecare bloc realizează o operaţie din repertoriul ALU asupra cuvintelor de intrare. Blocul circuitelor aritmetice, $n\times$CA, realizează operaţiile de adunare, scădere, incrementare/decrementare primind deja calculate, pentru fiecare pereche de biţi A_i şi B_i, valorile pentru sumă modulo de la blocul $n\times$XOR şi valorile transporturilor anticipate C_{i-1} de la circuitul de generare a transporturilor anticipate, CGTA. Transporturile anticipate sunt calculate pe baza semnalelor intermediare de generare g_i şi propagare produse p_i de blocurile $n\times$AND şi $n\times$OR, care asigură şi operaţiile logice corespunzătoare în ALU.

Aceste blocuri specializate operează simultan dar numai ieşirea unuia este selectată, prin blocul de multiplexoare $n\times$MUX8:1, la ieşirea ALU, prin aplicare codului operaţiei pe intrarea de selectare F_2, F_1, F_0. Valorile biţilor de condiţii se determină uşor, din biţii cuvântului de ieşire, conform definiţiilor date mai sus. De fapt, această organizare poate fi asimilată ca un bloc de selectare secvenţială a datelor pe bază de multiplexoare, Figura 2.36-b. Dimensiunea unei ALU organizată în această modalitate apare ca o sumă a dimensiunilor blocurilor specializate şi a blocului multiplexor de selectare, ultimul având ponderea cea mai ridicată în valoarea dimensiunii.

În a două variantă se concepe ALU ca o punere în paralel a n **unităţi logico-aritmetice elementare, ALUE**. Fiecare ALUE constituie o felie (slice) din ALU care realizează toate operaţiile din repertoriul ALU, dar numai pentru cuvinte de 1 bit. Printr-o optimizare a celulei ALUE, cu această organizare, se poate obţine o dimensiune mai redusă decât la prima variantă de organizare.

La ambele variante de ALU, deoarece se compun din blocuri cu adâncime constantă, performanţele de viteză pot fi îmbunătăţite prin modul de realizare al blocurilor CGTA. Se va prezenta în continuare structuri de ALUE.

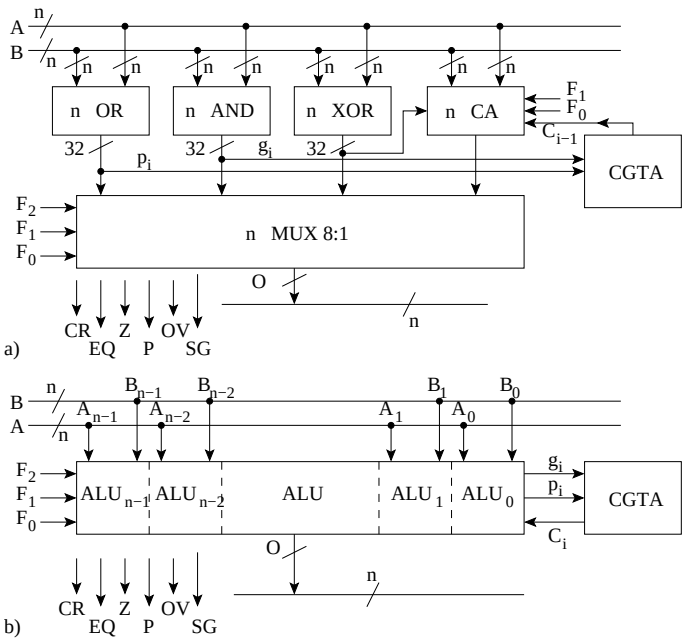


Figura 2.74 Variante de organizare a unei ALU: a) organizare ca o asamblare de blocuri/circuite specializate într-o structură de selectare secvenţială de date pe bază de multiplexor; b) organizare din n "felii" de unităţi logico-aritmetice de un bit conectate în paralel.

2.5.5.3 Structurarea unei ALU elementare

Funcţiile unei unităţi logico-aritmetice elementare, ALUE pot fi realizate cu ajutorul a diferite structuri de circuite. Este ales un anumit circuit de ALUE în funcţie de performanţele dorite sau modalitatea de implementare. Implementarea se poate face în oricare variantă expusă în acest capitol (cu porţi logice, DCD + porţi logice, multiplexoare, ROM, PLA; cu componente discrete sau integrate). Există o multitudine de "trasee" în realizarea unui sistem digital. Pentru funcţiile pe care trebue să le realizeze un sistem se poate defini o anumită arhitectură. Această arhitectură poate fi susţinută de mai multe organizării şi la rândul ei, o organizare poate fi implementată prin mai multe structuri de circuite. Ca exemplificare, pentru organizarea ALU, compusă din ALUE, se vor prezenta două modalităţi de structurare pentru celula de logico-aritmetică: ca un circuit logic combinaţional implementat prin porţi logice şi ca un circuit realizat pe un ROM sub forma unui tabel, LUT.

O celulă de unitate logico-aritmetică poate fi gândită ca o scalare la lungimea de cuvânt de 1 bit a unei organizări ALU de n biţi, de exemplu, pornind de la varianta de organizare din Figura 2.74-a. În acest sens, o variantă de circuit ALUE este cea din Figura 2.75-a care poate fi privită ca o structură de selectare secvenţială a datelor pe bază de multiplexor, vezi Figura 2.36-b, dar din porturi de un bit. Operaţiile realizate, dar la nivel de un bit, sunt aceleaşi ca cele definite în secţiunea anterioară (Figura 2.74-a) şi, de asemenea, sunt utilizate aceleaşi coduri pentru cuvântul $F_2F_1F_0$ de

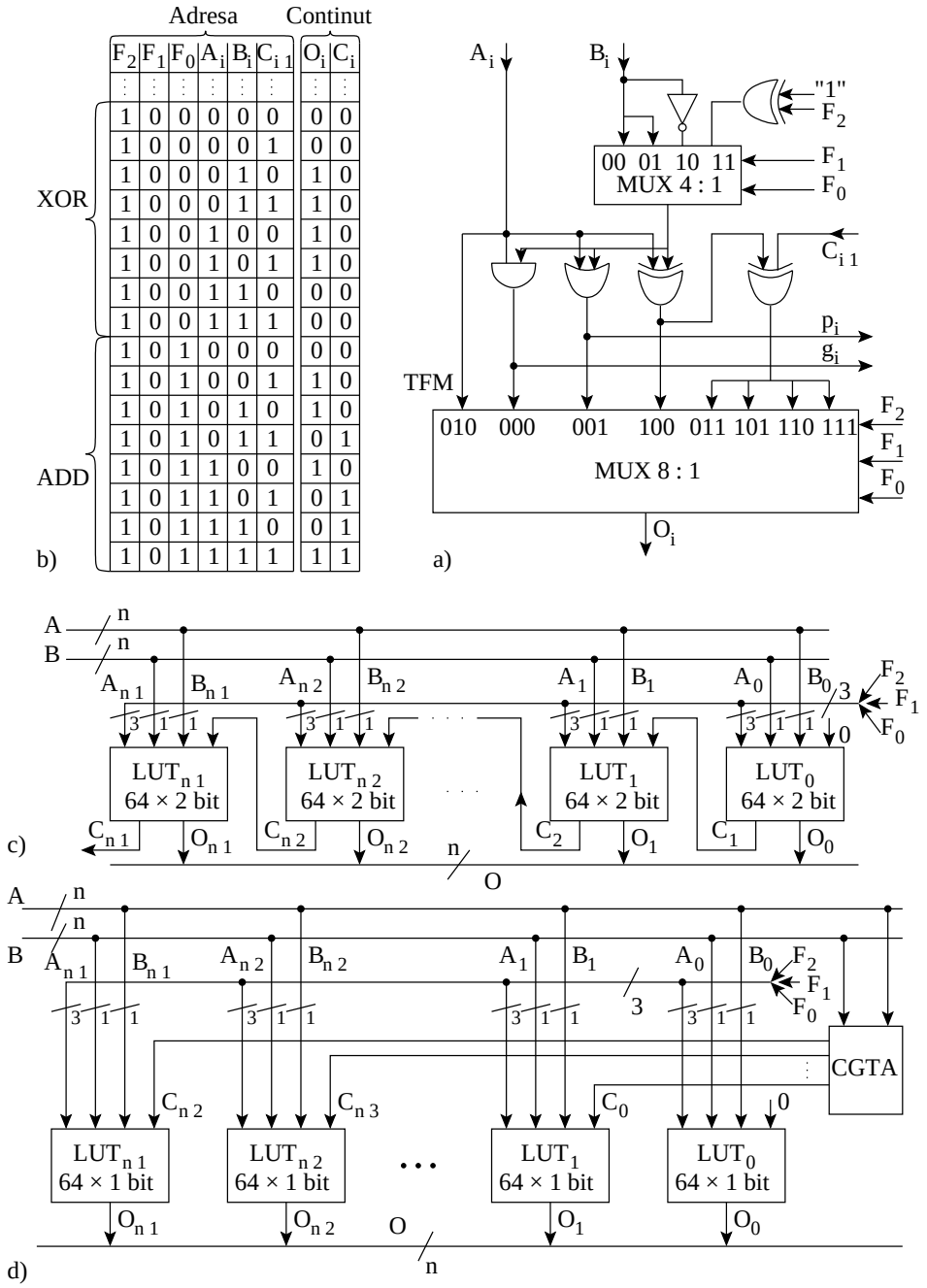


Figura 2.75 Structuri de unităţi logico-aritmetice elementare ALUE: a) ALUE structurată ca un circuit de selectare de date pe bază de multiplexor; b) tabelul de adevăr pentru operaţiile XOR şi ADD implementate în LUT; c) ALU structurată pe bază de LUT-uri cu transport progresiv; d) şi cu transport anticipat.

selectare la ieşirea MUX8:1 a unei operaţii logice sau aritmetice. Pentru operaţiile de produs logic, sumă logică şi sumă modulo 2 sunt introduse în circuit respectiv porţile AND, OR şi XOR, selectate respectiv prin codurile 000,001 şi 100. Codul de selectare 010 lasă cuvântul A să treacă nemodificat, TFM. Pentru aplicaţiile aritmetice care utilizează şi transportul anterior C_{i-1} (DCR - 010, ADD - 101, SUB - 110, INC - 111) este introdusă o a două poartă XOR. Cu un al doilea multiplexor $4:1$, selectat potrivit tot cu biţii F_1, F_0, se generează pentru bitul B_i următoarele valori: $B_i, \bar{B}_i, 0$ şi 1. Incrementarea, INC, se realizează ca o sumare cu unu $A+B+C_{-1} = A+0+1$, deci pentru B_i se generează valoarea 0 prin inversorul comandat XOR când $F_2 = 1$. Pentru decrementare, DCR, valorile lui B_i şi C_{-1} trebuie să fie inversate faţă de incrementare deci se generează pe inversorul comandat $B_i = 1$ când $F_2 = 0$. Porţile AND şi OR generează şi variabilele intermediare g_i şi p_i care se aplică circuitului de generare a transportului anticipat, CGTA, neinclus în acest desen. Ponderea în dimensiunea ALUE este determinată de partea de selectare, adică de multiplexoare.

Pentru implementarea ALUE sub forma unui LUT înscris într-un ROM se porneşte de la tabelul de adevăr a fiecărei operaţii ce trebuie realizată. Variabilele tabelului sunt intrările în ALUE adică intrările de selectare F_2, F_1, F_0, perechea de biţi A_i, B_i şi transportul anterior C_{i-1}, toate acestea formează un cuvânt de 6 biţi care va fi utilizat ca un cuvânt de adresare la circuitul ROM pe care se implementează LUT_i (fiecare LUT va avea 64 de adrese). Biţii înscrişi la o locaţie din ROM sunt cei doi biţi de ieşire din ALUE_i: C_i şi O_i, care acum sunt generaţi din LUT_i. În Figura 2.75-b este prezentat tabelul de adevăr numai pentru implementarea operatorilor AND şi XOR.

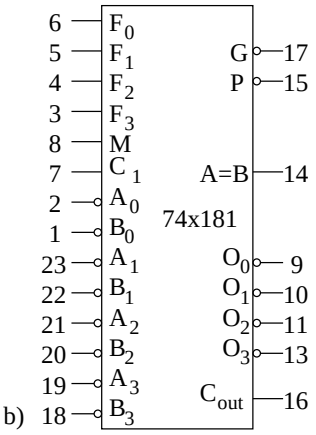
Implementarea unei ALU cu "felii", constând din LUT-uri înscrise în circuite ROM cu capacitatea de 64×2, biţi este reperezentată în Figura 2.75-c. Structurarea ALU este cu transport progresiv, bitul de transport C_i citit din LUT_i se aplică la LUT_{i+1}, deci un timp de calcul în $O(n)$. Se poate reduce acest timp de calcul pe ALU în $O(1)$ dacă se structurează ca în Figura 2.75-d. Transportul următor C_i nu se mai citeşte din ROM_i (capacitatea ROM-ului pentru un LUT se reduce la 64×1bit) ci se generează de catre un CGTA pe baza tuturor perechilor de biţi A_i şi B_i.

Ca unitate logico-aritmetică de patru biţi poate fi utilizat şi circuitul MSI 74xx181 reprezentat în Figura 2.76-b, iar operaţiile realizate sunt date în tabelul din Figura 2.76-a. Prin cuvântul de cod $F_3F_2F_1F_0$ se poate selecta una din cele 16 operaţii aritmetice asupra celor doi operanzi $A = A_3A_2A_1A_0$ şi $B = B_3B_2B_1B_0$ când valoarea semnalului de control este $M = 0$, respectiv se poate selecta una din cele 16 operaţii logice când $M = 1$. În expresiile funcţiilor din acest tabel produsul logic şi suma logică sunt notate prin simbolurile $\cdot$ şi $+$ iar pentru adunare şi scădere sunt folosite cuvintele plus şi minus. Operaţiile logice ($M = 1$) sunt realizate numai între perechile A_i şi B_i ($i = 0, 1, 2, 3$), nu există semnal de transport de intrare $C_{-1} = 0$, de asemenea nu există transporturi între ranguri. În schimb în operaţiile aritmetice ($M = 0$) pe lângă perechile A_i şi B_i trebuie considerat transportul de intrare C_{-1} şi transporturile între ranguri. Structura internă a circuitului 74xx181 este realizată pentru transport anticipat.

Atenţionăm faptul că semnalele de intrare $A_3\_L \div A_0\_L$, $B_3\_L \div B_0\_L$ şi cele de ieşire $O_3\_L - 0\_L$ sunt active în starea L. Se poate ca la circuitul 74xx181 să se considere aceste semnale active în starea H dar atunci expresiile funcţiilor date în tabelul anterior se modifică in felul următor. Ca circuit logic, $M = 1$, la aplicarea

Intrari de selectare				Functii	
F_3	F_2	F_1	F_0	Aritmetice (M=0)	Logice (M=1)
0	0	0	0	θ=A minus 1 plus C_1	$\theta=\overline{A}$
0	0	0	1	θ=A·B minus 1 plus C_1	$\theta=\overline{A}+\overline{B}$
0	0	1	0	$\theta=A\cdot\overline{B}$ minus 1 plus C_1	$\theta=\overline{A}+B$
0	0	1	1	θ=1111 plus C_1	θ=1111
0	1	0	0	$\theta=A$ plus $(A+\overline{B})$ plus C_1	$\theta=\overline{A}\cdot\overline{B}$
0	1	0	1	$\theta=A\cdot B$ plus $(A+\overline{B})$ plus C_1	$\theta=\overline{B}$
0	1	1	0	θ=A minus B minus 1 plus C_1	$\theta=A\oplus B$
0	1	1	1	$\theta=A+\overline{B}$ plus C_1	$\theta=A+\overline{B}$
1	0	0	0	θ=A plus (A+B) plus C_1	$\theta=\overline{A}\cdot B$
1	0	0	1	θ=A plus B plus C_1	$\theta=A\oplus B$
1	0	1	0	$\theta=A\cdot\overline{B}$ plus (A+B) plus C_1	θ=B
1	0	1	1	θ=A+B plus C_1	θ=A+B
1	1	0	0	θ=A plus A plus C_1	θ=0000
1	1	0	1	$\theta=A\cdot B$ plus A plus C_1	$\theta=A\cdot\overline{B}$
1	1	1	0	$\theta=A\cdot\overline{B}$ plus A plus C_1	$\theta=A\cdot B$
1	1	1	1	θ=A plus C_{in}	θ=A

a)



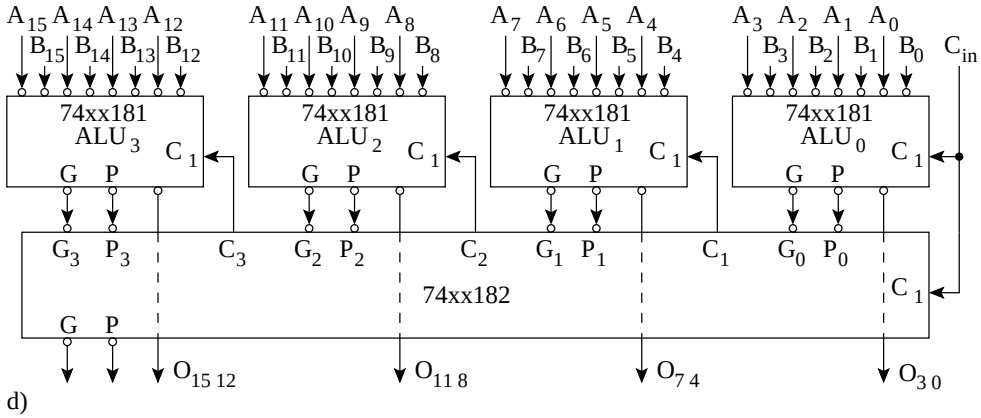


Figura 2.76 Realizarea unităţilor logico-aritmetice pe bază de circuite standard: a) tabelul de adevăr şi (b) reprezentarea circuitului unitate logică-aritmetică de patru biţi 74xx181; c) reprezentare circuitului pentru calculul transportului anticipat, 74xx182; d) structurarea unei ALU de 16 biţi prin punere în paralel a 4 circuite 74xx181 iar calculul transportului anticipat se realizează în exterior cu circuitul 74xx182.

unui cod de selectare $F_3F_2F_1F_0$ se obţine o funcţie logică duală (relaţia 1.2) celei care este indicată în tabel pentru acel cod de selectare. În schimb, ca circuit aritmetic, $M = 0$, la aplicarea unui cod de selectare se obţine o funcţie aritmetică, dar aceasta este diferită faţă de cea dată în tabel pentru acel cod de selectare (este necesar a se cunoaşte documentaţia tehnică a circuitului).

Există şi două ieşiri G_L şi P_L care exprimă o generare şi o propagare peste toate cele patru ranguri ale circuitului 74xx181 şi care se calculează conform relaţiilor 2.25 şi 2.26 astfel:

$$\begin{aligned} G\_L &= \overline{(g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2p_1g_0)} \\ P\_L &= \overline{p_3p_2p_1p_0} \end{aligned} \tag{2.28}$$

Aceste două ieşiri permit extensia, în implementarea de ALU, pentru cuvinte multiplu de patru biţi prin utilizarea de circuite 74xx181. Considerând fiecare circuit 74xx181 numai ca o pereche de biţi, pentru care sunt determinate valorile variabilelor intermediare G şi P, se pot calcula cu aceste variabile intermediare valorile de transport anticipat pe un grup de circuite. De exemplu, pentru un grup de patru circuite ALU (ALU_0,ALU_1,ALU_2,ALU_3) de patru biţi fiecare, pe baza perechilor de variabile intermediare de la fiecare circuit (G_0, P_0),(G_1, P_1),(G_2, P_2),(G_3, P_3) precum şi a transportului de intrare C_{-1}, se pot calcula valorile de transport anticipat C_0, C_1, C_2 respectiv la intrările de la ALU_1,ALU_2 şi ALU_3. Pe baza relaţiilor 2.25 şi 2.26 pentru C_0, C_1 şi C_2 se obţin expresiile:

$$\begin{aligned} C_0 &= G_0 + P_0C_{-1} \\ C_1 &= G_1 + P_1G_0 + P_1P_0C_{-1} \\ C_2 &= G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_{-1} \end{aligned}$$

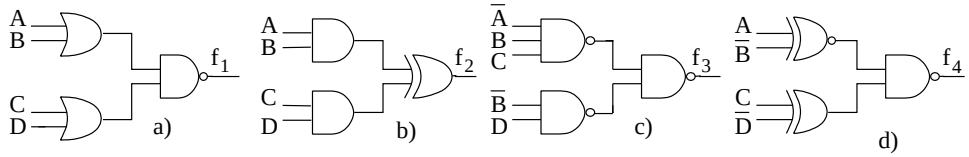
Circuitul MSI 74xx182, reprezentat în Figura 2.76-c, calculează transporturile anticipate C_0, C_1, C_2 pentru un grup de patru unităţi ALU. Acest circuit mai generează şi o pereche (G, P) de variabile intermediare pe un grup de patru unităţi ALU, această pereche fiind necesară pentru calculul transportului anticipat în exteriorul unui grup de patru circuite MSI 74xx182.

O exemplificare de realizare a unei unităţi logico-aritmetice de 16 biţi pe baza a 4 circuite ALU de 4 biţi (74xx181) este prezentată în Figura 2.76-d. Circuitul pentru calculul transportului anticipat 74xx182 calculează pe baza perechilor (G, P) de la fiecare unitate logico-aritmetică (74xx181) valorile de transport anticipat pentru ALU_3, ALU_2 şi ALU_1. Dar, unitatea logico-aritmetică se poate extinde de la 16 biţi la 64 de biţi, iar această structurare poate fi gândită ca fiind formată din 4 grupuri de câte 16 biţi. În exteriorul acestor patru grupuri de câte 16 biţi se realizează, cu un circuit 74xx182, o cale pentru calculul transporturilor anticipate, necesare grupurilor ce conţin rangurile de biţi $31-16, 47-32, 63-48$, pe baza perechilor G, P de grup generate de cele 4 circuite 74xx181 precum şi a transportului iniţial C_{-1}. Deci, la această ALU de 64 de biţi există în exteriorul unui grup de patru circuite ALU, de patru biţi, o cale realizată cu un 74xx182 pentru calculul anticipat al transportului pe grup. Apoi, în exteriorul acestor patru circuite 74xx182 există încă o cale pe un alt 74xx182 pentru calculul anticipat al transporturilor între grupuri. În total 16 circuite ALU 74xx181 şi 5 circuite pentru calculul transportului anticipat 74xx182.

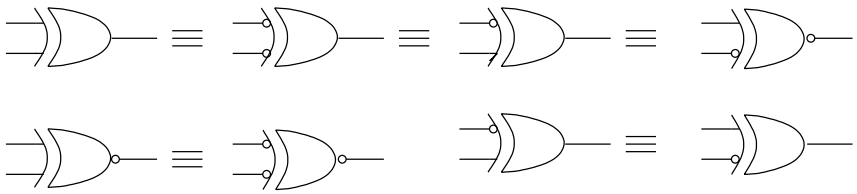
2.6 PROBLEME

P2.1 Pentru un cuvânt de n biţi, $x_{n-1}x_{n-2}\dots x_1x_0$, să se implementeze funcţia paritate f (=1 pentru un număr impar de biţi 1, =0 pentru un număr par de biţi 1 în cuvânt) cu porţi XOR2. În ce condiţii aceleaşi structuri de circuit, în care se face substituţia $XOR2 \rightarrow NXOR2$ calculează aceeaşi funcţie?

P2.2 Pentru circuitele din figură să se deducă expresiile f_1, f_2,f_3 si f_4, apoi să se construiască tabelele de adevăr.



P2.3 Să se demonstreze echivalenţele grafice din figură.



P2.4 Să se complementeze şi să se aducă la forma minimă expresiile logice următoare:

a) $f = \overline{[(\overline{\overline{AB}})A]} \cdot \overline{[(\overline{\overline{AB}})B]}$, funcţia A $\oplus$ B exprimată prin operatorul NAND;

b) $f = \overline{(\overline{A+B+\overline{C}})(\overline{\overline{AB}+\overline{CD}})+\overline{BCD}}$;

c) $f = (ABC + B\overline{C}D) + (\overline{\overline{ACD} + \overline{B}\,\overline{C}\,\overline{D} + BC\overline{D}})$.

P2.5 Să se minimizeze urmatoarele funcţii utilizând diagrame V-K:

a) $f_1(A,B,C) = \sum_0^7(0,2,3,4,5,7)$;

b) $f_2(A,B,C,D) = \sum_0^{15}(0,1,4,5,9,11,13,15)$;

c) $f_3(A,B,C,D,E) = \sum_0^{31}(0,4,8,9,10,11,12,13,14,15,16,20,24,28)$.

Pentru punctele a) si b), să se exprime forma minimă atât ca sumă de produse cât si ca produs de sume.

P2.6 Să se minimizeze următoarele funcţii utilizând diagrame V-K:
a)$F = \overline{A}BC + A\overline{B}\,\overline{C} + ABC$; *b)*$F(A,B,C) = \sum_0^7(1,3,5,6,7)$;
c)$F(A,B,C,D) = \overline{A}BC + A\overline{D} + BC\overline{D}$;
d)$F(A,B,C,D) = \sum_0^{15}(1,3,4,5,6,9,11,12,13,14)$;
e)$F(A,B,C,D,E) = \sum_0^{31}(0,2,8,10,16,18,24,26)$;
f)$F(A,B,C,D) = \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}B\overline{C}D + ABCD + A\overline{B}\,\overline{C}D$;

$g) F(A,B,C,D) = \prod_0^{15}(2,3,4,6,7,10,11,12)$;
$h) F(A,B,C,D,E) = \prod_0^{31}(0,2,4,6,8,9,10,11,12,14,16,17,18,19,24,25,26,27)$

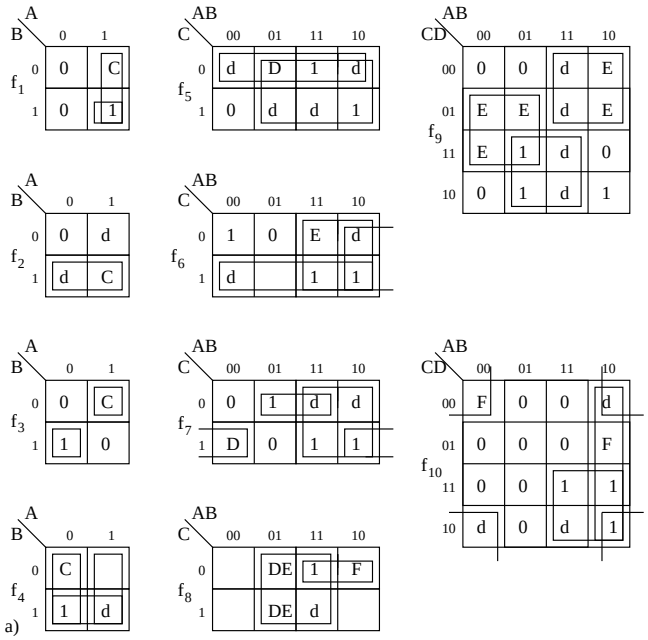
P2.7 Să se identifice implicanţii primi esenţiali pentru următoarele expresii:
$a)$ $f(A,B,C,D) = \sum_0^{15}(1,5,7,8,9,10,11,13,15)$;
$b)$ $f(A,B,C,D,E) = \sum_0^{31}(5,7,9,12,13,14,15,20,21,22,23,25,29,31)$.

P2.8 Utilizând diagrama V-K, să se arate că funcţia:
$f_1(A,B,C,D) = \sum_0^{15}(0,2,5,7,8,10,13,15)$
este negata funcţiei: $f_2(A,B,C,D) = \sum_0^{15}(1,3,4,6,9,11,12,14)$,
şi este identică cu funcţia: $f_3(A,B,C,D) = \prod_0^{15}(1,3,4,6,9,11,12,14)$

P2.9 Să se minimizeze următoarele funcţii utilizând diagrame V-K:
$a)$ $f(A,B,C,D) = \sum_0^{15}(2,3,4,5,13,15) + \sum_0^{15} d(8,9,10,11)$;
$b)$ $f(A,B,C,D) = \sum_0^{15}(1,5,7,9,13,15) + \sum_0^{15} d(8,10,11,14)$;
$c)$ $f(A,B,C,D) = \sum_0^{15}(0,2,4,8,10,14) + \sum_0^{15} d(5,6,7,12)$;
$d)$ $f(A,B,C,D,E) = \sum_0^{31}(1,3,4,6,9,11,12,14,17,19,20,22,25,27,28,30) +$
$+ \sum_0^{31} d(8,10,24,26)$;

P2.10 Se consideră funcţia:
$f(A,B,C,D) = \sum_0^{15}(3,6,11,14,15) + \sum_0^{15} d(2,5,12,13)$
Să se exprime ca o funcţie de trei variabile şi ca o funcţie de două variabile: a) utilizând tabelul de adevăr al funcţiei; b) utilizând diagrama V-K.

P2.11 Pentru funcţiile reprezentate în diagramele V-K din figură să se scrie forma minimă.



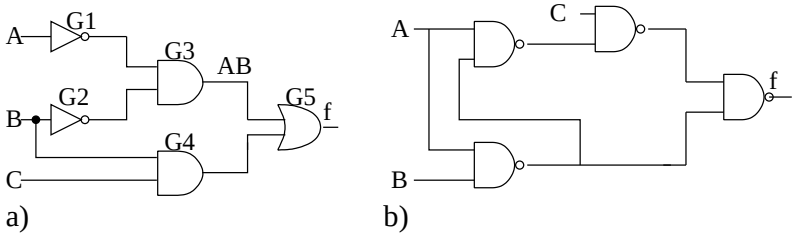
P2.12 Să se introducă variabile reziduu în expresia coeficienţilor funcţiilor de la problema P2.6, astfel încât în fiecare funcţie să fie numai două variabile. Reduceţi numărul de variabile atât folosind tabelul de adevăr cât si diagrama V-K. Pentru

funcţiile de două variabile rezultate din diagramele V-K să se scrie forma minimă. Pentru aceste forme minime să se compare rezultatele cu cele obţinute la problema P2.6.

P2.13 Să se realizeze sinteza unui circuit convertor de cod din codul zecimal codificat binar BCD 2-4-2-1 în codul de afişare pe matrice cu şapte segmente. Matricea de şapte segmente este TIL-312. Toate cele şapte LED-uri au anodul alimentat comun de la +5V. Un segment este luminat când pe catodul său este comandat potenţialul logic "0".

P2.14 Funcţionarea a două motoare M_1 şi M_2 este comandată de trei întrerupătoare: S_1, S_2 şi S_3. Motorul M_2 funcţionează tot timpul cât cele trei întrerupătoare sunt închise. Motorul M_1 funcţionează dacă fie S_2, fie S_1 (dar nu şi simultan) sunt închise iar S_3 este deschis.Să se realizeze circuitul logic combinaţional care implementează această comandă.

P2.15 Să se analizeze, utilizând diagramele de timp ale semnalelor şi diagramele V-K, dacă circuitul din figura (a) poate genera hazard static 1. În cazul apariţiei hazardului, să se propună soluţia de eliminare.



P2.16 Să se implementeze funcţia: $f(A,B,C,D) = \Pi_0^{15}(1,3,4,5,7,10,11,12,14,15)$ sub formă OR-AND fără a produce hazard static.

P2.17 Să se explice de ce circuitele care implementează mintermi pot produce hazard static 1 iar cele care implementează maxtermi pot produce hazard static 0.

P2.18 Să se pună în evidenţă hazardul static pentru fiecare din următoarele funcţii şi să se determine un circuit care elimină hazardul şi îndeplineşte aceeaşi funcţie logică:
a) $f(A,B,C,D) = \sum_0^{15}(5,7,8,9,10,11,13,15)$;
b) $f(A,B,C,D) = \sum_0^{15}(5,7,13,15)$;
c) $f(A,B,C,D) = \sum_0^{15}(0,2,4,6,12,13,14,15)$;
d) $f(A,B,C,D) = (\overline{A}+\overline{B}+D)(\overline{A}+\overline{C}+\overline{D})(A+B+\overline{C})$;
e) $f(A,B,C,D) = (\overline{A}+B+D)(A+B+\overline{C}+\overline{D})(A+C)(\overline{B}+C)$;
f) $f(A,B,C,D) = \overline{B}\,\overline{C}\,\overline{D} + A\overline{B}C + BC$.

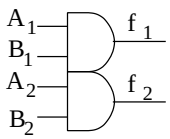
P2.19 Să se determine dacă circuitul din figura (b), de la P2.15, produce hazard static. În caz afirmativ, să se modifice structura astfel încât să fie eliminat hazardul.

P2.20 Pentru funcţia $f(x_2,x_1,x_0) = \sum_0^7(3,5,7)$ să se deducă diagrama de decizie binară redusă (ROBDD).

P2.21 Următoarea funcţie să se implementeze cu un număr minim de porţi NAND.

$$f = \overline{A}BD + AC\overline{D} + B\overline{C}D + A\overline{B}C$$

P2.22 Să se realizeze o poartă DAR care are patru intrări şi două ieşiri cu simbolul grafic
din figura alăturată. Funcţia realizată trebuie să fie adevărată când în statuarea valorilor variabilelor de intrare din primul grup A_1, B_1 sau din al doilea grup A_2, B_2 se aplică sensul conjuncţiei DAR. Privind simetria simbolului grafic funcţia realizată trebuie să fie simetrică în raport cu variabilele A si B din fiecare grup; de asemenea trebuie să fie simetrice cele două grupuri. Apoi, cu această poartă DAR să se implementeze funcţia $f(A_1, B_1, A_2, B_2) = \sum_0^{15}(3, 7, 11, 12, 13, 14)$ cu o poartă DAR plus o poartă OR cu două intrări.



P2.23 Utilizând numai 4 porţi NAND cu 8 intrări să se implementeze un codificator 16:4. Care sunt nivelurile active de semnal pentru intrări şi pentru ieşiri?

P2.24 Pe baza circuitului codificator prioritar cu 8 intrări 74XX148, , să se structureze un codificator prioritar cu 16 intrări.

P2.25 Să se implementeze un CLC la intrarea căruia se aplică un cuvânt M de opt biţi şi un cuvânt N de trei biţi. Ieşirea f a circuitului va fi activă când M este multiplu de 2^N.

P2.26 Să se implementeze un CLC la intrarea căruia se aplică un cuvânt M de 16 biţi şi un cuvânt N de 2 biţi. Ieşirea f a circuitului va fi activă când M este un multiplu de 2^{2^N}.

P2.27 Într-un cuvânt de un byte $X = x_7x_6x_5x_4x_3x_2x_1x_0$ să se determine când există doar un singur bit cu valoarea zero.

P2.28 Să se proiecteze un CLC cu opt intrări I_{i_L} şi opt ieşiri O_{i_L}, $i = 0, 1 \ldots 7$. Circuitul generează numai ieşirea $O_{i\_L} = 0$ în care i este poziţia celui mai semnificativ bit activat din cuvântul de opt biţi aplicat pe intrare.

P2.29 Să se realizeze un CLC prin intermediul căruia 8 periferice sunt legate la un microprocesor (μP). La activarea unuia sau a mai multor periferice acest circuit va genera către μP o cerere de întrerupere IRQ_L şi codul perifericului activat cu nivelul cel mai ridicat de prioritate; nivelul de prioritate creşte de la 0 la 7. Excepţie de la această regulă este numai în cazul în care sunt activate simultan mai multe periferice printre care sunt activate şi perifericele 7 şi 2, în acest caz se generează catre μP codul perifericului 2.

P2.30 Utilizând circuitul 74XX138, DCD3:8, să se realizeze o structură de DCD5:32.

P2.31 Următoarele funcţii: $Y_1 = 2^x$ şi $Y_2 = 2^{2^x}$ să fie implementate pe circuite DCD3:8, x fiind un număr binar în intervalul [000,111].

P2.32 Să se implementeze operaţia: $y = 2^{x_1} + 2^{x_2}, \quad x_1, x_2 \in [0, 7], \quad x_1 \neq x_2$

P2.33 Să se realizeze un CLC care implementează funcţia $y = 2^{(x_1+x_2)}, \quad x_1, x_2 \in [0, 7]$

P2.34 Utilizând circuitul 74LS138, decodificator 3:8, să se implementeze funcţia:
$f(A, B, C, D) = \sum(0, 1, 3, 5, 7)$

P2.35 Utilizând circuite 74LS138, decodificator 3:8 ,să se implementeze următoarele funcţii:
a) $f_1(A, B, C) = \sum(0, 2, 5, 7)$; *b)* $f_3(A, B, C, D) = \sum(0, 3, 5, 6, 9, 10, 11, 12, 13)$;
c) $f_2(A, B, C, D) = \prod(2, 3, 4, 7)$; *d)* $f_4(A, B, C, D) = \prod(2, 3, 6, 7, 8, 9, 13, 14, 15)$.

P2.36 Implementaţi o celulă sumator complet cu ajutorul unui circuit decodificator 74LS138.

P2.37 Să se implementeze cu circuite 74LS138, DCD3:8, circuitul logic combinaţional cu ieşiri multiple definit de funcţiile:

$f_1 = P_0 + P_3 + P_5 + P_7 \qquad f_2 = P_1 + P_2 + P_4 + P_5 + P_8 + P_{11} + P_{12} + P_{14} + P_{15}$

$f_3 = P_3 + P_4 + P_6 + P_{12} + P_{14} + P_{15} \qquad f_4 = P_1 + P_2 + P_5 + P_6 + P_7 + P_8 + P_9 + P_{15}$

P2.38 Se pot implementa următoarele funcţii doar cu două circuite integrate?

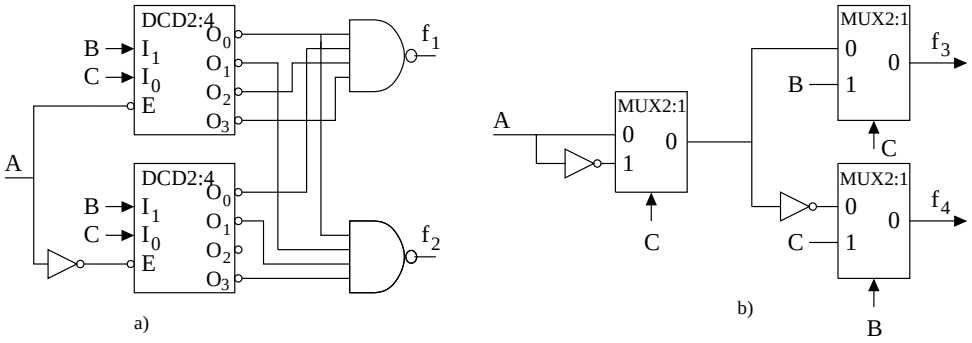
$f_1 = \overline{A}\,\overline{B}\,\overline{C} + ABC \quad , f_2 = \overline{A}\,\overline{B}C + AB\overline{C} \quad , f_3 = \overline{A}B\overline{C} + A\overline{B}C \quad , f_4 = A\overline{B}\,\overline{C} + \overline{A}BC$

P2.39 Un circuit 74XX138 DCD3:8 este comandat pe intările A,B,C cu semnalele de ieşire de la un numărător asincron modulo 8. Să se analizeze momentele când pe ieşirile circuitului pot apare glitch-uri.

P2.40 La celula sumator complet, implementată în problema P2.36, se aplică pe cele trei intrări A,B,C, în locul semnalelor A_i, B_i şi C_{i-1} semnalele de ieşire de la un număr asincron modulo 8. Să se determine la care comutarea dintre stările numărătorului asincron ieşirile s_i şi C_{i-1} ale celulei se pot genera glitch-uri.

P2.41 Numai cu două circuite 74XX138, DCD3:8, să se realizeze un DCD4:16.

P2.42 Să se determine funcţiile implementate pe următoarele circuite.

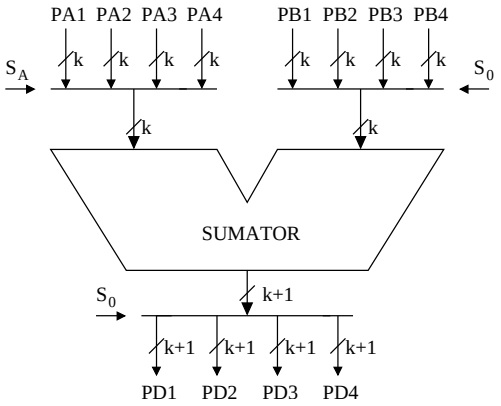


P2.43 Să se organizeze un MUX32:1 pe baza circuitelor: 74XX151 MUX8:1; 74XX153 MUX4:1; 74XX139 DCD2:4. Toate aceste circuite au o intrare de validare G activă în starea low; circuitul 74XX151 generează atât ieşirea negată cât şi nenegată.

P2.44 Să se realizeze o selectare de date de la patru porturi P_3, P_2, P_1 şi P_0, fiecare de patru biţi, la o magistrală.

P2.45 În figură este schiţată posibilitatea de sumare a unui operand cu lungimea de k biţi de la unul din porturile PA1,PA2,PA3,PA4 cu un alt operand de aceeaşi lungime de la unul din porturile PB1,PB2,PB3,PB4 iar rezultatul este distribuit la unul din porturile destinaţie PD1,PD2,PD3,PD4. Să se structureze cele două circuite de selectare de pe intrare şi circuitul de distribuţie de pe ieşire. Să se scrie cuvintele de selectare S_A, B_B, S_D pentru următoarele transferuri: PA1+PB1→PD1 şi PA2+PB4→PD3.

P2.46 Pentru funcţia logică, $f(A,B,C) = \sum_0^7(0,1,3,6,7)$ să se realizeze o implementare:



a) numai cu MUX 2:1; *b*) numai cu MUX 8:1; *c*) cu MUX 2:1 şi MUX 4:1. Considerând ca o măsură a dimensiunii numărul de terminale $(2^n + n + 1)$ ale unui MUX $2^n : 1$, să se determine dimensiunea pentru fiecare dintre aceste implementări.

P2.47 Să se implementeze funcţiile:
$f_1(A, B, C, D) = \sum(0, 1, 3, 4, 7, 8, 10, 11, 15)$, $f_2(A, B, C, D) = \sum(3, 4, 5, 6, 13, 14, 15)$, $f_3(A, B, C, D) = \sum(0, 1, 4, 5, 6, 9, 12, 14)$
în următoarele variante: *a*) numai cu MUX 2:1; *b*) numai cu MUX 4:1; *c*) numai cu MUX 16:1; *d*) cu MUX 2:1 şi MUX 4:1 sau MUX 8:1. Să se aprecieze dimensiunea implementării prin numărul de circuite utilizate şi numărul de terminale.

P2.48 Fie funcţia $f(A, B, C, D) = \sum(3, 6, 11, 14, 15) + \sum d(2, 5, 12, 13)$. Să se implementeze cu circuitul 74LS151, MUX 8:1.

P2.49 Implementaţi cu circuite 74LS151, MUX 8:1 celula sumator complet. Prin introducerea unei variabile reziduu, să se realizeze o implementare şi cu circuitul 74LS153, MUX4:1.

P2.50 Fie funcţia $f(A, B, C, D, E) = \overline{A}BE + \overline{A}\,\overline{B}DE + A\overline{B}CE + AC\overline{D}E$. Să se implementeze cu un circuit 74LS151, MUX8:1.

P2.51 Utilizând circuitul 74LS151 MUX8:1, să se implementeze funcţia:
$f(A, B, C, D, E) = \sum(3, 6, 7, 10, 11, 19, 22, 26, 27, 30, 31)$
Se va folosi intrarea de validare pentru una din variabilele reziduu.

P2.52 Să se implementeze funcţia: $f(A, B, C, D, E) = \overline{A}BE + \overline{A}\,\overline{B}DE + A\overline{B}CE + AC\overline{D}E$
(prezentată şi în problema P2.50), utilizând circuitele 74LS138 DCD3:8 şi 74LS153 MUX4:1.

P2.53 Să se implementeze funcţia:
$f(A, B, C, D, E) = \sum(3, 6, 7, 10, 11, 19, 22, 26, 27, 30, 31)$
utilizând circuitul 74LS138 DCD3:8 şi circuitul 74LS153 MUX4:1. Să se compare structura obţinută cu implementarea problemei P2.51.

P2.54 Să se implementeze în ROM un convertor din codul binar 7 segmente în BCD.

P2.55 Pentru problema P2.27 să se conceapă o implementare pe circuit ROM.

P2.56 Să se implementeze pe un circuit ROM calculul pătratului numerelor cuprinse

în intervalul [0,7].

P2.57 Un sistem cu microprocesor cu o magistrală de adresare de 10 biţi are alocat pentru cele patru periferice ale sale #0, #1, #2, #3 următoarele adrese: 3C0H, 3C1H, 3C2H şi 3C3H. Să se realizeze variante de circuite de decodificare pentru aceste adrese.

P2.58 Un sistem cu microprocesor cu o magistrală de adresare de 16 biţi are alocat pentru cele patru periferice ale sale #0,#1,#2 şi #3 următoarele adrese: ED0CH, ED0DH, ED0EH şi ED0FH. Să se realizeze o variantă de decodificare completă pentru periferice utilizând un circuit 74LS138, un circuit comparator 74LS682 şi porţi.

P2.59 *a*) Să se conceapă un modul de memorie ROM de capacitate $8K \times 16$ biţi utilizând circuitele EPROM 2716. Într-un spaţiu de adresare de 64k ($A_{15} \div A_0$) acest modul va acoperi intervalul de adrese 64k÷ 56k. Circuitul EPROM2716 prezintă două semnale de control unul este selectare circuit CE_L/PRM (Chip Enable/ Programare) iar celălalt este validare ieşire OE_L (Output Enable - comanda ieşirii din TSL în stare normală); pentru obţinerea datelor pe ieşire trebuie activate ambele semnale simultan.

b) Utilizând circuite EPROM 2708, $1K \times 8$ biţi, (prezintă un singur semnal CS_L (Chip Enable/ Output Enable)) şi circuite decdificator 74XX138, 74XX139, să se structureze următoarele module de memorie: $1K \times 16$ biţi, $8K \times 8$ biţi, $8K \times 16$ biţi. Fiecare din aceste module are asignat un interval de adresare începând cu adresa 0000H; se consideră un spaţiu de adresare de 64K ($A_{15} \div A_0$).

P2.60 Pe un circuit ROM cu organizarea 64×1 bit să se implementeze funcţia:
$f(F,E,D,C,B,A) = \sum(0,1,2,3,4,5,6,9,11,12,13,15,16,19,21,25,$
$26,28,29,30,31,33,34,36,37,39,41,44,45,46,49,51,53,55,57,59,60,61,62)$

P2.61 Să se implementeze următoarele funcţii pe o structură de ROM cu capacitatea 64×1: $f_1(F,E,D,C,B,A) = \overline{F}\,\overline{D}\,\overline{B} + FDB\overline{A} + \overline{F}E\overline{D}\,\overline{C}B\overline{A}$;
$f_2(F,E,D,C,B,A) = FE\overline{D}A + FE\overline{C}A + FE\overline{D}\,\overline{C}B + FEDCB\overline{A}$;
$f_3(F,E,D,C,B,A) = F\overline{E}\,\overline{D}\,\overline{B} + F\overline{E}\,\overline{D}C\overline{A}$;
$f_4(F,E,D,C,B,A) = F\overline{E}DA + F\overline{E}D\overline{C}B + F\overline{E}DC\overline{B}$;

P2.62 Să se implementeze pe o memorie ROM de capacitate 32×8 biţi următoarele convertoare de cod: *a*) convertor din cod 2-4-2-1 la matrice cu şapte segmente (vezi P2.13 şi P2.65); *b*) convertor din BCD în cod EXCESS3 (codul EXCESS3 se obţine din codul BCD la care se adună cuvântul $0011|_2 = 3|_{10}$). Să se calculeze gradul de utilizare al memoriei, exprimat prin raportul dintre numărul de biţi înscrişi şi capacitatea memoriei.

P2.63 Utilizând circuite ROM de capacitate 256×8 biţi, în care sunt înscrise toate înmulţirile între numerele binare de patru biţi, precum şi circuite sumatoare de lungimi corespunzătoare, să se structureze un multiplicator simultan pentru cuvinte de un byte.

P2.64 Să se implementeze pe structuri PLA şi PAL (cu patru termeni produs cablaţi pe fiecare poartă OR) un convertor din BCD în cod Gray.

P2.65 Să se implementeze pe o structură PLA generic un convertor din codul 2421 în codul pentru o matrice cu şapte segmente. Codul 2421 este: $0 \rightarrow 0000; 1 \rightarrow 0001;$ $2 \rightarrow 1000; 3 \rightarrow 1001; 4 \rightarrow 1010; 5 \rightarrow 1011; 6 \rightarrow 1100; 7 \rightarrow 1101; 8 \rightarrow 1110; 9 \rightarrow 1111;$, iar configurarea cifrelor zecimale din LED-uri se consideră ca în Figura 2.37.

P2.66 Să se implemeteze pe o structură de circuit PLA generic un comparator digital pentru două cuvinte A şi B cu lungimea de 4 biţi. Circuitul generează la ieşire:

$A = B, A > B$ şi $A < B$. Nu se primesc semnale pentru relaţiile de ordonare de la un modul comparator de rang superior.

P2.67 Să se realizeze circuite pentru identitatea a două cuvinte.

P2.68 Utilizând circuitul comparator digital de patru biţi 74XX85 să se implementeze structuri care să realizeze următoarele operaţii: *a*) Pentru două cuvinte A şi B ieşirea f_s să fie adevărată când A<B. Când A>B ieşirea f_i să fie adevărată. *b*) Pentru două cuvinte A şi B să se realizeze următoarele relaţii de ordonare: egal(=), mai mic(<), mai mic sau egal($\leq$), mai mare(>), mai mare sau egal($\geq$), diferit($\neq$).

P2.69 De la ieşirile de pondere $2^3, 2^2, 2^1, 2^0$ ale unui numărător sincron în cod binar natural modulo 16 se obţine cuvântul $A = A_3A_2A_1A_0$, iar de la ieşirile complementare se obţine cuvântul în complement faţa de 1, $\overline{A} = \overline{A_3}\,\overline{A_2}\,\overline{A_1}\,\overline{A_0}$. Cuvintele A şi $\overline{A}$ se aplică pe intrările unui circuit comparator de patru biţi 74XX85. Să se deseneze formele de undă la cele trei ieşiri ale circuitului ($A > B, A = B, A < B$) când numărătorul parcurge toate cele 16 stări ($0 \div 15$).

P2.70 Utilizând circuitul comparator de patru biţi 74XX85 şi porţi să se realizeze o structură care generează următoarele relaţii de ordine:

$$A \geq N; M \leq A; M \leq A \leq N; A < M; A > N$$

ude M şi N sunt două constante cuprinse în intervalul $[0, 2^n - 1]$, n=4

P2.71 Utilizând trei circuite comparator digital de patru biţi 74XX85 şi porţi să se realizeze o structură care pentru trei variabile A, B, C activează (în stare 1) una din cele trei ieşiri f_A, f_B, f_C în funcţie de relaţia adevărată din următoarele trei:

$f_A = 1$ dacă $B < A > C$ $\quad f_B = 1$ dacă $A < B > C$ $\quad f_C = 1$ dacă $A < C > B$

P2.72 Utilizând circuitul 74XX283, sumator cu transport anticipat pentru cuvinte de patru biţi, să se realizeze diferite variante de sumatoare de trei biţi şi de doi biţi.

P2.73 Să se realizeze sinteza unei celule semisumator pentru sumare în sistemul de numeraţie în bază trei. Implementarea se va face cu porţi NAND.

P2.74 Să se structureze o celulă sumator modulo trei, apoi cu aceasta să se organizeze un sumator cu transport anticipat modulo 3^8.

P2.75 Să se structureze o celulă sumator pentru adunarea în cod BCD.

P2.76 Utilizând circuite 74LS181, unitate logico-aritmetică de patru biţi şi circuite 74LS182, unitate pentru calculul transportului anticipat pe patru ranguri, să se organizeze o unitate logico-aritmetică pentru cuvinte de 64 de biţi.

P2.77 Să se realizeze un circuit, format numai din celule sumator complet $\sum(3, 2)$, care să calculeze numărul (exprimat în binar natural, NBCD) de biţi care au valoarea 1 într-un cuvânt de 7 biţi, apoi să se extindă pentru un cuvânt de un byte.

P2.78 Utilizând celule sumator complet $\sum(3, 2)$ să se realizeze un modul care incrementează un cuvânt de trei biţi, la fel, un modul, pentru decrementare.

P2.79 Pentru un sumator de şase biţi realizat în varianta de sumator cu propagarea transportului , SPT, şi în varianta cu transport anticipat, STA, să se determine:

1. expresiile pentru C_5; 3. timpul minim de sumare, $T_{\sum}$.
2. numărul de porţi pentru implementarea lui C_5;

Se va considera organizarea de celulă $\sum(3, 2)$ din Figura 2.60-b, iar $C_{-1} = 0$.

P2.80 Realizaţi un circuit pentru sumarea a trei cuvinte de doi biţi.

P2.81 Realizaţi un circuit pentru multiplicarea a două cuvinte, unul de doi biţi $X = x_1x_0$ iar celălalt de trei biţi $Y = y_2y_1y_0$ utilizând fie numai module sumator pentru cuvinte de doi biţi şi porţi AND2 fie numai module sumator pentru cuvinte

de trei biţi şi porţi AND2.

P2.82 Să se structureze o Unitate Aritmetico - Logică Elementară, ALUE, care să realizeze următoarele operaţii:1) trece $\overline{B}_i$; 2) $A_i > B_i$; 3) $A_i + B_i + C_{i-1}$; 4) $A_i - B_i - C_{i-1}$.

P2.83 Utilizând o (celulă) ALUE (structurată la P2.82) să se organizeze o ALU de patru biţi.

P2.84 Să se structureze o ALUE care să realizeze următoarele opt operaţii: 1) înscrie zero (ştergere); 2) $(B - A)$; 3) $(A - B)$; 4) $(A + B)$; 5) $A \oplus B$; 6) $(A \cup B)$; 7) $(A \cap B)$; 8) înscrie 1 (set).

Capitolul 3

CIRCUITE LOGICE SECVENŢIALE, CLS

Circuitul Logic Combinaţional, CLC, exprimat formal prin tripletul(X,Y,F), determină pentru o configuraţie binară de intrare de n biţi, definită pe mulţimea $X = \{0,1\}^n$, o configuraţie de ieşire de m biţi, aparţinând mulţimii $Y \subseteq \{0,1\}^m$, adică realizează aplicaţia $F : X \to Y$. În funcţionarea unui CLC, cel puţin teoretic, nu se ia în considerare variabila timp, transferul intrare-ieşire fiind instantaneu; în suportul formal utilizat, algebra Booleeană, nu există variabila timp. Această lacună a suportului formal se plăteşte, uneori, sub forma generării de hazard la unele implementări ale CLC-ului, când pe durata timpului de transfer(neinstantaneu) valorile obţinute la ieşirea circuitului apar ca o violare a postulatului de existenţa complementarului: $x + \overline{x} = 0$ şi $x\overline{x} = 1$!. Totodată, la un CLC transferul este unidirecţional, nu există şi un transfer de la ieşire către intrare, adică o reacţie.

Într-o abordare foarte generală un **C**ircuit **L**ogic **S**ecvenţial, **CLS**, se poate considera ca o extensie a unui CLC cu o conexiune de reacţie şi în funcţionarea sa, implicit, se consideră şi variabila timp(propagarea neinstantanee). Se poate lua şi reciproca, adică un CLS numai cu transfer unidirecţional şi instantaneu este un CLC. Această abordare de definire, implicit, determină şi o structurare generală a unui CLS pornind de la un CLC.

Secvenţialitatea implică evenimente/stări care se succed în timp unul după altul. În funcţie de modul cum este marcată această tranziţie, de la un eveniment/stare la următorul, circuitele/sistemele pot fi asincrone sau sincrone. Pentru cele asincrone, această tranziţie/evoluţie, este determinată de însăşi structura circuitului/sistemului prin timpul său propriu de propagare(constanta de timp), pe când la cele sincrone tranziţia este marcată din exterior prin timpul, perioada/frecvenţa unui semnal de ceas/clock.

3.1 CIRCUITE LOGICE SECVENŢIALE ASINCRONE

Structura generală a unui **CLS asincron** este reprezentată în Figura 3.1, referită

uneori ca **structură Huffman**, care poate fi considerată că rezultă pornind de la un CLC căruia i s-au ataşat **căi de reacţie** prin intermediul elementelor de întârziere, notate cu $\Delta_0, \Delta_1, \ldots, \Delta_{k-1}$. **Variabilele de intrare(principale)** sunt: $x_{n-1}(t), \ldots x_1(t), x_0(t)$ iar **mărimile de ieşire** sunt: $y_{m-1}(t), \ldots, y_1(t), y_0(t)$, similar ca la un CLC cu n intrări m ieşiri. O configuraţie a variabilelor de intrare principale este un vector de intrare $X_i, i = 0, 1, 2, \ldots, (2^n - 1)$, aceşti vectori formează mulţimea intrărilor, notată cu X. De asemenea, configuraţiile mărimilor de ieşire formează mulţimea ieşirilor, notată cu Y. Asupra celor două mulţimi X şi Y, sunt corecte aceleaşi consideraţii care s-au expus la CLC(vezi relaţia 2.1). În plus, CLC-ul mai produce la momentul t încă alte k variabile de ieşire $w_{k-1}(t), \ldots, w_1(t), w_0(t)$ care prin legături de reacţie şi prin intermediul elementelor de întârziere Δ_i, $i = 0, 1, \ldots, (k-1)$, se aplică pe intrare ca **variabile de intrare secundare** la momentul $t + \Delta_i$, $z_{k-1}(t + \Delta_{k-1}), \ldots, z_1(t + \Delta_1), z_0(t + \Delta_0)$. Considerând că întârzierile Δ_i, introduse de elementele de întârziere sunt toate egale cu Δ, se pot scrie următoarele relaţii pentru cele k intrări secundare $z_i, i = 0, 1, 2, \ldots, (k-1)$: $z_{k-1}(t + \Delta) = w_{k-1}(t), \ldots, z_1(t + \Delta) = w_1(t), z_0(t + \Delta) = w_0(t)$. Rezultă că circuitul combinaţional din structura CLS-ului este un circuit cu $(n + k)$ intrări şi $(m + k)$ ieşiri.

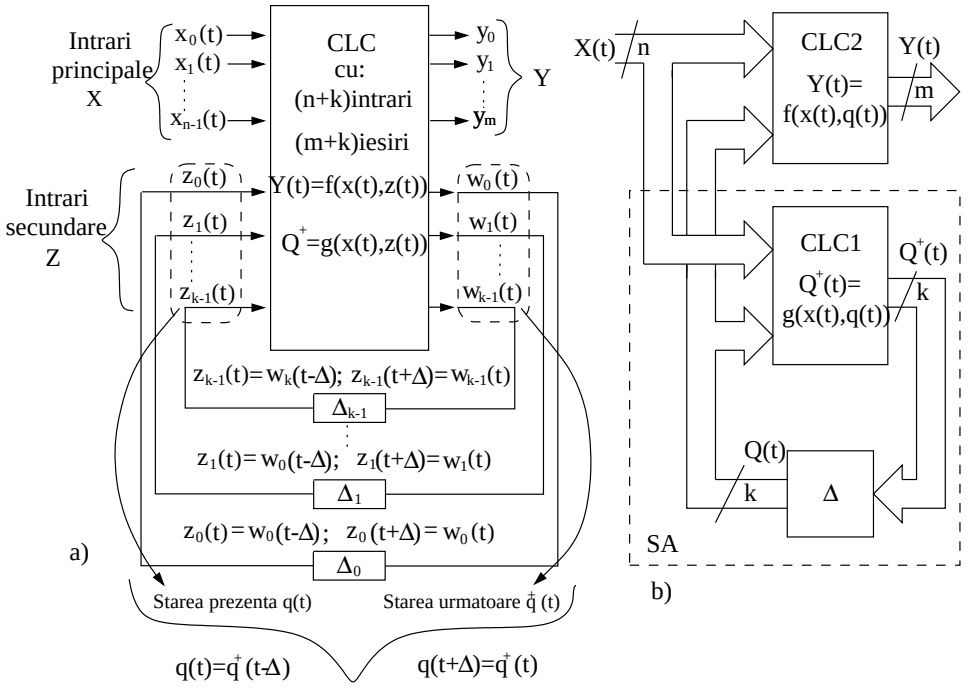


Figura 3.1 Circuitul logic secvenţial, CLS: a) structurare de principiu pentru un **CLS asincron(Huffman)**; b) structurare cu separarea funcţiei de transfer intrare-ieşire, de cea de tranziţie a stărilor.

Pentru întârzierile Δ_i nu trebuie să se înţeleagă şi să se considere câte un element fizic de întârziere introdus special pe câte o linie de reacţie. În implementările normale ale unui CLS asincron legătura de reacţie se realizează printr-o conectare directă de

la ieşire la intrarea secundară, deci ieşirea w_i devine instantaneu intrarea z_i(evident dacă timpul de propagare pe legătura de reacţie respectivă este zero). În schimb, propagarea de semnal în structura CLC-ului, de la intrările principale x_i şi intrările secundare z_i înspre ieşirile w_i, se realizează pe trasee cu anumite adâncimi (niveluri logice), deci există intârzieri care trebuie luate în considerare; de exemplu o întârziere de $5 \div 10$ns pe fiecare nivel logic. Dar, în explicaţia anterioară, pentru fiecare din semnalele $w_i,\ i = 0, 1, \ldots, (k-1)$ s-a considerat că există câte o întârziere Δ_i. Într-o abordare didactică, analiza unui CLS asincron se poate simplifica dacă se consideră că întârzierile de propagare din interiorul CLC-ului sunt zero dar, în compensaţie, pe fiecare linie de reacţie între w_i spre z_i se introduce întârzierea Δ_i aferentă ieşirii w_i din CLC. Simplificarea poate merge şi mai departe prin supoziţia facută anterior, că toate aceste întârzieri Δ_i, introduse pe legăturile de reacţie, sunt egale cu Δ. În acest mod de abordare didactic la CLS transferul intrare-ieşire este instantaneu prin CLC, în schimb variabilele w_i devin variabile de intrare secundară, z_i, numai după intervalul de timp Δ.

Configuraţia intrărilor secundare sau cuvântul format cu valorile binare ale celor k variabile secundare la momentul t,$z_{k-1}(t) \ldots, z_1(t), z_0(t)$ defineşte o mărime internă proprie circuitului secvenţial denumită **starea prezentă a circuitului, notată cu** $\mathbf{q}(t)$. Cuvântul format din valorile de ieşire $w_{k-1}(t) \ldots w_1(t), w_0(t)$ formează **starea următoare notată cu** $\mathbf{q}^+(t)$. În raport cu intrările principale şi ieşirile circuitului, care sunt mărimi vizibile în exterior(la borne), starea $q(t)$ este o mărime internă proprie circuitului, neaccesibilă la bornele circuitului. Datorită introducerii elementelor de întârziere pe linile de reacţie apar evidente următoarele relaţii între starea prezentă şi starea următoare:

$$q(t) = q^+(t - \Delta);\ q(t + \Delta) = q^+(t) \tag{3.1}$$

Adică starea prezentă $q(t)$ este identică cu starea următoare $q^+(t-\Delta)$, care a fost în urmă la momentul $(t-\Delta)$, sau starea prezentă $q(t+\Delta)$, care va fi la momentul $(t+\Delta)$, este egală cu starea următoare $q^+(t)$ din momentul t (starea următoare devine stare prezentă după întârzierea Δ).

Deoarece starea prezentă $q(t)$ este aceeaşi cu starea următoare $q^+(t-\Delta)$, care a fost în urmă la momentul $(t-\Delta)$ şi care a fost calculată în funcţie de stările anterioare, înseamnă că în funcţionarea circuitului, prin intrările de reacţie, intervine şi evoluţia anterioară a circuitului, adică **influenţează şi "istoria"**. Apare clar faptul că la un CLS, în raport cu un CLC, ieşirile sunt dependente nu numai de intrările prezente ci şi de intrările anterioare care au determinat succesiunea stărilor anterioare şi care se regăsesc în valoarea stării prezente. Partea combinaţională a CLS pe baza unei configuraţii de intrare $X(t)$ şi a stării prezente $q(t)$ calculează valorile configuraţiei de ieşire $Y(t)$ precum şi starea următoare $q^+(t)$. Aceste două transferuri, spre cele două tipuri de mărimi de ieşire, sunt evidenţiate în Figura 3.1-b prin reprezentarea separată a celor două parţi combinationale CLC2 şi CLC1.

Pentru o formalizare a exprimării funcţionării unui CLS se introduc următoarele notaţii:

> **Mulţimea** $Q = \{q_p, \ldots, q_2, q_1, q_0\}$ **este mulţimea stărilor circuitului**; pot exista maximum $p = 2^k$ stări exprimate printr-un cuvânt cu lungimea de k biţi de forma $z_{k-1} \ldots z_l \ \ldots z_1 z_0\ z_l \in \{0, 1\}$. Şi mulţimea Q, ca şi mulţimea Y, este incomplet definită $|Q| \leq 2^k$; mulţimea stărilor realizate de circuit este definită pe mulţimea nevidă a părţilor lui Q, adică pe $\mathcal{P}^*(Q)$

Funcţia de transfer f intrare-ieşire a CLS-ului exprimă procesul de modificare a ieşirilor în dependenţa de cuvântul de intrare şi de cuvântul stării prezente prin relaţii de forma:

$$\begin{array}{c} y_0(t) = f_0(x_{n-1}(t), \ldots, x_1(t), x_0(t);\ z_{k-1}(t), \ldots, z_1(t), z_0(t)) \\ y_1(t) = f_1(x_{n-1}(t), \ldots, x_1(t), x_0(t);\ z_{k-1}(t), \ldots, z_1(t), z_0(t)) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ y_{m-1}(t) = f_{m-1}(x_{n-1}(t), \ldots, x_1(t), x_0(t);\ z_{k-1}(t), \ldots, z_1(t), z_0(t)) \\ f : X \times Q \to Y \end{array} \tag{3.2}$$

Funcţia de tranziţie a stărilor g exprimă determinarea stării următoare; adică pe baza stării prezente şi a cuvântului de intrare se calculează starea următoare $q^+(t)$ prin relaţii de forma:

$$\begin{array}{c} w_0(t) = g_0(x_{n-1}(t), \ldots, x_1(t), x_0(t);\ z_{k-1}(t), \ldots, z_1(t), z_0(t)) \\ w_1(t) = g_1(x_{n-1}(t), \ldots, x_1(t), x_0(t);\ z_{k-1}(t), \ldots, z_1(t), z_0(t)) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ w_{k-1}(t) = g_{k-1}(x_{n-1}(t), \ldots, x_1(t), x_0(t);\ z_{k-1}(t), \ldots, z_1(t), z_0(t)) \\ g : X \times Q \to Q \end{array} \tag{3.3}$$

Fiecărui element din produsul cartezian $X \times Q$ poate să-i corespundă mai multe elemente din mulţimile Y sau Q. Deoarece ieşirile circuitului sunt definite pe părţi ale mulţimii de ieşire $\mathcal{P}^*(Y)$ şi la fel stările sunt definite pe părţi ale mulţimii stărilor $\mathcal{P}^*(Q)$ relaţiile 3.2 şi 3.3 trebuie scrise sub forma:

$$\begin{array}{c} f : X \times Q \to \mathcal{P}^*(Y) \\ g : X \times Q \to \mathcal{P}^*(Q) \end{array} \tag{3.4}$$

totuşi, uneori, pentru uşurinţa scrierii se utilizează formele anterioare. Aplicaţiile anterioare (3.2 şi 3.3) corespund cu cele din relatiile 3.4 când oricare element din $\mathcal{P}^*(Y)$ şi $\mathcal{P}^*(Q)$ are cardinalul 1(conţine un singur element).

Cu notaţiile introduse un CLS poate fi exprimat ca un cvintuplu:

$$CLS = (X, Y, Q, f, g) \tag{3.5}$$

Dacă mulţimea stărilor este mulţimea vidă $Q \equiv \emptyset$ (circuitul nu are "istorie") atunci: $g : X \times \emptyset \to \emptyset$ nu există funcţie de tranziţie a stărilor iar, $f : X \times \emptyset \to Y$ se reduce la $f : X \to Y$, deci se obţine:

$$CLS|_{Q \equiv \emptyset} \to CLC = (X, Y, f)$$

Rezultă că un CLS la care mulţimea stărilor este mulţimea vidă se reduce la un CLC (ieşirile sunt funcţie doar de vectorul de intrare X şi nu există reacţie). Se poate concluziona că **elementul care face trecerea de la combinaţional la secvenţial este conexiunea de reacţie**; şi invers, desfacerea reacţiei la un circuit secvenţial îl transformă în circuit combinaţional.

La un CLS, pe baza produsului cartezian între cuvântul $X(t)$ aplicat pe intrare şi al stării prezente $Q(t)$, circuitul combinaţional calculează cuvântul ieşirii $Y(t)$, relaţia

3.2, şi la fel, calculează starea următoare $Q^+(t)$, relaţia 3.3. După întârzierea Δ cauzată de transferul pe liniile de reacţie, starea următoare $Q^+(t)$ devine noua stare prezentă în momentul $(t+\Delta)$, $Q(t+\Delta) = Q^+(t)$.

Să presupunem că la intrarea principală se aplică cuvântul $X_1(t)$ şi starea prezentă a circuitului este $q_1(t) = z_{k-1}(t)\dots z_1(t)z_0(t)$. Pe baza produsului cartezian $X_1(t) \times q_1(t)$, pe circuitul combinaţional, se calculează cuvântul de ieşire $Y_1(t)$ precum şi cuvântul stării următoare $q_1^+(t) = w_{k-1}(t)\dots w_1(t)w_0(t)$, iar după întârzierea Δ, de propagare pe liniile de reacţie, starea următoare devine starea prezentă $q_2(t+\Delta) = q_1^+(t)$. Considerăm nemodificat cuvântul de intrare $X_1(t+\Delta)$ $(= X_1(t))$ din nou pe baza produsului cartezian $X_1(t+\Delta) \times q_2(t+\Delta)$ se calculează ieşirea $Y_2(t+\Delta)$ şi cuvântul stării următoare $q_2^+(t+\Delta)$ care după întârzierea Δ devine starea prezentă $q_3(t+2\Delta) = q_2^+(t+\Delta)$ şi aşa mai departe!. Dar, aceasta înseamnă că acest proces de evoluiţie din stare în stare (traseu), pentru acelaşi cuvânt aplicat pe intrare, este infinit, adică circuitul nu se stabilizează într-o anumită stare, denumită stare stabilă. **Condiţia ca o stare a circuitului să fie stabilă** se reduce la identitatea cuvântului stării prezente $q(t)$ cu cel al stării următoare $q^+(t)$, calculată pe baza produsului cartezian $X(t) \times q(t)$, adică $\mathbf{w}_i(t) = z_i(t)$ pentru toate valorile $i = 0, 1, 2, \dots, (k-1)$. Pentru o anumită intrare aplicată când cuvântul a ajuns într-o stare stabilă, adică nu se mai modifică în timp, circuitul poate fi considerat că evoluează permanent din aceeaşi stare tot cu aceeaşi stare. Uneori variabilele $w_i(t)$ ale stării următoare calculate sunt referite ca variabile de excitaţie, deoarece prin intermediul acestora circuitul este trecut într-o altă stare.

Exemplul 3.1 Pentru circuitul din Figura 3.2 să se analizeze stările stabile.

Soluţie. Circuitul prezintă o singură intrare x, o singură ieşire y, doi biţi pentru starea prezentă, z_1, z_0, exprimată prin cuvântul $z = z_1z_0$, şi doi biţi pentru starea următoare calculată, w_1, w_0, (funcţii de excitaţie), exprimată prin cuvântul $w = w_1w_0$. Întârzierile de propagare pe traseele porţilor logice şi pe conexiunea de reacţie s-au concentrat în două valori de timp de întârziere, Δ_1, Δ_0 care s-au introdus pe liniile de reacţie. Timpul de propagare pe o poartă poate fi specificat ca fiind, de exemplu, în intervalul $5 \div 10ns$, deci chiar pentru trasee cu topologie identică, pentru calculul funcţiilor de excitaţie w_1, w_0, timpii de întârziere sunt în general diferiţi, $\Delta_1 \neq \Delta_0$.

Din structura circuitului se deduc funcţiile de excitaţie (tranziţia circuitului), w_1, w_0 şi funcţia de transfer intrare-ieşire y

$$w_1 = x + z_1\overline{z_0}$$
$$w_0 = x + \overline{z_1}z_0$$
$$y = \overline{z_1}z_0 + z_1\overline{z_0}$$

Cu aceste relaţii pentru cele patru stări posibile ale circuitului, $z_1z_0 = 00, 01, 11, 10$ şi cele două valori ale variabilei de intrare $x = 0, 1$ se calculează valorile pentru w_1, w_0, care sunt prezentate în primele două tabele din Figura 3.2-b. În ultimul tabel din această figură, obţinut din cele două tabele anterioare, sunt figurate pentru fiecare pereche a produsului cartezian, $z_1z_0 \times x$, format între starea prezentă şi intrarea aplicată, valorile calculate ale stării următoare şi a ieşirii, $w_1, w_0; y$. Cuvintele de cod ale stărilor prezente sunt notate în partea stângă a tabelului, la începutul fiecărei linii, iar valorile aplicate intrării sunt figurate la capătul superior al fiecărei coloane din tabel. Din acest tabel, referit ca **tabelul de evoluţie al stărilor**, se poate deduce de exemplu că: pentru starea prezentă $z_1z_0 = 11$ şi intrarea aplicată $x = 0$ se calculează starea următoare $w_1w_0 = 00$ şi ieşirea $y = 0$, $(00/0)$,

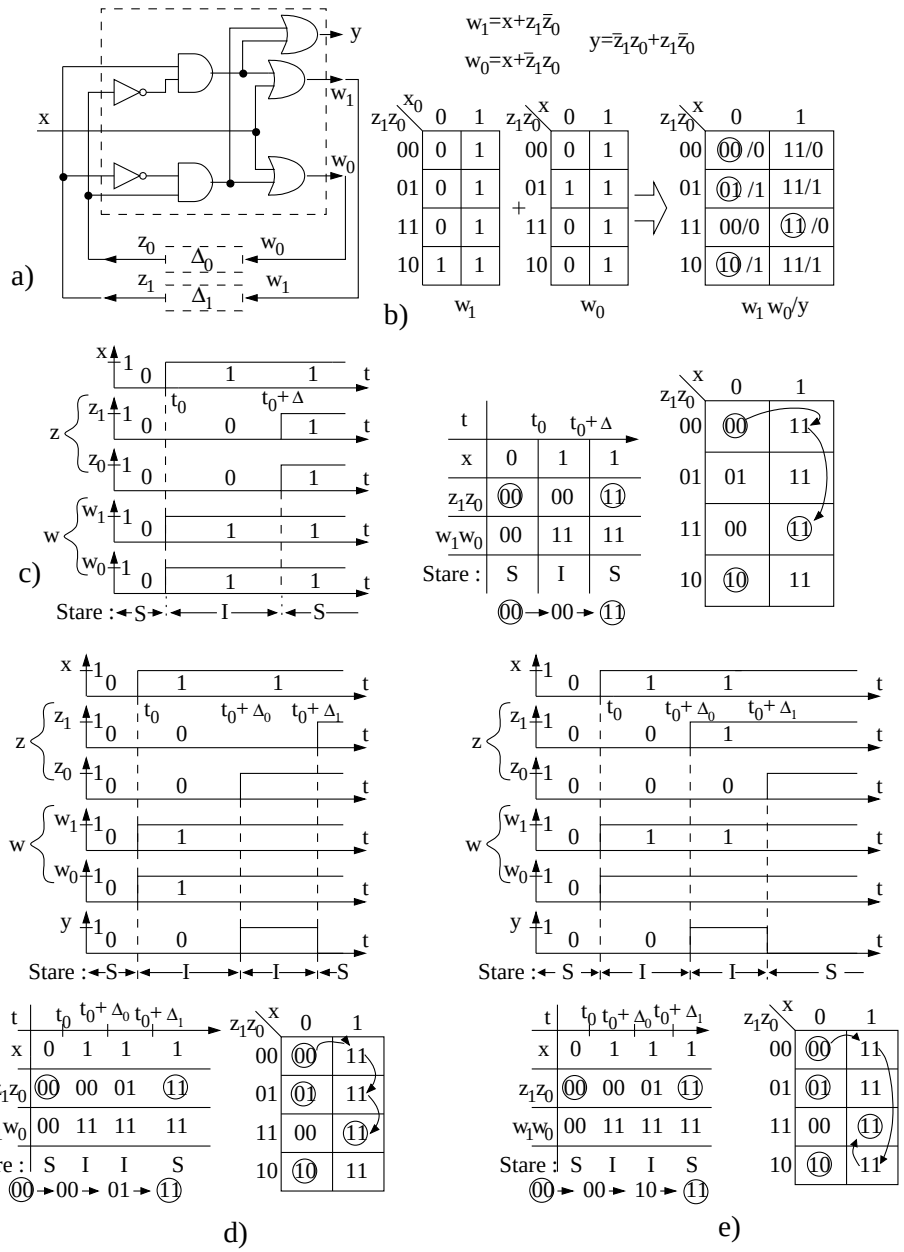


Figura 3.2 Analiza stărilor stabile pentru un circuit CLS asincron:
a) structură circuit; b) tabelul de evoluţie al stărilor; c) analiza traseului obţinut $\textcircled{00} \rightarrow 00 \rightarrow \textcircled{11}$ pentru cazul $\Delta_1 = \Delta_2 = \Delta$; d) analiza traseului obţinut $\textcircled{00} \rightarrow 00 \rightarrow 01 \rightarrow \textcircled{11}$ pentru cazul $\Delta_0 < \Delta_1$; e) analiza traseului obţinut $\textcircled{00} \rightarrow 00 \rightarrow 10 \rightarrow 11$ pentru cazul $\Delta_1 < \Delta_0$.

iar pentru aceeaşi stare prezentă 11 dar cu intrarea aplicată $x = 1$, se calculează starea următoare 11 şi $y = 0$, (11/0).

Condiţia de stabilitate pentru o stare, $w_1(t)w_0(t) \equiv z_1(t)z_0(t)$ impune ca pe fiecare linie din tabelul de evoluţie al stărilor, corespunzătoare unei stări prezente, să existe cel puţin o stare următoare calculată cu un cod identic cu cel al stării prezente. De exemplu, pe prima linie, corespunzătoare stării prezente $z_1z_0 = 00$, se genereaza 00/0 pentru $x = 0$ şi se generează 11/0 pentru $x = 1$, deci starea următoare calculată $w_1w_0 = 00$ este o stare stabilă(circuitul rămâne tot în starea prezentă 00, respectă condiţia $00 = 00$); dar în schimb starea următoare calculată $w_1w_0 = 11$ nu este o stare stabilă, pentru starea prezentă $z_1z_0 = 00$, deoarece $11 \neq 00$ (circuitul nu rămâne în starea prezentă). Stările următoare calculate care sunt stabile, se evidenţiază în tabelul de evoluţie al stărilor prin încercuire. Pe aceeaşi linie a tabelului pot exista mai multe stări stabile(încercuite), de exemplu la un tabel care are 2^n coloane, acesta corespunde unui circuit cu n intrări principale, deci aceleiaşi stări prezente i se pot aplica 2^n configuraţii de intrare. Tabelul de evoluţie al stărilor este referit ca **tabelul primitiv de evoluţie al stărilor** dacă pe fiecare linie a sa există doar o singură stare stabilă (incercuită), cum este cazul acestui tabel analizat. De asemenea, cuvântul care reprezintă **produsul cartezian $Q \times X$ este referit ca starea totală a circuitului**. O stare totală este stabilă dacă la intersecţia corespunzătoare a stării prezente şi a intrării aplicate, în tabelul de evoluţie al stărilor, este o stare următoare calculată notată încercuit şi respectiv, este o stare totală instabilă pentru o stare următoare calculată notată neîncercuit. Pentru acest circuit, din tabelul de evoluţie, rezultă că există patru stări totale stabile $z_1z_0x = 000, 010, 111, 100$ şi patru stări totale instabile $001, 011, 110, 101$.

La un circuit secvenţial asincron, având o stare prezentă stabilă prin aplicarea unui cuvânt de intrare, evoluţia poate fi: tot în aceeaşi stare stabilă, într-o altă stare stabilă, în una dintre mai multele stări stabile posibile (cursă critică) sau o ciclare între mai multe stări instabile (oscilator). Pentru circuitul din figură se vor analiza trei cazuri posibile de evoluţie a stărilor când se afla în starea totală stabilă $z_1z_0x = 000$ şi intrarea se modifică de la 0 la 1. Pentru fiecare caz evoluţia stărilor va fi reprezentată simultan prin trei modalităţi: diagrama de evoluţie în timp a semnalelor, tabelul de evoluţie în timp al stărilor şi tabelul de evoluţie al stărilor(notaţiile S şi I indică pentru starea prezentă atinsă că: este stabilă, respectiv instabilă).

1. $\Delta_1 = \Delta_0 = \Delta$ Figura 3.2-c. (Propagarea semnalelor pe ambele conexiuni de reacţie se realizează în acelaşi timp Δ, iar propagarea pe partea combinaţională se face instantaneu, timp zero). Pentru $z_1(t_0)z_0(t_0) = 00$ şi $x(t_0) = 1$ se calculează, pe baza relaţiilor anterioare, starea următoare $w_1(t_0)w_0(t_0) = 11$. Deci circuitul trece din starea stabilă (00) în starea instabilă 00, $z_1(t_0)z_0(t_0) = 00 \neq w_1(t_0)w_0(t_0) = 11$, şi rămâne în această stare instabilă pe durata Δ după care trece în noua stare prezentă de la momentul $t_0+\Delta$, $z_1(t_0+\Delta)z_0(t_0+\Delta) = 11$. În noua stare prezentă se calculează starea următoare care este $w_1(t_0+\Delta)w_0(t_0+\Delta) = 11$ şi pentru care este îndeplinita condiţia $z_1(t_0+\Delta)z_0(t_0+\Delta) = 11 = w_1(t_0+\Delta)w_0(t_0+\Delta) = 11$, deci noua stare prezentă este o stare stabilă,(11), circuitul ramâne în această stare. Evoluţia a fost starea stabilă (00), starea instabilă 00 şi apoi în starea stabilă (11).

2. $\Delta_0 < \Delta_1$, Figura 3.2-d. Această relaţie arată că bitul w_0 al stării următoare calculate se propagă în bitul z_0 al stării prezente mai repede decât bitul w_1 în z_1. În momentul t_0 când x se modifică din 0 în 1 se trece din starea prezentă stabilă (00) în starea instabilă 00, $z_1(t_0)z_0(t_0) = 00 \neq w_1(t_0)w_0(t_0) = 11$. După durata Δ_0 noua stare prezentă este $z_1(t_0 + \Delta_0)z_0(t_0 + \Delta_1) = 01$ care pentru $x = 1$ generează $w_1(t_0 + \Delta_o)w_0(t_0 + \Delta_0) = 11 \neq 01$, deci şi această stare prezentă 01 este instabilă. În momentul $t_0 + \Delta_1$ şi bitul w_1 se propagă în z_1 iar noua stare prezentă este $z_1(t_0 + \Delta_1)z_0(t_0 + \Delta_1) = 11$ pentru $x = 1$ calculând biţii

stării următoare rezultă $w_1(t_o + \Delta_1)w_0(t + \Delta_1) = 11$, deci starea prezentă (11) este o stare stabilă. Evoluţia a fost starea stabilă (00), trecerea în starea instabilă 00(la momentul t_0), trecerea în starea instabilă 10(la momentul $t_0 + \Delta_0$) şi în final trecerea în starea stabilă (11) (la momentul $t_0 + \Delta_1$).

3. $\Delta_1 < \Delta 0$, Figura 3.2-e. De data aceasta bitul w_1 al stării următoare calculate se propagă în bitul z_1 al stării prezente înaintea propagării bitului w_0 în z_0. Analiza evoluţiei stărilor se realizează similar ca la punctul 2 şi rezultă o succesiune (00) $\rightarrow 00 \rightarrow 10 \rightarrow$ (11).

În concluzie acest circuit din starea totală 000 poate trece în starea totală 001(în funcţie de valorile întârzierilor Δ_1 şi Δ_0) prin una din următoarele trei succesiuni: (00)$\rightarrow 00\rightarrow$ (11) sau (00)$\rightarrow 00\rightarrow 01\rightarrow$ (11) sau (00)$\rightarrow 00\rightarrow 10\rightarrow$ (11). Cursa pentru un circuit este necritică dacă pornind dintr-o stare, pe oricare traseu, se ajunge în final în aceeaşi stare totală stabilă, cum este în cazul analizat. În opoziţie, un circuit prezintă o **cursă critică** dacă din aceeaşi stare totală iniţială se ajunge în stări totale stabile diferite în funcţie de traseul urmat.

Condiţia de cursă critică pentru un circuit asincron poate aparea când două sau mai multe din variabilele binare de stare z_i îşi schimbă valoarea la o modificare în cuvântul de intrare $X(t)$. În consecinţă, printr-o judicioasă codificare a stărilor circuitului, în tabelul de evoluţie al stărilor, se poate realiza un circuit fără curse critice. La un circuit fără curse critice trebuie ca pe traseul de evoluţie al stărilor în fiecare moment de trecere de la o stare instabilă la o altă stare instabilă, să se modifice doar un singur bit în cuvântul stării calculate. Poate exista şi cazul când evoluţia stărilor este pe un traseu ciclic format numai pe stări instabile ceea ce determină o **funcţionare de oscilator** pentru circuit. Dar să analizăm un astfel de regim în Exemplul 3.2. Analiza şi sinteza circuitelor secvenţiale asincrone este destul de dificilă dacă nu se introduc anumite restricţii. În acest sens, în primul rând se admite că în configuraţia de intrare $X(t)$ la un moment dat se modifică doar un singur bit. Şi în al doilea rând, se consideră că o modificare a semnalului de intrare nu are loc până când circuitul nu a ajuns într-o stare stabilă. Un **circuit secvenţial asincron** care respectă aceste două restricţii este referit ca circuit ce funcţionează **în mod fundamental**.

Exemplul 3.2 Să se analizeze funcţionarea circuitelor secvenţiale asincrone realizate numai cu inversoare înseriate.

Soluţie. Se disting două structuri cu număr impar şi număr par de inversoare înseriate.

a) În Figura 3.3-a este prezentat un circuit secvenţial asincron realizat printr-o legătură de reacţie la o poartă NAND2. Pentru intrarea $x = 1$ poarta NAND2 are o funcţionare de inversor $w = \overline{z}$. Din tabelul de evoluţie al stărilor, Figura 3.3-b, se observă că pentru coloana $x = 1$ nu există nici o stare stabilă. Pentru starea prezentă $z = 0$ şi intrarea $x = 1$ se calculează starea următoare $w = 1$ care generează o tranziţie, $w \rightarrow z = 1$, pe linia a doua a tabelului, unde pentru starea prezentă $z = 1$ şi intrarea $x = 1$ se calculează starea următoare $w = 0$ care generează o tranziţie, $w \rightarrow z = 0$, pe linia întâia a tabelului, unde pentru starea prezentă $z = 0$ şi intrarea $x = 1$ se calculează starea următoare $w = 1$ care generează o tranziţie $w \rightarrow z = 1$ pe linia a doua a tabelului şi procesul se repetă, adică trece încontinuu între stările instabile $z = 0$ şi $z = 1$. Pe baza schemei echivalente, ca CLS asincron Huffman, din Figura 3.3-a, se pot reprezenta semnalele z şi w,Figura 3.3-c, considerând inversorul cu timp de transfer zero iar timpii săi de propagare τ_{pHL} şi τ_{pLH} sunt introduşi ca întârzieri pe legătura de reacţie. Pentru $x = 0$ modificările stării prezente z nu se transmit prin poarta

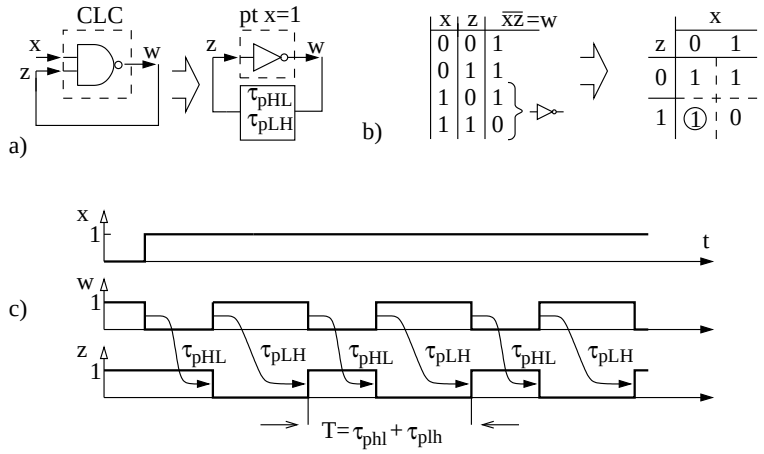


Figura 3.3 CLS asincron realizat prin înserierea unui număr impar de inversoare: a) structură realizată pe baza unei singure porţi NAND2; b) tabelul de evoluţie al stărilor; c) variaţia în timp a semnalelor generate.

NAND2, w este permanent 1, dar pentru $x = 1$ starea prezentă z se transmite prin poartă cu valoarea complementată, $w = \overline{z}$ (vezi tabelul de adevăr). Pentru starea prezentă $z = 1$ şi $x = 1$ se calculează instantaneu starea următoare $w = 0$ care devine stare prezentă $z = 0$, după întârzierea τ_{pHL}, pe baza căruia se calculează noua stare următoare $w = 1$ care devine stare prezentă $z = 1$ după întârzierea τ_{pLH} şi procesul se repetă. Deci circuitul are funcţionare de generator de semnal dreptunghiular cu perioada $T = \tau_{\text{pHL}} + \tau_{\text{pLH}}$. Functia de generator de semnal dreptunghiular o are orice CLS asincron la care partea combinaţională este o înserie a unui număr impar n de inversoare, perioada semnalului generat fiind $T = n(\tau_{\text{pHL}} + \tau_{\text{pLH}})$ vezi Exemplul 1.17.

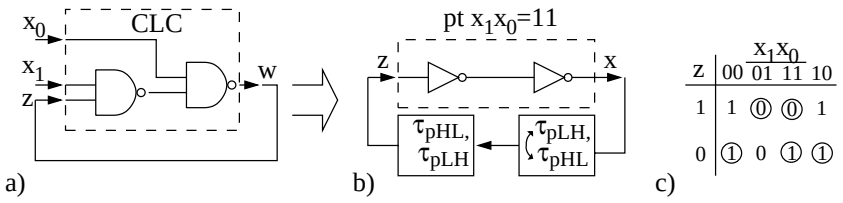


Figura 3.4 CLS asincron realizat prin înserierea unui număr par de inversoare: a) structură realizată pe baza a două porţi NAND2; b) structură echivalentă ca CLS Huffman; c) tabelul de evoluţie al stărilor.

b) În Figura 3.4-a este prezentă o structură de CLS asincron realizat prin înserarea a două porţi NAND2, iar pentru cuvântul de intrare $x_1x_0 = 11$ fiecare poartă este echivalentă cu un inversor, Figura 3.4-b. Funcţia de tranziţie este $w = \overline{\overline{zx_1} \cdot x_0} = zx_1 + \overline{x_0}$ ale cărei valori sunt trecute în tabelul de evoluţie al stărilor din Figura 3.4-c. Pentru coloana $x_1x_0 = 11$ se observă că atât starea $z = 0$ cât şi starea $z = 1$ sunt stabile, adică starea următoare calculată w este identică cu starea prezentă şi se tranferă în stare prezentă după întârzierea $\Delta = \tau_{\text{pHL}} + \tau_{\text{pLH}}$. Din acest tabel se observă că pentru toate combinaţiile de intrare $x_1x_0 = 00, 01, 11, 10$ există stări stabile pentru circuit(o discuţie necesită cazul când $x_1x_0 = 00$, vezi secţiunea 3.3.1).

Se poate generaliza această afirmaţie, un CLS asincron compus prin înserierea unui număr par de inversoare are pentru fiecare cuvânt de intrare cel puţin o stare stabilă.

3.2 CIRCUITE LOGICE SECVENŢIALE SINCRONE

La un circuit logic secvenţial asincron variabila timp "curge" continuu, există incontinuu transfer de la ieşire spre intrare, pe calea de reacţie, deci o evoluţie dintr-o stare în alta(chiar şi când este într-o stare totală stabilă se poate considera că circuitul evoluează incontinuu din acea stare tot în acea stare). Spre deosebire de circuitul asincron, **la un circuit logic secvenţial sincron variabila timp are o "curgere" discretă**, trecerea/tranziţia dintr-o stare în următoarea se face numai în momentele bine definite de către semnalul de ceas/clock. Semnalul de ceas (clock) este un semnal dreptunghiular periodic care comută între L şi H şi invers cu un coeficient de umplere $\leq 50\%$, Figura 3.5, simbolizat în circuite prin abreviaţia CLK. Comutarea repetată între L şi H este utilizată pentru a marca timpul, unitatea de timp fiind perioada T a acestui semnal; un moment pe scara timpului se marchează doar printr-un multiplu de perioadă de clock: $\ldots, iT, (i+1)T, (i+2)T, \ldots$.

Ceasul în circuitele sincrone poate acţiona pentru marcarea timpului prin unul din palierele sale, referit ca **palier activ**. De exemplu, marcarea timpului pe palierul activ H este reprezentată în Figura 3.5-a, iar marcarea timpului cu **frontul activ** crescător/ anterior/ pozitiv sau cu frontul activ descrescător/posterior/negativ este reprezentată respectiv în Figura 3.5-b şi 3.5-c. Marcarea pe palier(=interval) se face într-un mod mai grosier, pe când marcarea pe front este un mod foarte precis.

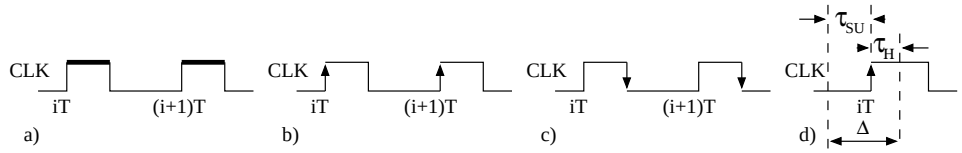


Figura 3.5 Semnale de ceas pentru sincronizare pe: a) pe palier; b,c) pe frontul pozitiv/crecător respectiv negativ/descrescător; d) fereastra de decizie $\Delta = \tau_{SU} + \tau_H$ axată pe frontul de sincronizare, pe durata căruia semnalul de sincronizat nu trebuie să îşi modifice valoare.

3.2.1 Sincronizarea semnalelor asincrone

Înainte de prezentarea CLS sincrone se va analiza modul în care un semnal este sincronizat. Prin sincronizarea unui semnal se înţelege aducerea acelui semnal în "acelaşi timp" cu semnalul de ceas. Aceasta înseamnă că semnalul asincron este citit/eşantionat şi înscris într-un element de memorare numai în momentele discrete dictate de semnalul de ceas. Dar care va fi valoarea citită pentru un semnal asincron

când sincronizarea se face pe palier? dificil de spus deoarece semnalul fiind cu variaţia asincronă poate să îşi modifice valoarea între "0" şi "1" de nenumărate ori pe durata palierului impulsului de ceas; în această situaţie în elementul de memorare se va înscrie valoarea pe care o are semnalul asincron pe frontul de sfârsit al palierului activ. Evident că, pentru a face o citire corectă trebuie să se impună ca semnalul să nu se modifice pe durata palierului. La eşantionarea pe front, consideând că panta acestuia este infintă(durata frontului $\tau_f = 0$), condiţia de a se menţine nemodificat semnalul asincron doar într-un singur punct este mai usor de îndeplinit. Dar şi la sincronizarea pe front se impune ca semnalul asincron să nu îşi modifice valoarea pe un interval de **timp de prestabilire τ_{SU}(set-up)** înainte de front şi apoi pe încă un interval de **timp de menţinere τ_H(hold)** după front, relaţia 3.23; **deci nici o modificare pe lăţimea unei ferestre(de decizie) cu lăţimea $\Delta = \tau_{SU} + \tau_H$ axată pe frontul semnalului de ceas**, Figura 3.5-d. Această restricţie de nemodificare pe intervalul Δ este impusă de către funcţionarea elementului fizic – bistabilul – care realizează citirea şi memorarea semnalului asincron. Bistabilul(a se vedea Figura 3.5-d) la o modificare a semnalului asincron(de sincronizat) în intervalul Δ, poate memora fie valoarea logică 1, fie valoarea logică 0, deci nu se poate controla valoarea care se înscrie(funcţionare nedeterministă). Teoretic, aceste restricţii de nemodificare în momentul eşantionării duc la afirmaţia că un semnal asincron (fiind asincron poate să se modifice oricând) nu poate fi sincronizat corect. Din analiza cazurilor următoare se va vedea când această concluzie este corectă şi când nu.

Se consideră cazul când trei semnale asincrone $A = 0, B = 0, C = 1$ sunt eşantionate simultan dar numai semnalul C îşi modifică valoarea de la 1 la 0 în fereastra de decizie Δ, centrată pe frontul pozitiv de ceas de la momentul iT, Figura 3.6-a. După frontul pozitiv de sincronizare se memorează fie cuvântul $ABC = 001$ fie cuvântul $ABC = 000$, dar ambele cuvinte sunt corecte deoarece oricare dintre acestea sunt generate de către sursa cuvântului ABC. Dacă la frontul iT se obţine cuvântul $ABC = 001$ la următorul front de sincronizare $(i + 1)T$, considerând că semnalul C îşi păstrează nemodificată noua valoare 0, se memorează valoarea corectă, adică $ABC = 000$. Deci necitirea corectă la momentul iT poate fi privită ca o întârziere a sincronizării cu un tact a cuvântului $ABC = 000$.

Cazul când pentru cele trei semnale asincrone, în fereastra de decizie de pe frontul pozitiv al semnalului de ceas de la momentul iT, se modifică două dintre semnale, C de la 1 la 0 şi B de la 0 la 1, este reprezentat în Figura 3.6-b. După frontul pozitiv de sincronizare se pot memora unul din următoarele patru cuvinte $ABC = 001, 010, 000, 011$, iar după al (i+1)T front de sincronizare, dacă nu mai apar modificări, cu întârziere de un tact se obţine cuvântul corect $ABC = 010$. Deci în intervalul dintre iT şi (i+1)T, datorită indeciziei semnalului citit, pot apărea şi cuvintele $ABC = 000$ şi $ABC = 011$ care sunt incorecte deoarece sursa nu a generat aceste cuvinte ci doar cuvintele 001 şi 010. Deci la modificarea a doi sau mai mulţi biţi, sincronizarea nu poate fi sigură.

În concluzie pentru sincronizarea corectă a unui grup(cuvânt) de semnale asincrone este necesară respectarea restricţiei ca la un moment dat să se modifice doar un singur semnal. Cazul particular când aceste semnale se obţin ca un cuvânt de la un CLC cu ieşiri multiple, posibil încărcate cu hazard, pentru o sincronizare corectă este necesar să se aştepte până se consumă timpul de propagare pentru producerea acestor semnale. Dacă pentru un grup de semnale când sunt sincronizate se consideră fiecare

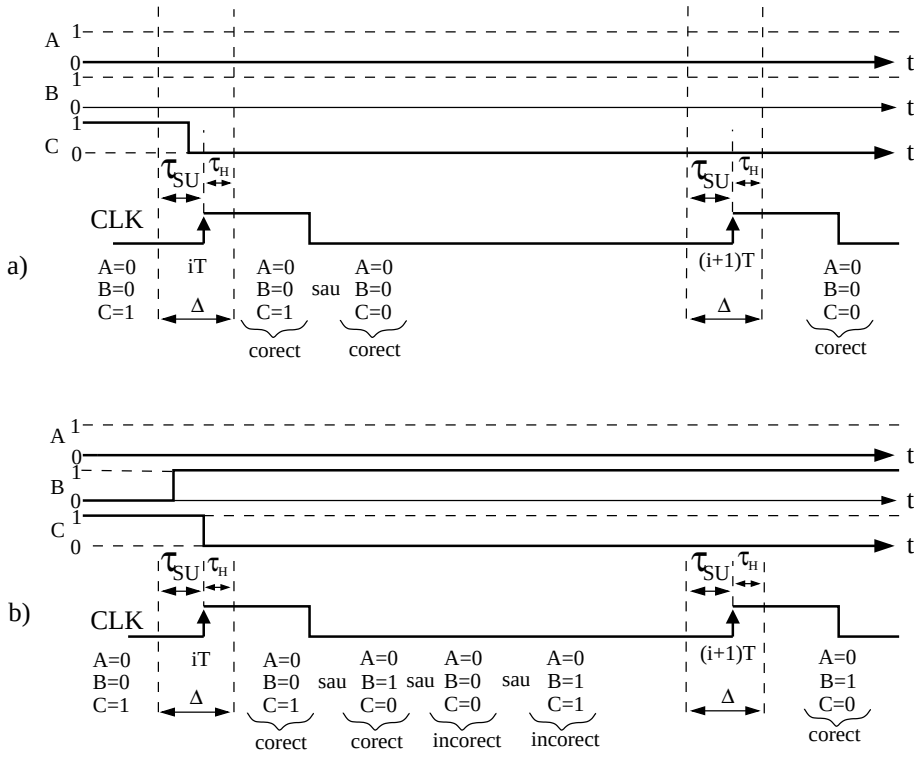


Figura 3.6 Sincronizarea(memorarea) semnalelor asincrone: a) înscrierea corectă a unui cuvânt când se modifică doar un singur bit; b) posibilitatea de înscriere incorectă când se modifică mai mult de un singur bit în fereastra de decizie Δ.

semnal cu semnificaţie individuală, deci nu o semnificaţie în grup/în cuvânt, atunci sincronizarea fiecăruia în parte este corectă în oricare moment de timp. Pe când la un grup de semnale asincrone care se constituie într-un cuvânt de cod, codul exprimat de acel cuvânt depinde de valoarea tuturor semnalelor din acel cuvânt şi nu doar de unul singur, sincronizarea corectă este imposibilă.

3.2.2 Automate finite: structură, definiţii, clasificări

Structura de principiu a unui CLS sincron se obţine din cea a unui CLS asincron, Figura 3.1, în care elementele de întârziere Δ_i sunt substituite cu elemente de memorare/(registru de stare) ca în Figura 3.7-a (Registrul de stare este un circuit registru prezentat în secţiunea 3.5. Registrul este un circuit de memorare în care se poate înscrie un cuvânt binar numai la aplicarea unui semnal de încărcare/înscriere, momentan considerăm că semnalul de încărcare este semnalul de ceas. După înscriere, cuvântul este permanent generat la ieşirea registrului şi poate fi utilizat ca intrare la alte circuite.)

Biţii w_i calculaţi ai funcţiei de excitaţie, cuvântul stării următoare, vor fi înscrişi în registrul de stare numai în momentele de aplicare ale fronturilor active(pozitive)

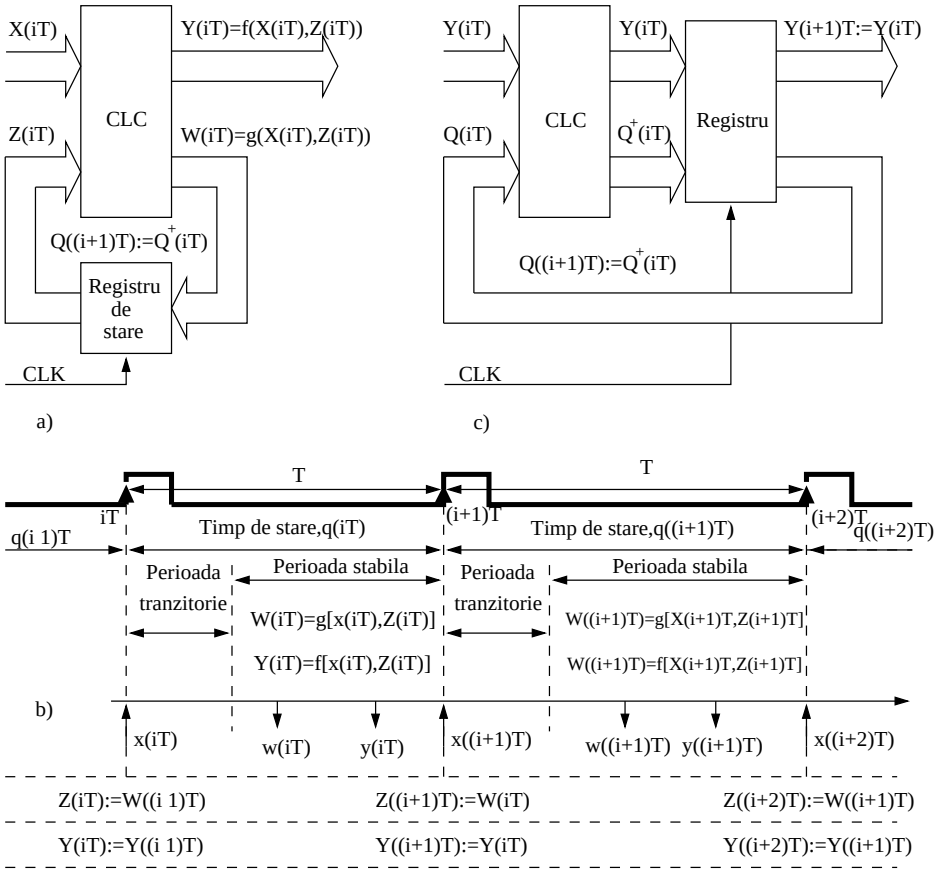


Figura 3.7 Circuitul secvenţial sincron: a) structură de principiu; b) succesiunea în timp a operaţiilor de transfer f şi de tranziţie g; c) structura de principiu pentru CLS sincron cu ieşirea înârziată.

din semnalul de ceas $\ldots, (i-1)T, iT, (i+1)T, \ldots$. Odata înscrişi biţii stării următoare în momentul iT, în registrul de stare, aceştia vor devenii biţii stării prezente $Z(iT)$.

Pe baza stării prezente $Z(iT)$ şi intrării $X(iT)$ (se consideră că intrarea este sincronizată cu acelaşi semnal de ceas ca şi biţii stării prezente) se va calcula transferul intrare ieşire şi funcţia de excitaţie pentru starea următoare,

$$\begin{aligned} Y(iT) &= f[X(iT), Z(iT)] \\ W(iT) &= g[X(iT), Z(iT)] \end{aligned} \tag{3.6}$$

La următorul impuls de ceas $(i+1)T$, biţii calculaţi $W(iT)$ ai stării următoare $q^+(iT)$ sunt înscrişi în registrul de stare şi devin biţii $Z((i+1)T)$ ai stării prezente $q((i+1)T)$. Din nou, pe baza stării prezente $q((i+1)T)$ şi a intrării $X((i+1)T)$ se calculează după relaţiile 3.6, ieşirea $Y((i+1)T)$ şi biţii funcţiei de excitaţie $W((i+1)T)$ care înscrişi în registrul de stare, la momentul $(i+2)T$, devin biţii $Z((i+2)T)$ ai stării prezente $q((i+2)T)$. Se observă că asignarea stării următoare calculate $Q^+(iT)$, reprezentată

prin biţii funcţiei de excitaţie $W(iT)$, în stare prezentă $Q((i+1)T)$ se face numai în momentul $(i+1)T$ prin înscrierea în registrul de stare $Q((i+1)T) := Q^+(iT)$. Numai în momentul aplicării frontului activ al semnalului de ceas, pentru înscrierea în registrul de stare, se poate considera că bucla este închisă în restul intrevalului de timp, până la aplicarea următorului front activ al impulsului de ceas, bucla este deschisă. Deci, **nu există un transfer continuu a stării următoare calculată Q^+ în stare prezentă Q pe legătura de reacţie ca la CLS asincron**. În intervalul de timp dintre două fronturi active ale semnalului de ceas, circuitul poate fi privit ca un circuit combinaţional, care pe baza cuvintelor de intrare $X(iT)$ şi $Z(iT)$ calculează ieşirile $Y(iT)$ şi $W(iT)$. Inexistenţa unui transfer continuu pe conexiunea de reacţie, cu excepţia momentelor aplicării fronturilor active ale semnalelor de ceas, îi conferă CLS-ului asincron o funcţionare de circuit deschis care totdeauna este stabil, adică trece dintr-o stare stabilă în altă stare stabilă.

Transferurile între două impulsuri succesive de ceas, pe partea combinaţională a circuitului secvenţial sincron sunt prezentate în Figura 3.7-b. Din momentul aplicării impulsului de ceas iT se calculează, pe partea combinaţională, $W(iT)$ şi $Y(iT)$, dar biţii acestor configuraţii calculate pot fi afectaţi de hazard datorită parcurgerii unui număr diferit de niveluri logice, deci citirea lor trebuie facută numai după consumarea regimului tranzitoriu. Asignarea stării prezente $Q((i+1)T) := Q^+(iT)$ şi citirea ieşirii $Y(iT)$ sunt corecte numai în perioada de regim stabilizat. Aplicarea următorului impuls de ceas nu trebuie să fie mai devreme de consumarea regimului tranzitoriu, deci frecvenţa maximă a semnalului de ceas cu care se poate comanda CLS-ul se determină cu relaţia:

$$f_{max} = \frac{1}{T_{min}} = \frac{1}{\tau_{\text{pmaxCLC}} + \tau_{\text{pBQ}} + \tau_{SUQ}} \tag{3.7}$$

în care:

τ_{pmaxCLC} - timpul de propagare maxim pe circuitul combinaţional(care determină durata regimului tranzitoriu, afectat de hazard).

τ_{pBQ} - timpul de propagare pe registrul(bistabilul) de stare.

τ_{SUQ} - timpul de prestabilire (figura 3.5-d). Biţii calculaţi w_i ai stării următoare şi aplicaţi pe intrarea registrului de stare trebuie să nu se modifice cu cel puţin τ_{SU} înainte de aplicarea frontului activ de ceas.

De foarte multe ori, la fel ca şi biţii calculaţi $W(iT)$ ai stării următoare şi biţii calculaţi $Y(iT)$ ai cuvântului de ieşire înainte de a fi disponibili la ieşire sunt memoraţi (înscrişi într-un registru de ieşire), Figura 3.7-c. Deci, biţii $Y(iT)$ calculaţi pentru impulsul de ceas iT vor fi disponibili la ieşire la următorul impuls de ceas $(i+1)T$ numai după ce au fost înscrişi în registru, $Y((i+1)T) := Y(iT)$. Un astfel de **CLS** este referit cu **întârziere** în contrast cu cel fără registru care este referit **CLS imediat**.

La un CLS sincron imediat ieşirea depinde de starea curentă. Modificarea intrării se simte la ieşire pe durata stării prezente. Ieşirea poate fi afectată de hazard, deci trebuie citită doar după consumarea regimului tranzitoriu. CLS sincron cu întârziere are ieşirea dependentă de starea anterioară(corespunzătoare ciclului anterior de ceas). Deoarece se presupune că biţii de ieşire au fost înscrişi în registrul de ieşire după

consumarea regimului tranzitoriu, ieşirea întârziată nu este afectată de hazard. Deci la un circuit cu întârziere o modificare a intrării $X(iT)$ se va "simţii" la ieşire numai după frontul activ de ceas $(i+1)T$, adică cu o întârziere de un tact.

Structura de CLS sincron care conţine în bucla de reacţie un circuit de memorare/registru, care la rândul său constituie un CLS, este referită prin termenul de **automat (finit)** sau cu abreviaţia **FSM (Finite State Machine)**. În continuare se vor prezenta succint doar acele definiţii şi clasificări ale automatelor [Creangă '73][Ştefan '00] [Maican '99] care vor fi utilizate în această lucrare.

Definiţia unui automat finit A este similară cu cea data prin relaţia 3.5 pentru un CLS asincron cu deosebirea că timpul nu mai este o variabilă continuă ci discretă, multiplu de perioada semnalului de ceas, iT, adică variabilele circuitului sunt eşantionate dar pe fronturile active

$$A = (X, Y, Q, f, g) \tag{3.8}$$

Definiţia 3.1 Dacă pentru oricare configuraţie de intrare şi de stare prezentă, funcţiile de transfer, f, şi de tranziţie, g, au cardinalul unitate(conţine un singur element), atunci **automatul este determinist**, altfel este nedeterminist $\diamond$

$$\forall q \in Q,\ \forall x \in X,\ |f(q,x)| = 1,\ |g(q,x)| = 1 \tag{3.9}$$

La un automat determinist din starea prezentă şi pentru intrarea aplicată se cunoaşte exact ieşirea precum şi starea pentru tactul următor, pe când la unul nedeterminist, starea următoare şi ieşirea sunt cunoscute doar ca elemente care aparţin $\mathcal{P}^*(Q)$, respectiv $\mathcal{P}^*(Y)$ (vezi relaţia 3.4).

Definiţia 3.2 Un **automat** A **este finit** dacă cele trei mulţimi X, Y şi Q sunt finite. $\diamond$

Este necesar a se evidenţia faptul că un automat cu un număr finit de stări poate prelucra şiruri infinite aplicate pe intrare, definite pe mulţimea finită X, şi poate genera la ieşire şiruri infinite, definite pe mulţimea finită Y.

Definiţia 3.3 Un **automat** A **autonom**, sau orologiu, este un automat la care mulţimea finită a intrărilor are cardinalul unitate $|X| = 1$ $\diamond$

Pentru automatul autonom, mulţimea configuraţiilor de intrare este permanent aceeaşi ceea ce dă automatului o comportare independentă de exterior, vezi Exemplul 3.2. Automatul generează variaţii la ieşire fără să se modifice configuraţia de intrare.

Definiţia 3.4 O **evoluţie/traiectorie a automatului** A este un şir de stări înlănţuite $q_0 \to q_1 \to \ldots \to q_i \to q_{i+1} \to \ldots$, iar starea q_0 este starea iniţială a evoluţiei. O evoluţie cu proprietatea:

$$\forall q_i \in Q,\ \exists X_i \in X,\ q_{i+1} \in g(q_i, X_i) \tag{3.10}$$

este o evoluţie admisibilă $\diamond$

Definiţia 3.5 Totalitatea stărilor din care poate porni automatul este mulţimea stărilor iniţiale $Q_0, Q_0 \subset Q$ $\diamond$

În funcţie de cardinalul mulţimii Q_0 automatele pot fi:

- automat cu o singură stare iniţială $Q_0 \in Q,\ |Q_0| = 1$
- automat cu mai multe stări iniţiale $Q_0 \in Q,\ |Q_0| > 1$
- automat neiniţial(toate stările pot fi iniţiale), $Q_0 = Q$

Într-o stare iniţială nu se ajunge printr-o evoluţie generată de configuraţii din mulţimea semnalelor de intrare X ci printr-o altă comandă exterioară, de exemplu printr-o comandă de iniţializare/reset sau prin conectarea la tensiune se generează starea iniţială.

Definiţia 3.6 O **stare** a automatului este considerată **inaccesibilă** dacă oricare traiectorie în spaţiul stărilor ce porneşte din stări iniţiale nu o conţine. ◇

Evident, se poate vorbi de stări inaccesibile numai în cazul automatelor iniţiale $|Q_0| \geq 1$. De exemplu, dacă pentru un automat cu trei biţi de stare $z_2z_1z_0$ pot exista opt stări $q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8$, dar sunt definite numai cinci stări, $q_0 \div q_4$, în funcţionarea automatului atunci cele trei stări, q_5, q_6, şi q_7 sunt inaccesibile.

Definiţia 3.7 Două **stări** q_i şi q_j sunt **echivalente** $q_i \sim q_j$ dacă pentru oricare configuraţie de intrare aplicată X se generează aceleaşi ieşiri, iar stările următoare $q_{i+1} = g(q_i, X)$, $q_{j+1} = g(q_j, X)$ sunt identice sau echivalente.

$$f(q_i, X) = f(q_j, X);\ g(q_i, X) \sim g(q_j, X) \tag{3.11}$$

◇

În etapa de definire a funcţionării unui automat se poate introduce informaţie redundantă care duce la un număr total de stări mai mare decât numărul de stări care poate descrie funcţionarea automatului. Dacă un grup de stări se demonstrează că sunt echivalente, toate stările acelui grup sunt substituite printr-o singură stare, astfel reducându-se numărul total de stări ale automatului.

Definiţia 3.8 Un **semiautomat,** SA, este definit prin tripletul:

$$SA = (X, Q, g) \tag{3.12}$$

◇

Prin noţiunea de semiautomat se identifică în cadrul unui automat numai acea parte care este responsabilă pentru generarea stării interne. Structural, semiautomatul se obţine prin eliminarea din structura automatului a parţii care calculează funcţia de transfer, f. Pentru cele patru organizări de automate reprezentate în Figura 3.8 sunt delimitate structurile care formează semiautomatul, SA. Semiautomatul este format din partea de circuit combinaţional, CLC1, care calculează funcţia g şi registrul din calea de reacţie. Unui semiautomat i se pot ataşa diferite părţi combinaţionale, CLC2, pentru diferite funcţii f, deci pe baza unui semiautomat se pot realiza mai multe automate. Dar şi unui automat i se pot asocia diferite semiautomate. În proiectarea unui automat se porneşte de la definirea funcţionării automatului adică de la descrierea funcţiei de transfer intrare-ieşire. Deoarece transferul intrare-ieşire este dat, proiectantului îi rămâne "spaţiu de manevră" doar în proiectarea şi optimizarea semiautomatului.

În funcţie de modul cum se calculează funcţia de transfer f a automatului, există două tipuri de automate:Mealy şi Moore.

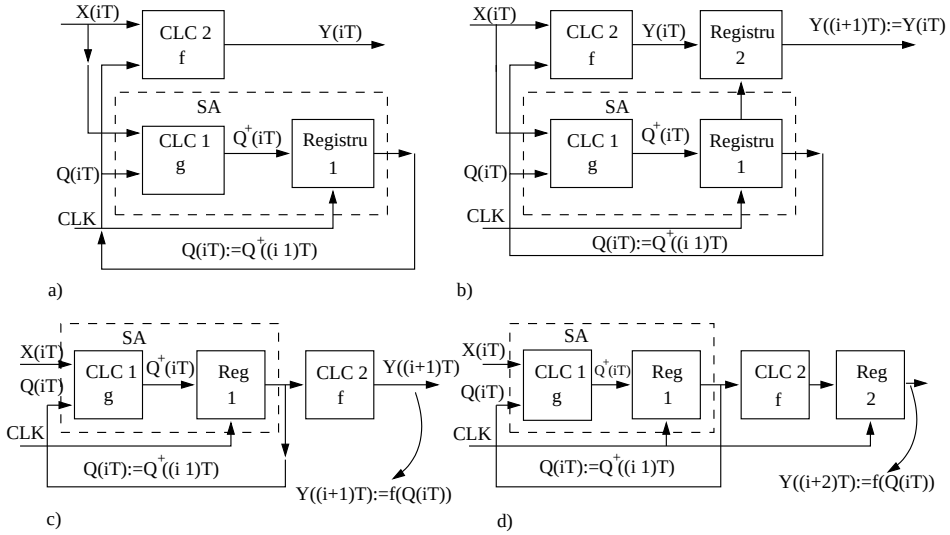


Figura 3.8 Variante de automate Mealy şi Moore structurate pe baza unui semiautomat, SA**:** a,b) structurare fundamentală pentru automatele Mealy imediat şi Mealy cu întârziere; c,d) structurare fundamentală pentru automatele Moore imediat şi Moore cu întârziere.

Definiţia 3.9 Un **automat Mealy** este definit prin cvintuplu A=(X,Y,Q,f,g) cu:

$$\begin{aligned} f &: X_{\times}Q \to Y \\ g &: X_{\times}Q \to Q \end{aligned} \tag{3.13}$$

◇

La automatul Mealy parte combinaţională calculează atât funcţia de transfer f cât şi cea de tranziţie g pe baza produsului cartezian $X_{\times}Q$.

Definiţia 3.10 Un **automat Moore** este definit prin cvintuplu A(X,Y,Q,f,g) cu:

$$\begin{aligned} f &: Q \to Y \\ g &: X_{\times}Q \to Q \end{aligned} \tag{3.14}$$

◇

La automatul Moore funcţia de tranziţie g se calculează la fel ca la automatul Mealy, pe baza produsului cartezian $X_{\times}Q$, pe când funcţia de transfer f este calculată doar pe baza stării prezente Q. Ieşirea automatului Moore este dependentă de intrare doar prin intermediul tranziţiilor realizate în spaţiul stărilor.

Fiecare din aceste două tipuri de automate poate fi realizat cu funcţionare imediată sau întârziată, Figura 3.8.

Automatul Mealy imediat, Figura 3.8-a. Se observă că structura acestui automat se obţine prin completarea semiautomatului SA cu CLC2 care calculează ieşirea

$Y(iT)$ pe baza produsului cartezian dintre starea prezentă şi intrarea prezentă

$$Y(iT) = f(X(iT), q(iT))$$

Ieşirea $Y(iT)$ citită înainte de consumarea perioadei de regim tranzitoriu va fi afectată de hazard. Mai mult, dacă intrarea nu este sincronizată,cum am considerat, ci se modifică asincron oricând în intervalul iT şi $(i+1)T$, aceste modificări se simt în ieşire.

Automatul Mealy cu înârziere, Figura 3.8-b. Completarea SA se face atât cu circuitul CLC2 pentru calculul funcţiei de transfer, f, cât şi cu un registru, Reg2, pentru memorarea ieşirii $Y(iT)$. Introducerea unui registru va face ca modificările provocate în ieşirea $Y(iT)$ pe intervalul $iT-(i+1)T$, datorate regimului tranzitoriu, de la o intrare sincronizată, sau datorate variaţiilor intrării, de la o intrare asincronă, sa nu mai poata ajunge la ieşire. Valoarea ieşirii este asignată, doar în momentele de aplicare a frontului activ de ceas la Reg2, cu valoarea calculată pe baza intrării şi stării anterioare

$$Y((i+1)T) := f(X(iT), q(iT))$$

deci ieşirea reflectă intrarea cu o întârziere de un tact.

Automatul Moore imediat, Figura 3.8-c. Deoarece la acest automat ieşirea se calculează ca funcţie numai de stare prezentă, relaţia 3.14, la SA se adaugă circuitul CLC2 care are ca intrări doar starea prezentă. Deoarece în starea prezentă se reflectă intrarea anterioară şi nu cea prezentă, ieşirea este întârziată cu un tact faţă de intrare(la fel ca la automatul Mealy cu întârziere). Deci pentru intrarea prezentă se obţine ieşirea numai peste un tact.

$$Y((i+1)T) := f(q((i+1)T)) = f(g(X(iT), q(iT)))$$

Deşi se obţine o izolare a ieşirii faţă de intrare, nu există doar CLC2 intre intrare şi ieşire, totuşi automatul Moore imediat poate genera hazard pe ieşire. Hazardul este generat de faptul că circuitul CLC2 poate avea de la intrarea(ieşirea Reg1) până la ieşirea sa un număr diferit de niveluri logice pentru fiecare traseu de transfer. Foarte frecvent, la acest tip de automat se elimină circuitul CLC2, ieşirea fiind identică cu starea

$$Y \equiv Q \tag{3.15}$$

ceea ce elimină hazardul combinaţional din ieşire.
Comportamental, un automat Moore imediat poate fi echivalent cu un automat Mealy cu întârziere deoarece la ambele ieşirea calculată pe baza stării prezente şi a intrării prezente va fi asignată la ieşire dar pe următorul front activ de ceas.

Automatul Moore cu întârziere, Figura 3.8-d. Pentru a se elimina posibilitatea de hazard pe ieşire de la automatul Moore imediat se adaugă pe ieşire un registru, Reg2, obţinându-se automatul Moore cu întârziere. La întârzierea ieşirii faţă de intrare cu un tact de la Moore imediat, de data aceasta se mai introduce încă o întârziere de un tact, deci automatul Moore cu întârziere "simte" o intrare transferată la ieşire cu o întârziere de doua tacturi.

$$Y((i+2)T) := f(q((i+1)T)) = f(g(X(iT), q(iT)))$$

O Clasificare Neclasică a Circuitelor Digitale. Clasificarea clasică a circuitelor digitale în două mare clase: circuite logice combinaţionale, CLC şi circuite logice secvenţiale CLS este foarte "încăpătoare", prezentând o puternică lipsă de nuanţare. În fiecare din cele două clase sunt cuprinse atât circuite cu o definiţie simplă cât şi circuite cu o definiţie complexă, atât circuite nestructurate cât şi circuite puternic structurate. Mai mult, dacă pentru circuitele combinaţionale se poate face o mapare directă între ce vrem să facem **(funcţie)** şi cum vrem să facem **(structură)**, **la circuitele secvenţiale această mapare structură–funcţie nu totdeauna este biunivocă**. O tendinţa de structurare a circuitelor a existat permanent în concepţia inginerească, atât pentru a realiza un suport de cuprindere şi înţelegere dar şi pentru modalităţile de implementare dictate de tehnologie.

În evoluţia sistemelor în natură dar şi în abordarea mentală a acestora este evident saltul care apare la trecerea de la sistemul deschis la sistemul închis(cu reacţie), adică introducerea conexiunii de reacţie. Un sistem cu reacţie în raport cu cel deschis(fără reacţie) poate căpăta o anumită **autonomie faţă de intrare**, o anumită adaptabilitate. În funcţie de modul cum se structurează sistemul, bazându-se pe reacţie/reacţii, acesta poate deveni chiar complet autonom faţă de intrare(vezi Definiţia 3.3), iar spectrul funcţiilor realizate poate fi extins.

Bazat pe noţiunea de reacţie, în extensiile structurate, profesorul Gh.Ştefan [Ştefan '91] a postulat o abordare de clasificare pentru sistemele digitale care este redată în continuare.

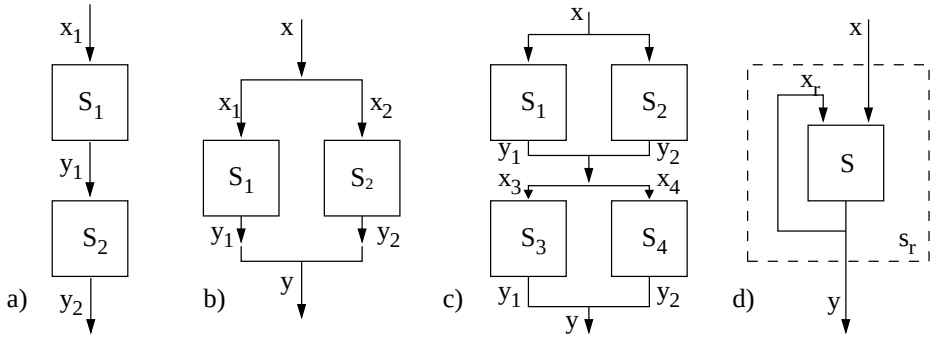


Figura 3.9 Procedee pentru structurarea sistemelor digitale: a) extensia serie; b) extensia paralel; c) extensia serie-paralel; d) extensia prin închiderea unei bucle de reacţie.

1. Extensia sistemelor digitale se face prin următoarele procedee:

   a - extensia serie, Figura 3.9-a

   b - extensia paralel, Figura 3.9-b

   c - estensia serie-paralel, Figura 3.9-c

   d - extensia prin introducerea unei bucle de reacţie, Figura 3.9-d

2. Sistemul de ordinul zero, $SO-0$, este circuitul combinaţional, deoarece nu are nici o buclă de reacţie. Ordinul unui sistem poate fi de la 0 la n, respectiv notat

prin($SO-0$, $SO-n$) corespunzător numărului de la 0 la n al buclelor de reacţie care se includ una în alta.

3. Extensiile de tip a, b, c, ale unor sisteme ce au un ordin egal cu n, generează tot un sistem de ordin n.

4. Extensia de tip d, (printr-o conexiune de reacţie) într-o reţea de sisteme ce conţin cel puţin un sistem de ordin n, generează un sitem de ordin $n+1$. De exemplu, în această secţiune s-a introdus noţiunea de automat care se încadrează în $SO-2$ deoarece cuprinde registrul de stare, care are o reacţie în structura sa, deci este un $SO-1$, peste care se închide încă o buclă(cea care transformă starea următoare calculată în stare prezentă), deci două bucle de reacţie.

Utilizând această abordare de clasificare, implicit, se înţelege că sistemele de ordin superior au o mai evidentă slăbire a corespondenţei biunivoce, structură-funcţie, în raport cu cele de ordin mai redus, pentru că includ mai multe conexiuni de reacţie.

În această lucrare se utilizează totuşi, clasificarea clasică de circuite combinaţionale şi circuite secvenţiale, dar în cadrul circuitelor secvenţiale pentru fiecare grup de circuite se va indica şi ordinul în care se încadrează conform postulării anterioare.

3.2.3 Modalităţi de reprezentare ale automatelor

În capitolul doi, după definirea CLS, s-au prezentat modalităţi de reprezentare ale funcţiei de transfer intrare-ieşire, sectiunea 2.2. Şi în acest capitol, după fundamentarea structurării şi funcţionării unui CLS, se vor prezenta modalităţi de reprezentare ale celor două funcţii f şi g. De data aceasta sunt mai numeroase modalităţile de reprezentare, dintre acestea vor fi expuse doar: Graficul de tranziţie al stărilor, Tabelul de tranziţie al stărilor, Diagrama de variaţie în timp a semnalelor, Organigrama ASM.

Evident că se va alege acea modalitate de reprezentare care duce la o măsură a complexităţii cât mai simplă. Complexitatea, după cum s-a arătat în Definiţia 2.1, poate fi apreciată printr-o mărime asociată dimensiunii definiţiei. O variantă de estimare a complexităţii unui CLS ar fi prin numărul total de biţi în produsul cartezian $X \times Q$. Pentru un automat cu n intrări principale, k intrări secundare şi m ieşiri se estimează o masură a complexităţii aparente $CA_A(n,k) \in O((n+k)2^{n+k})$; 2^{n+k} combinaţii(cuvinte) de intrare, fiecare cuvânt cu lungimea de $(n+k)$ biţi), deci o valoare foarte ridicată. S-ar putea estima complexitatea şi numai după numărul de ieşiri $CA_A(m) \in O(m)$. Din aceste două exprimări rezultă, evident, că un automat este cu atât mai simplu de definit cu cât produsul $X \times Q$ este de dimensiune mai mică. Dar această complexitate aparentă trebuie interpretată ca limită superioară deoarece complexitatea reală poate fi mult mai mică. Complexitatea reală poate fi mult mai mică pentru că în funcţionarea automatului se poate identifica o anumită ordine a ascunsă, anumite regularităţi în structurare ceea ce în final determină o simplificare în definiţie. Oricum, în implementarea unui automat se preferă o creştere a dimensiunii în contul unei scăderi a complexităţii. Un sistem de complexitate scăzută şi dimensiune ridicată poate fi implementat, pe când implementarea unui sistem de complexitate foarte ridicată, fie şi de dimensiune scăzută, poate fi greu implementabil.

3.2.3.1 Graful de tranziţie al stărilor

Pentru un CLS sincron graful de tranziţie al stărilor **(diagrama de tranziţie a stărilor)** este o reprezentare grafică a funcţiei de tranziţie a stărilor, g, şi a funcţiei de transfer intrare-ieşire, f, pentru toate configuraţiile de intrare X. În acest graf fiecărui nod (cerculeţ) îi este asignată o stare, iar fiecare arc orientat între cele două noduri reprezintă tranziţia între cele două stări. Pentru un automat Mealy pe un arc de tranziţie se notează expresia logică a variabilelor de intrare(expresia logică a tranziţiei) sau configuraţia valorilor variabilelor de intrare pentru care această expresie este adevarată, care va provoca tranziţia respectivă, precum şi valoarea configuraţiei de ieşire Y generată de această tranziţie. Evident, tranziţia stărilor pe acel arc se realizează numai când expresia logică a tranziţiei scrisă pe acel arc este adevarată. Pentru un automat Moore, deoarece ieşirile sunt funcţii doar de starea prezentă, $f : Q \rightarrow Y$, pe arce nu sunt trecute configuraţiile de ieşire Y generate de tranziţiile respective, configuraţia de ieşire este înscrisă în cerculeţul stării prezente din care se generează ieşirea respectivă, iar pe arce se înscrie doar expresia logică a tranziţiei sau configuraţia valorilor variabilelor de intrare pentru care expresia logică a tranziţiei este adevarată.

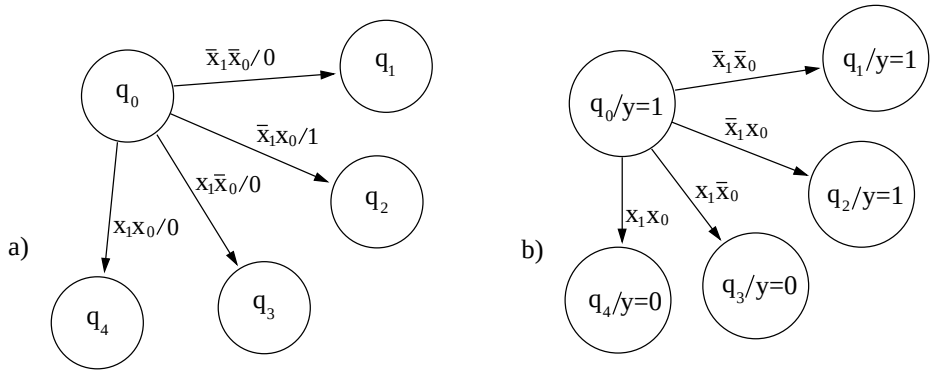


Figura 3.10 Graful de tranziţie al stărilor: a) segment de graf pentru automat de tip Mealy; b) segment de graf pentru automat de tip Moore.

La un automat, cu două intrări x_1x_0 şi o ieşire y, tranziţia din starea q_0 în una din următoarele patru stări posibile q_1, q_2, q_3, q_4, graful de tranziţie este reprezentat în Figura 3.10-a, pentru o funcţionare de tip Mealy şi în Figura 3.10-b, pentru o funcţionare de tip Moore. La graful de tip Mealy în noduri sunt înscrise numai stările, iar pe fiecare arc este trecută expresia logică a tranziţiei/valoarea configuraţiei de ieşire; la graful de tip Moore în noduri, pe lângă stare este înscrisă şi configuraţia de ieşire, iar pe fiecare arc este trecută doar expresia logică a tranziţiei. Dacă la impulsul de ceas iT automatul ajunge în starea q_0 la cel de tip Moore se generează ieşirea $y = 1$, iar la cel de tip Mealy în această stare q_0 se generează $y = 1$ numai dacă configuraţia de intrare este $01(\overline{x}x_0 = 1)$ pentru celelalte trei configuraţii (00,10,11) se generează $y = 0$. La următorul impuls de ceas $(i + 1)T$ se trece în starea următoare pentru care expresia logică a variabilelor de intrare este adevărată, de exemplu pentru $10(x_1\overline{x_0} = 1)$ se trece în starea q_3 (unde $y = 0$ pentru automatul Moore, valoarea

pentru y la automatul Mealy nu este precizată deoarece din această stare, în acest caz, nu mai este figurat nici un arc de tranziţie la o altă stare).

Exemplul 3.3 Un automat cu două intrări x_1, x_0 şi o singură ieşire y, generează ieşirea egală cu 1 numai când din sirul de succesiuni aplicate pe intrare se identifică secventă $00, 00, 11, 10$. Pentru acest automat să se deseneze graful de tranziţie al stărilor atât pentru o funcţionare de tip Mealy cât şi pentru o funcţionare de tip Moore.

Soluţie. În general, pentru aceste automate de recunoaştere, care pot fi considerate chei electronice, în elaborarea grafului de tranziţie se urmăreşte realizarea unei succesiuni de stări care corespund aplicării pe intrare tocmai a succesiunii căutate. Se porneşte dintr-o stare q_0, considerată starea iniţială.

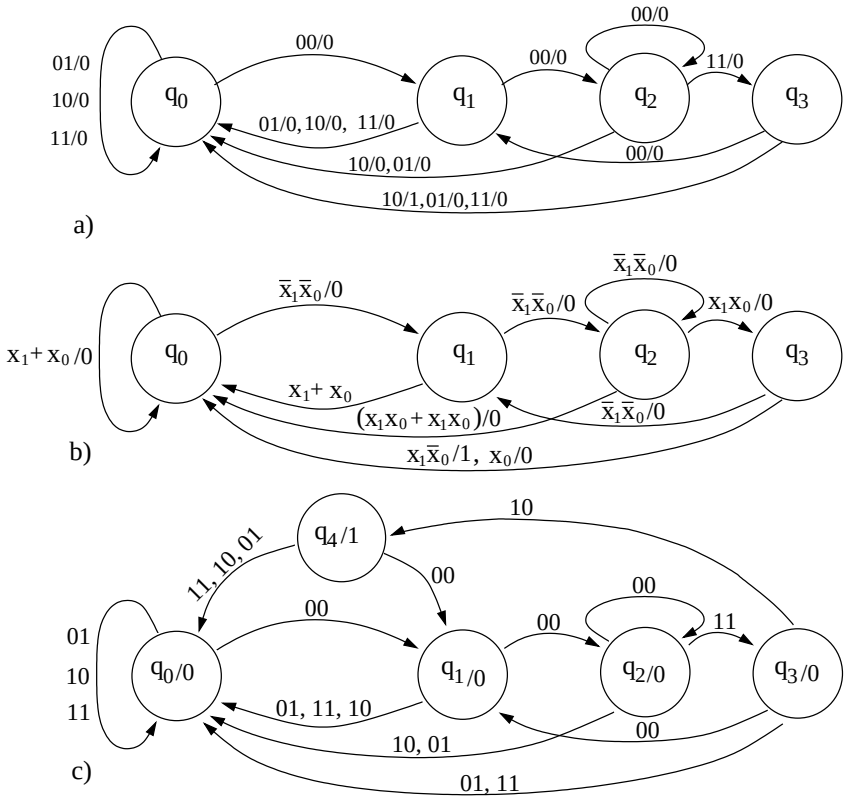


Figura 3.11 Grafuri de tranziţie pentru automatul care identifică succesiunea 00 00 11 10**:** a) pentru un model Mealy cu marcarea pe arce a configuraţiilor de intrare şi ieşire; b) pentru un model Mealy dar cu înscrierea pe arce a expresiei logice a funcţiei de tranziţie; c) pentru un model Moore.

Pentru varianta de automat Mealy numai configuraţia $x_1x_0 = 00$ corespunde începutului identificării succesiunii impuse, deci pentru aceasta se realizează o tranziţie în starea q_1, generându-se ieşirea $y = 0$ iar pentru celelalte trei configuraţii pe intrare posibile se ramâne tot în q_0 şi $y = 0$, Figura 3.11-a. La fel, în q_1 numai configuraţia $x_1x_0 = 00$ produce tranziţia în q_2 şi generează $y = 0$, celelalte trei configuraţii produc tranziţia în q_0 de unde se poate începe de la capăt identificarea succesiunii impuse. În q_2 intrarea potrivită este $x_1x_0 = 11$

care determină tranziţia în q_3 şi generează $y = 0$; intrarea $x_1x_0 = 00$ poate fi considerată ca o a doua combinaţie de 00 din succesiunea impusă, după cea care a produs tranziţia dintre q_1 şi q_2, deci determină o tranziţie tot în q_2. Celelalte doua configuraţii 10 şi 01 nu se potrivesc succesiunii impuse, deci provoacă o tranziţie în q_0 pentru o reluare a identificării de la început. În starea q_3 configuraţia de intrare potrivită este 10 care încheie identificarea cu succes a succesiunii impuse şi care realizează o tranziţie în q_1, pentru o eventuală nouă identificare, şi se generează $y = 1$. Dacă în q_3 configuraţia aplicată este 00 aceasta este considerată ca prima configuraţie din succesiunea impusă deci salt în q_1 şi continuarea identificării. Configuraţiile 01 şi 11 aplicate în q_3 nu se potrivesc succesiunii impuse, deci salt în q_0 cu $y = 0$ pentru, eventual, o nouă identificare. Acelaşi graf de tranziţie al stărilor este prezentat în Figura 3.11-b, dar de data aceasta pe arce, în locul configuraţiilor valorilor de intrare, sunt înscrise expresiile logice ale variabilelor de intrare care determină tranziţiile respective.

Graful de tranziţie al stărilor pentru automat în varianta Moore, Figura 3.11-c, se construieşte exact, ca şi pentru varianta Mealy, până în starea q_3. Din starea q_3 la aplicarea configuraţiei 10, care încheie cu succes identificarea succesiunii impuse, este necesară o tranziţie în starea q_4 în care se generează ieşirea $y = 1$(ieşire dependentă doar de stare). Din starea q_4 pentru configuraţia 00 se realizează o tranziţie în q_1(această configuraţie 00 este considerată ca fiind prima din succesiunea impusă, deci se reia procesul de identificare), iar pentru $11, 10, 01$ salt în q_0 de unde se poate relua de la început procesul de cautare.

Construcţia unui graf de tranziţie al stărilor trebuie să fie fără ambiguităţi, ceea ce impune pentru expresiile de tranziţie înscrise pe arcele care pornesc dintr-o stare să fie mutual exclusive şi toate incluse.

Tranziţiile sunt mutual exclusive dacă, dintr-o stare, pentru aceeaşi configuraţie a valorilor de intrare nu există simultan transferuri la două s-au mai multe stări(există transfer pe un arc numai atunci când expresia respectivă de transfer are valoarea 1). Se poate verifica dacă dintr-o stare nu există transferuri simultane, efectuând produsul logic între expresiile de transfer de pe câte două arce de transfer care pornesc din starea respectivă, aceste produse trebuie să aibă totdeauna valoarea zero pentru oricare configuraţie a valorilor variabilelor de intrare. Pentru n arce care pornesc dintr-o stare numărul total de perechi pentru care se face produsul expresiilor de transfer este egal cu $n(n-1)/2$. Configuraţiile valorilor de intrare testate, pentru care produsul logic al expresiilor de transfer de pe două arce are valoarea 1, sunt referite ca fiind dublu acoperite (de două transferuri).

Tranziţiile sunt toate incluse dacă suma logică a expresiilor de pe arcele care pornesc dintr-o stare este totdeauna egală cu 1 pentru oricare configuraţie a valorilor variabilelor de intrare testate; configuraţiile de valori ale variabilelor de intrare testate pentru care această sumă logică este egală cu zero sunt referite prin configuraţii neacoperite. Cu ajutorul unei diagrame V-K, de toate variabilele care se testează în starea respectivă, se pot determina configuraţiile neacoperite precum şi configuraţiile dublu acoperite.

Pentru aceeaşi funcţionare o implementare sub formă de automat Moore necesită mai multe stări decât implementarea sub formă de automat Mealy, ceea ce s-a văzut şi din Exemplul 3.3. În general, modelul Mealy este uzual pentru circuitele secvenţiale sincrone, iar modelul Moore pentru circuitele asincrone. Graful de tranziţie al stărilor/ieşirilor este un instrument portrivit pentru verificarea funcţionării automatului.

3.2.3.2 Tabelul de tranziţie al stărilor

Informaţia conţinută în graful de tranziţie al stărilor/ieşirilor poate fi transpusă într-o formă mult mai practică pentru sinteza automatului – tabelul de tranziţie al stărilor(şi ieşirilor). În tabelul de tranziţie al stărilor există atâtea linii câte stări destincte prezintă automatul şi atâtea coloane câte cuvinte diferite se pot aplica la intrare. Un element al tabelului tranziţiilor, aflat la intersecţia unei linii cu o coloană, va reprezenta starea următoare/ieşirea a automatului obţinută la tranziţia din starea prezentă corespunzătoare acelei linii prin aplicarea la intrare a cuvântului de pe acea coloană. Uneori, elementele tabelului cuprind numai starea următoare, pentru ieşiri se întocmeşte un tabel de aceeaşi formă ale cărui elemente sunt ieşirile corespunzătoare – denumit tabelul ieşirilor. Tabelul ieşirilor pentru automatul Moore se reduce doar la o singură coloană, având atâtea elemente câte stări distincte există(ieşirile sunt funcţii numai de starea prezentă).

Tabelul de tranziţie al stărilor, care este o reprezentare abstractă a automatului, permite o comparare sistematică a stărilor pentru identificarea perechilor de stări echivalente(vezi Definiţia 3.7)

Exemplul 3.4 Pentru automatul reprezentat în cele două modele Mealy şi Moore, în Figura 3.11, să se construiască tabelele de tranziţie ale stărilor/ieşirilor

Modelul Mealy

Starea prezenta	Starea urmatoare: Intrari x_1x_0 = 00	01	11	10	Iesirea y: Intrari x_1x_0 = 00	01	11	10
q_0	q_1	q_0	q_0	q_0	0	0	0	0
q_1	q_2	q_0	q_0	q_0	0	0	0	0
q_2	q_2	q_0	q_3	q_0	0	0	0	0
q_3	q_1	q_0	q_0	q_0	0	0	0	1

a)

Modelul Moore

Starea prezenta	Starea urmatoare: Intrari x_1x_0 = 00	01	11	10	Iesire y
q_0	q_1	q_0	q_0	q_0	0
q_1	q_2	q_0	q_0	q_0	0
q_2	q_2	q_0	q_3	q_0	0
q_3	q_1	q_0	q_0	q_4	0
q_4	q_1	q_0	q_0	q_0	1

b)

Figura 3.12 Tabelele de tranziţie/ieşire pentru modelul Mealy(a) şi Moore(b) ale automatului cu grafurile de tranziţie din figura 3.11.

Soluţie. Tabelul de tranziţie pentru modelul Moore, Figura 3.12-a, prezintă patru linii corespunzătoare celor patru stări q_0, q_1, q_2, q_3 şi patru coloane câte una pentru fiecare cuvânt de intrare $00, 01, 11, 10$; la fel se construieşte şi tabelul ieşirilor. Pentru modelul Moore, Figura 3.12-b, tabelul stărilor va avea cinci linii, deoarece există o stare în plus faţa de modelul Mealy, în schimb tabelul ieşirilor se reduce la o singură coloană.

3.2.3.3 Diagrame de variaţie în timp ale semnalelor

Uneori funcţionarea automatului poate fi descrisă prin variaţia în timp a semnalelor de intrare, de stare şi de ieşire. Deşi sugestivă, pentru dinamica automatului, această modalitate devine foarte "stufoasă" când numărul acestor semnale este mare. Diagrama de semnale este protrivită în general la automatele generatoare care depind minimal de intrare sau chiar de loc(funcţionare autonomă). Pornind de la diagrama de semnale, în care se identifică stările automatului, se poate deduce tabelul de tranziţie al stărilor/ieşirilor şi apoi se poate construi graful de tranziţie.

Exemplul 3.5 Pentru automatul a cărui funcţionare este prezentată prin variaţia în timp a semnalelor din Figura 3.13-a, să se deducă graful de tranziţie al stărilor/ieşirilor.

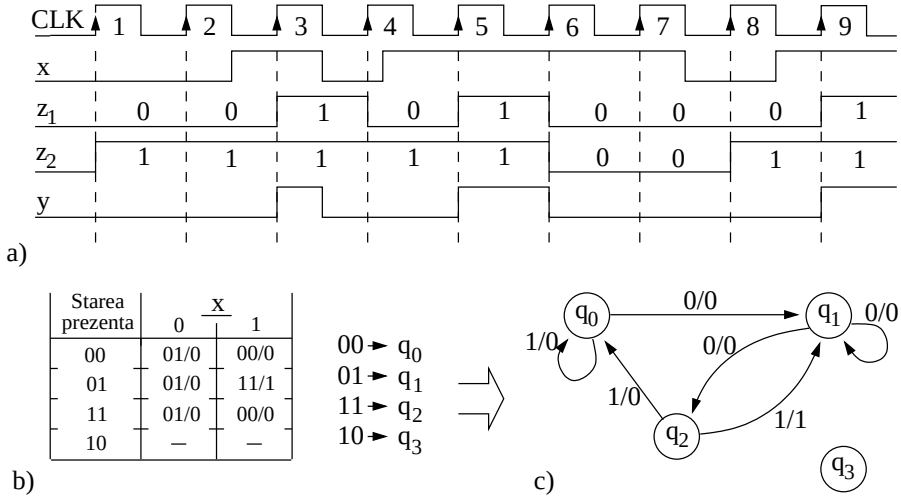


Figura 3.13 Descrierea funcţionării automatului prin diagrame de variaţie în timp ale semnalelor(a): b) tabelul de tranziţie dedus din diagramele de semnal; c) graful de tranziţie al stărilor/ieşirilor.

Soluţie. Automatul prezintă o singura intrare asincronă x, o ieşire y şi două intrări de stare, z_1, z_0, care pot realiza următoarele patru cuvinte de stare:$00, 01, 11, 10$. Din diagrama de semnale la fiecare front activ de ceas pentru valoarea intrării şi a cuvântului de stare prezentă se poate deduce valoarea cuvântului stării următoare precum şi valoare ieşirii; deci pe baza acestor informaţii se poate completa tabelul de tranziţie al stărilor/ieşirilor din Figura 3.13-b. Considerând pentru fiecare din cele patru valori de cuvânt câte o stare $00 \rightarrow q_0, 01 \rightarrow q_1, 11 \rightarrow q_2, 10 \rightarrow q_3$ se obţine graful de tranziţie al stărilor/ieşirilor din Figura 3.13-c. Se observă că starea q_3 este indiferentă, automatul nu va realiza niciodată, la o funcţionare normală, starea q_3

3.2.3.4 Organigrama ASM

Pentru "stăpânirea" sistemelor digitale atât din punct de vedere al descrierii (complexităţii) cât şi din punct de vedere al implmentării(dimensiunii) foarte frecvent se

apelează la segregarea acestuia("divide et impera"). În acest sens structurile digitale pentru procesare fluxurilor de date sunt separate într-o cale de date şi o cale de control, Figura 3.14. Calea de date operează asupra datelor, pentru a realiza o anumită procesare conform unui algoritm, şi este compusă din elemente combinaţionale şi secvenţiale: sumatoare, decodificatoare, multiplexoare, numărătoare, registre, etc, vezi secţiunea 2.5.5.1. Calea de control controlează secvenţialitatea algoritmului de procesare realizat în calea de date. Structurat într-o astfel de manieră circuitul digital mai este referit şi ca **maşina cu stări algoritmice, ASM**(**A**lgorithmic **S**tate **M**achine), denumire corectă deoarece implementează în hardware un algoritm de procesare date. Fluxurile de date sunt aplicate la intrarea în calea de date, unde supuse anumitor secvenţe de procesare corespunzătoare unui algoritm, şi se generează la ieşire datele procesate. Calea de control pe baza unor comenzi aplicate din exterior şi a unor condiţii citite din calea de date, generează semnale de control care dirijează secvenţele de procesare din calea de date. Calea de control are o structură de automat finit.

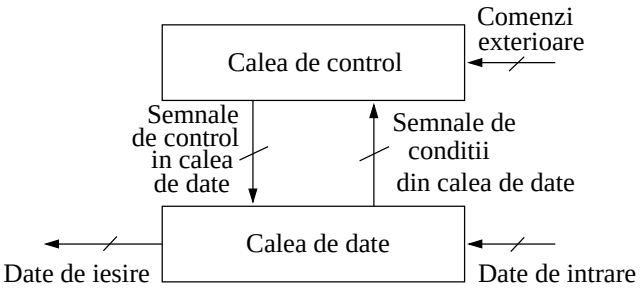


Figura 3.14 Structurarea unui sistem digital într-o cale de date şi o cale de control.

Descrierea fără ambiguitate a funcţionării unui ASM printr-un graf de tranziţie al stărilor/ieşirilor poate deveni destul de greoaie şi cu dificultăţi. Pentru evitarea acestor dificultăţi s-a preluat, cu unele modificări pentru descrierea masinilor hardware algoritmice, ASM, modalitatea de descriere a algoritmilor utilizată în dezvoltarea de soft, adică organigrama(flow chart). Astfel s-a obţinut o organigramă pentru descrierile hardware referita prin organigrama ASM. Simbolurile grafice utilizate în construcţia unei organigrame ASM sunt următoarele trei:

1. Dreptunghiul, Figura 3.15-a, repreznţă o stare a automatului. Prezintă o singură intrare, o singură ieşire, în partea stângă se plasează încercuit numele stării, în colţul dreapta sus codul stării (un cuvânt binar), iar în interior se înscriu ieşirile care devin active când automatul este în această stare. Deoarece ieşirea înscrisă devine activă totdeauna când automatul este în starea respectivă, indiferent de valoarea intrărilor, această ieşire corespunde unei ieşiri de tip Moore intr-un graf de tranziţii(**ieşire necondiţionată**)

2. Rombul, Figura 3.15-b, reprezintă elementul de decizie binar. În interiorul său se înscrie variabila de intrare/expresii de variabile de intrare care se testează.

3. Dreptunghiul rotunjit, Figura 3.15-c, este simbolul pentru **ieşire condiţionată**, corespunde unei ieşiri de tip Mealy intr-un graf de tranziţii. Intrarea într-un

dreptunghi rotunjit este intotdeauna ieşirea de la un element de decizie în care se testează condiţia îndeplinită de anumite variabile de intrare pentru care ieşirea înscrisă în dreptunghiul rotunjit este îndeplinită. Ieşirea acestui simbol poate fi aplicată la un alt romb sau la o stare următoare.

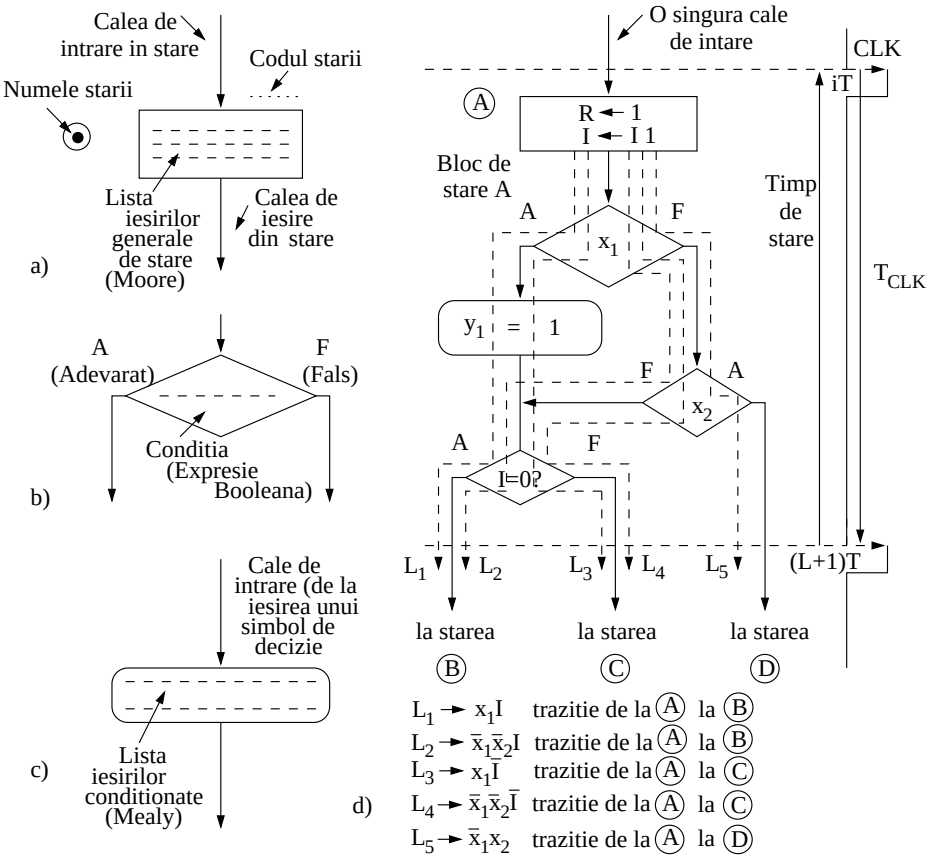


Figura 3.15 Elemente componente ale unei organigrame ASM: a) simbolul pentru stare; b) simbolul pentru decizie; c) simbolul pentru ieşire condiţionată; d) structură pentru un bloc de stare.

Aceste elemente pot fi combinate impreună pentru a forma celula de bază a unei organigrame *ASM*-numită **blocul de stare**, Figura 3.15-d. La un bloc de stare este obligatoriu prezenţa unei stări(dreptunghi), poate exista şi o reţea de simboluri de decizie şi de simboluri de ieşire condiţionată. Într-un bloc de stare există doar o singură cale de intrare, dar pot exista mai multe căi de tranziţie înspre alte blocuri de stare.

Diferenţa principală între un bloc de stare şi o organigramă pentru un program este modul cum se interpretează timpul; la o organigramă operaţiile sunt realizate succesiv în timp, una după alta, pe când într-un bloc de stare, care este o unitate dintr-o organigramă *ASM*, toate operaţiile se realizează în acelaşi timp, concurent. De

exemplu, dacă acest bloc de stare este interpretat ca o organigramă de program(flow chart), înâi se intră în starea (A), efectuându-se înscrierea valorii 1 în registrul R şi variabila de contor I se decrementeză, apoi se realizează operaţia de testare a intrării x_1. Dacă x_1 este adevărată se trece la activarea ieşirii y_1, după care se testează valoarea variabilei de contor I(valoare care a fost decrementată în starea (A)) şi în funcţie de valoarea de adevăr sau fals al acesteia, se realizează o trecere respectiv la stările (B) sau (C). Iar dacă x_1 are valoarea de fals se trece la testarea lui x_2 şi în funcţie de valoarea de adevăr sau fals se realizează trecerea la starea (D) sau se testează variabila de contor după care se ajunge în (B) sau (C).

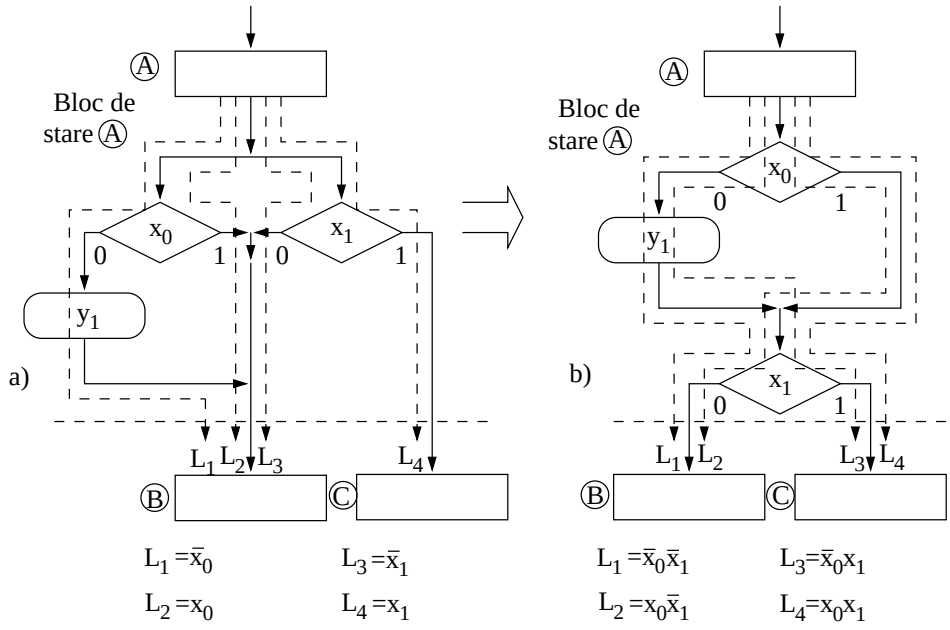


Figura 3.16 Exemplu de organigrame ASM(incorectă) cu validarea simultană a două cai de tranziţie L_1 şi L_3 sau L_2 şi L_4(a); formă de organigramă corectată(b)

Într-o organigramă ASM, timpul are o **curgere discretă**, realizându-se o trecere în salturi de la un bloc de stare la un alt bloc de stare, trecere marcată de impulsurile de ceas. De exemplu, pe frontul pozitiv al impulsului de ceas iT se intră în blocul de stare A reprezentat de starea (A). Operaţiile efectuate în paralel în blocul de stare sunt: se înscrie registrul R cu valoarea 1; se decrementează contorul I şi se poate înscrie condiţionat ieşirea y_1 şi tot în paralel se testează intrările x_1, x_2 şi I obţinându-se în funcţie de acestea ca una din cele cinci căi de tranziţie $L_1 \div L_5$ să aibă valoarea 1 pentru expresia funcţiei de tranziţie. **Atenţie!** în elementul de decizie pentru contorul I nu se testează valoarea variabilei de contor I care s-a obţinut în urma decrementării în starea (A), ci se testează valoarea lui I cu care a intrat în blocul

de stare; de fapt cele două operaţii, cea de decrementare din starea (A), $I \leftarrow I-1$, şi cea de testare din simbolul de romb, $I = 0$?, se realizează în paralel şi nu succesiv. Această valoare decrementată a contorului se va utiliza în blocurile următoare sau când se revine în blocul de stare (A), dacă între timp aceasta nu a mai fost modificată în stările care le-a mai parcurs. Mai sugestiv, am putea privi că pe frontul pozitiv al impulsului de ceas iT se intră în blocul de stare A, iar pe frontul pozitiv următor al impulsului $(i+1)T$ se realizează **simultan** toate operaţiile înscrise în blocul de stare A şi se trece prin una din caile de tranziţie, la una din starile următoare (B) (C) (D).

Pentru realizarea unei organigrame ASM corecte, fără ambiguităţi, se impun anumite restricţii:

1. Într-un bloc de stare, prin reţeaua de elemente de decizie, pentru o configuraţie a valorilor variabilelor de intrare testată, trebuie să fie validată doar o singură cale de tranziţie(restricţie similară cu evitarea dublei acoperiri de la graful de tranziţii). De exemplu, în Figura 3.16-a pentru configuraţia $x_1x_0 = 00$ există o tranziţie din (A) în (B), dar această tranziţie s-a realizat pe calea de transfer L_1 sau L_3? imposibil de spus. Sau, fie configuraţia $x_1x_0 = 11$ care va valida simultan căile de tranziţie L_2 şi L_4 cu trecere atât la starea (B) cât şi la starea C. Soluţia, pentru eliminarea ambiguităţilor este rearanjarea elementelor de decizie prin înserierea lor ca în Figura 3.16-b

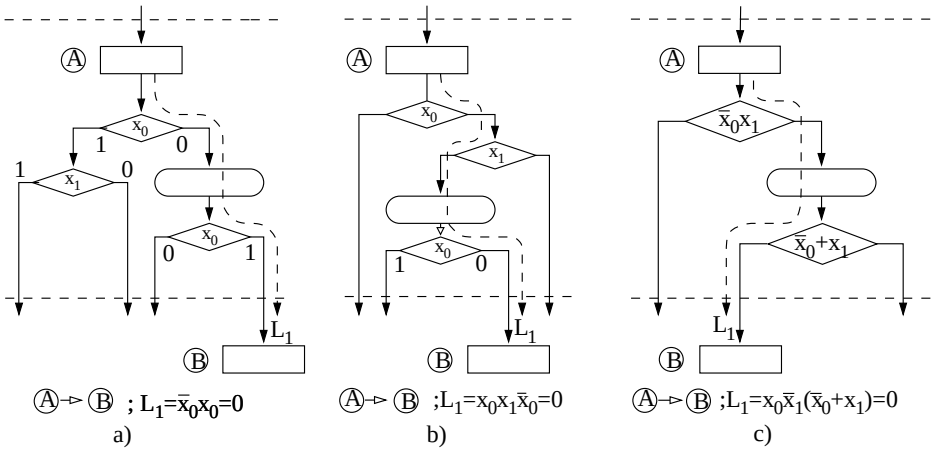


Figura 3.17 Exemple de organigrame ASM (incorecte) care testează pe aceeaşi cale de transfer L_1 o variabilă atât pentru valoarea falsă cât şi pentru valoarea adevărată($x_0\overline{x_0} = 0; x_1\overline{x_1} = 0$), deci testează o constantă:

2. Evitarea conectării simbolurilor de decizie, într-o reţea a unui bloc de bază, care să determine condiţii logice imposibile pentru o cale de tranziţie. Prin testarea unei variabile în mai multe simboluri de decizie care toate concură la validarea unei căi de tranziţie, se poate ajunge ca acea variabilă sa fie testată atât pentru

valoare falsă cât şi pentru valoare adevărată $x_0\overline{x_0} = 0$, de exemplu calea de tranziţie L_1 din Figura 3.17. Totuşi este perfect posibil ca aceeaşi variabilă să fie testată în mai multe simboluri de decizie, dar printr-o structurare potrivită să nu fie testată de două ori pe aceeaşi cale de tranziţie.

3. Orice cale de tranziţie în aceeaşi stare(buclă) trebuie să includă simbolul de stare. O buclă de tranziţie este necesară când automatul trebuie să rămână în aceeaşi stare un număr de tacturi până când este îndeplinită condiţia de test. O buclă aferentă unui bloc de stare trebuie să iasă şi să reintre în acel bloc numai pe cale normală ca în Figura 3.18-b.

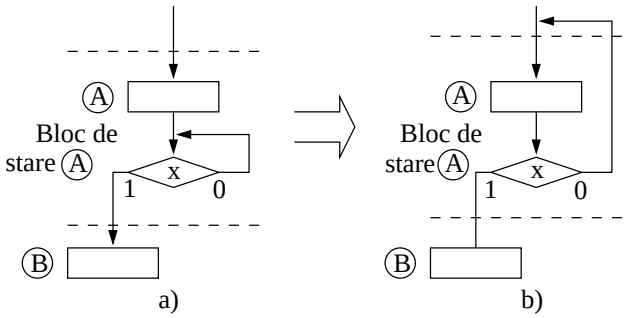


Figura 3.18 Bloc de stare în care o buclă este realizată incorect(a), aceeaşi buclă realizată corect(b)

Organigram ASM se obţine dintr-o reţea de blocuri de stare, numărul blocurilor de stare fiind egal cu numărul de stări $|Q|$ ale automatului. O astfel de organigramă descrie nu numai funcţionarea secvenţială a automatului dar exprimă şi funcţiile logice pentru sinteza acestuia. De exemplu blocul j corespunzător stării q_j a automatului descrie: starea prezentă q_j; ieşirile necondiţionate(Mealy) din acea stare, $f_j(q_j)$; ieşirile condiţionate $f_j^*(q_j, X_j)$ de tip Moore, tranziţiile spre stările următoare $g_j(X_j, q_j)$ în care X_j sunt cuvintele de intrare care se testează în blocul j. Luând toate aceste componente pentru toate cele $|Q|$ blocuri de stare ale automatului se obţine organigrama ASM.

$$g(X,Q) = \sum_{j=1}^{|Q|} g_j(X_j, q_j)$$

$$f(X,Q) = \sum_{j=1}^{|Q|} [f_j^*(X_j, q_j) + f(q_j)] \tag{3.16}$$

Organigramele ASM, spre deosebire de grafurile de tranziţie care se construiesc separat pentru automatul de tip Mealy sau pentru automatul de tip Moore, pot reprezenta simultan atât ieşiri condiţionate cât şi ieşiri dependente(numai de stare) deci pot îmbina funcţionarea celor două tipuri de automate.(Evident un graf de tranziţie de tip Mealy poate fi convertit într-un graf de tranziţie de tip Moore şi invers).

Exemplul 3.6 Pentru un automat de vânzare a sticlelor de suc să se conceapă organigrama ASM şi să se deducă tabelul de tranziţie al stărilor/ieşirilor. Funcţionarea automatului de vânzare a sticlelor cu valoarea de 10 unităţi, care poate fi acoperită din monede de două unităţi (2U), de cinci unităţi(5U) şi de zece unităţi (10U), este următoarea: când este în FUNCŢIUNE trebuie să semnalizeze aceasta printr-un semnal FUN (un LED), când este în curs de introducere monede, un semnal URM va indica dacă este necesară URMĂTOAREA MONEDĂ. Când monedele introduse totalizează 10 unităţi se comandă ELIBERARE STICLĂ, ELS, iar dacă suma monedelor nu totalizează 10U, atunci se comandă RETURNARE MONEDE, RTM.

Soluţie. În conceperea organigramei se urmăreşc variantele de a realiza 10 unităţi din 10, 5 sau 2 unităţi. Rezultă un număr de şapte stări $q_0 \div q_6$, Figura 3.19. Ar fi rezultat şi starea a opta, q_7, dacă comanda ELS nu s-ar fi generat ca o ieşire condiţionată ci ca o ieşire de starea q_7 (Moore). În opoziţie, s-ar fi putu reduce numărul de stări de la şapte la şase dacă comanda RTM din starea q_6 ar fi fost generată ca o ieşire condiţionată (Mealy) şi nu ca o ieşire dependentă de starea q_6.

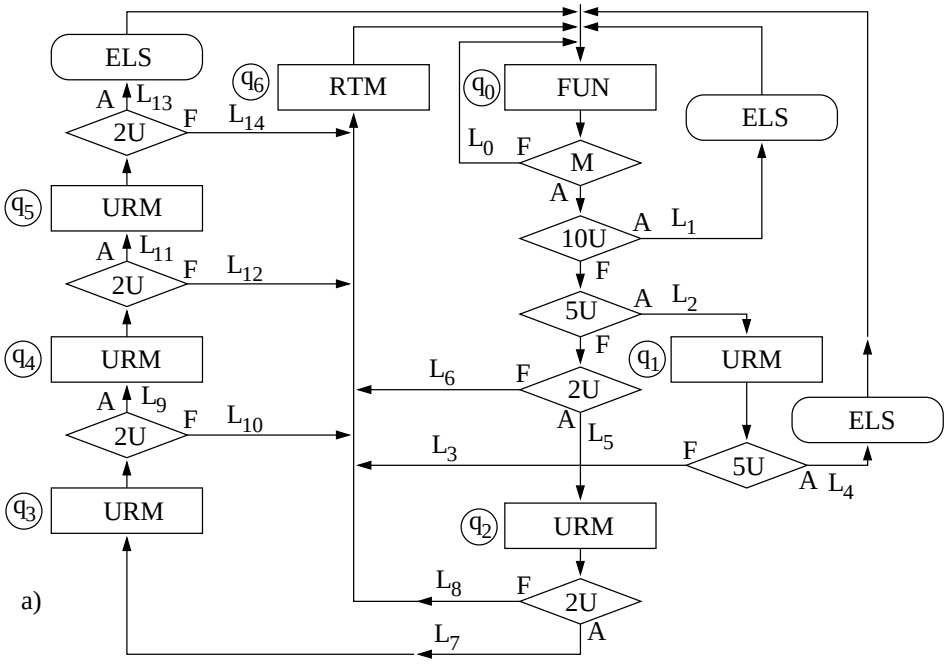
Dispozitivul de detectare a introducerii monedelor generează patru semnale de intrare pentru organigrama ASM: un semnal M pentru detectarea prezenţei monedei($M = A$); trei semnale corespunzătoare valorii monedelor detectate (10U, 5U şi 2U). Aceste semnale sunt semnale impuls care trebuie să fie sincrone cu semnalul de ceas al automatului. Atât timp cât nu este introdusă nici o monedă (M=F) automatul aşteaptă (buclează) în starea q_0.

Tabelul de tranziţie al stărilor se deduce din analiza organigramei. Acest tabel apare mai simplu decât cel definit în secţiunea 3.2.3.2, deoarece intrările sunt mutual exclusive (nu se pot introduce simultan două monede). Tabelul, Figura 3.19-b, apare mult mai intuitiv deoarece pentru fiecare bloc de stare(notate cu $B_0, B_1 \rightarrow, B_6$) se evidenţiază şi căile de tranziţie din acea stare, sunt în total 15 notate cu $L_1, L_2 \rightarrow L_{15}$

Uneori reducerea complexităţii unui ASM, precum şi o structurare pentru o implementare mai uşoară a unui automat de dimensiuni mari, se obţine prin descompunerea în mai multe ASM-uri componente şi conectarea acestora într-o anumită modalitate pentru a compune automatul iniţial. Fiecare din automatele componente sunt proiectate separat şi apoi interconenctate astfel ieşirile unui automat vor fi utilizate ca intrări la un alt automat şi viceversa. Aceste interconectări se reduc la conexiui serie sau paralel ale automatelor.

Un exemplu de conectare serie a două automate este schiţat în Figura 3.20-a. Maşina (automatul)A va fi starea (A_1) testând repetat, la fiecare impuls de ceas, variabila CALL_B şi va părăsi această stare, parcurgând restul stărilor din organigrama ASM până în starea (A_N), numai în momentul când ieşirea CALL_A din starea (B_1) a maşinii B devine activă. De asemenea, maşina B în starea (B_1) va testa repetat variabila CALL_A şi va părăsi această stare, parcurgând restul stărilor din organigrama ASM a sa, numai în momentul în care maşina A în starea (A_N) va activa ieşirea CALL_B care se aplică în starea (B_1) a maşinii B, ca variabilă testată. Apoi, maşina A ajunge iarăşi în starea (A_1) unde aşteaptă activarea ieşirii CALL_A

a)



b)

Blocul de stare si calea de tranzitie	Starea prezenta	Intrari				Starea urmatoare	FUN	URM	ELS	RTM
		M	2U	5U	10U					
B_0,L_0	q_0	–	–	–	–	q_0	1	0	0	0
B_0,L_1	q_0	A	–	–	A	q_0	1	0	1	0
B_0,L_2	q_0	A	–	A	–	q_1	1	0	0	0
B_0,L_5	q_0	A	A	–	–	q_1	1	0	0	0
B_0,L_6	q_0	A	F	–	–	q_6	0	0	0	0
B_1,L_4	q_1	–	–	A	–	q_0	0	1	1	0
B_1,L_3	q_1	–	–	F	–	q_6	0	1	0	0
B_2,L_7	q_2	–	A	–	–	q_3	0	1	0	0
B_2,L_3	q_2	–	F	–	–	q_6	0	1	0	0
B_3,L_9	q_3	–	A	–	–	q_4	0	1	0	0
B_3,L_{10}	q_3	–	F	–	–	q_6	0	1	0	0
B_4,L_{11}	q_4	–	A	–	–	q_5	0	1	0	0
B_4,L_{12}	q_4	–	F	–	–	q_6	0	1	0	0
B_5,L_{13}	q_5	–	A	–	–	q_0	0	1	1	0
B_5,L_{14}	q_5	–	F	–	–	q_6	0	1	0	0
B_6,L_{15}	q_6	–	–	–	–	q_0	0	0	0	1

Figura 3.19 Explicativă pentru Exemplul 3.6: a) descrierea funcţionării automatului prin organigramă ASM;b) tabelul de tranziţie al stărilor (neasignate) şi al ieşirilor.

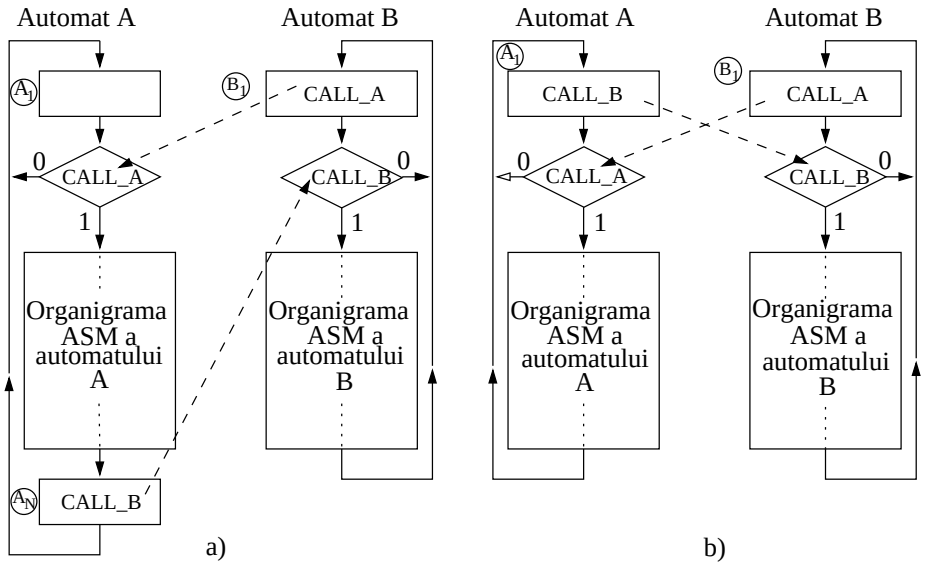


Figura 3.20 Interconectarea automatelor: a) interconectare de tip serie; b) interconectare de tip paralel.

realizată în starea (B_1) la care se ajunge numai după parcurgerea celorlalte stări din organigrama ASM a maşinii B. Deci se execută alternativ algoritmii implementaţi de fiecare maşină.

Pentru interconectarea serială se poate extinde organizarea la mai multe maşini, considerând una din ele ca maşina principală care va apela mai multe maşini pe parcursul executării algoritmului sau. La fiecare apelare a maşinii principale către o altă maşină (apelată), maşina principală va intra într-o buclă de aşteptare pe o stare şi nu va părăsi această bucla, până când maşina apelată nu va returna către maşina principală confirmarea realizării segmentului de algoritm încredinţat. O astfel de funcţionare este realizată de organizarea din Figura 3.21 cu particularizarea că maşina principală A apelează din stările (A_1), (A_R) şi (A_S) repetat numai o singură maşină, (C) pentru realizarea aceluiaşi subalgoritm(subrutină). Se recunoaşte aici acelaşi procedeu de apelare repetată în software a unei subrutine de către un program principal. Se poate extinde organizarea la cazul când o maşină apelează o altă maşină care la rândul său apelează o alta s.a.m.d; evident fiecare maşină apelantă intră în buclă de aşteptare până când primeşte confirmarea, de realizarea segmentului de algoritmi încredinţat, de la maşina apelată. Se recunoaşte mecanismul de apelare, în software, de subrutine imbricate.

Conectarea în paralel a două ASM-uri realizează execuţia în paralel a celor doi algoritmi. Cea mai simplă organizare corespunde cazului când cele două maşini sunt iniţiate împreună pentru procesare, Figura 3.20-b. Când maşina A porneşte procesarea din starea iniţială (A_1) simultan validează părăsirea stării iniţiale şi pe maşina

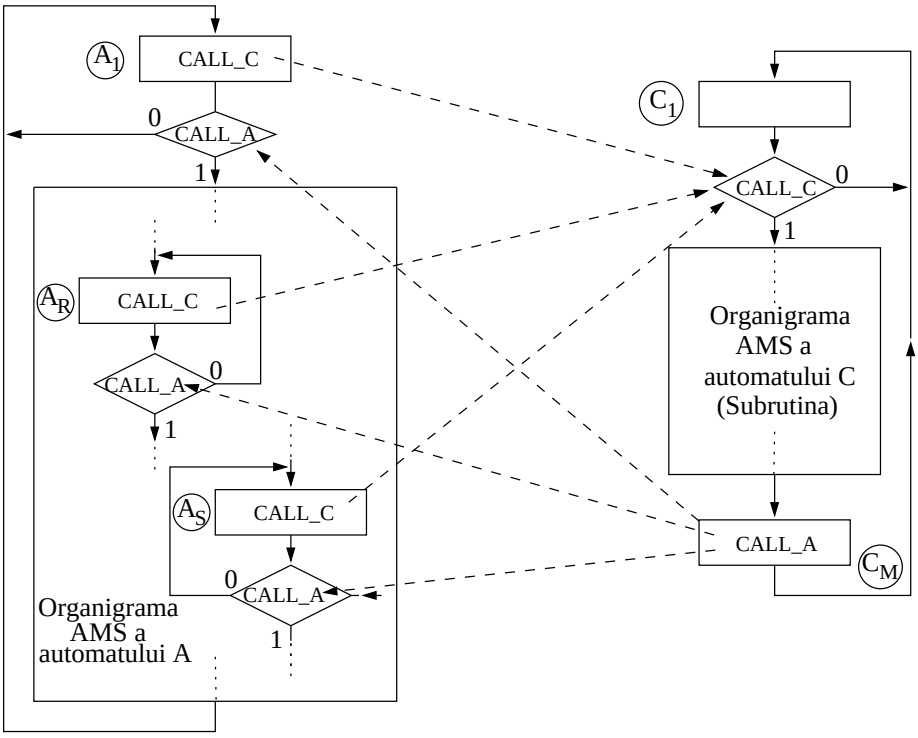


Figura 3.21 Interconectarea de tip serie a două automate pentru apelarea repetată a automatului C:

B care la rândul său prin CALL_A validează părăsirea stării inițiale (A_1). După parcurgerea în paralel a celorlalte stări din fiecare organigramă maşina care revine prima în starea inițială nu poate porni la reexecutarea algoritmului său pâna când şi cealaltă maşină nu revine în starea inițială.

Deşi din cele expuse, intreconectarea automatelor din punct de vedere logic apare relativ uşor de organizat dar, pentru ca şi din punct de vedere funcţional să fie corect, trebuie analizată foarte atent implementarea. Interconectarea ridică probleme de implementare chiar când cele două automate sunt sincronizate cu acelaşi semnal de ceas, deci timpii de stare, sunt identici, Figura 3.15-d. Există probleme în cazul în care semnalele de ceas ale celor două automate sunt unul multiplu faţă de celălalt sau, mai dificil, sunt complet asincrone. În acest caz semnalele de ieşire al unui automat se aplică la celalalt automat ca semnale de intrare asincrone sau se sincronizează înainte de aplicare, vezi secţiunea 3.2.1. Oricum, semnalele cu variaţie asincronă nu pot fi citite corect când sunt considerate că formează un cuvânt de cod ci numai când sunt considerate cu semnificaţie independentă fiecare semnal.

3.2.3.5 Limbaje de transfer între registre, RTL

Noţiunea de registru a fost introdusă în secţiunea 3.2.2 ca registru de stare. Până la prezentarea sa în secţiunea 3.5, specificăm că un registru este un circuit pentru

memorarea informaţiei sub formă de cuvânt, deci primele două operaţii care se pot realiza cu un registru este cea de înscriere în registru şi cea de citire din registru. În prezentările din capitolul 2 toate procesările pe un circuit combinaţional implicau prezenţa unui registru la intrare, din care se citea informaţia aplicată CLC, şi un registru la ieşire în care se înscriau, dacă era necesar, cuvântul rezultat din procesare. Alte operaţii în afară de înscriere(load) şi citire, registrul mai permite operaţii de ştergere(clear), incrementare, deplasare(shift) stânga dreapta(echivalent respectiv cu multiplicare sau împărţire cu puteri ale lui doi). Registrul împreună cu diferite circuite combinaţionale sau cu circuite secvenţiale, formează baza circuistică pentru procesarea paralelă sau secvenţială. De fapt, această circuistică este ceea ce se poate găsi într-o cale de date.

Formalismul cu ajutorul căruia se descriu operaţiile şi transformările între registrele dintr-o cale de date(şi nu numai) este referit prin limbaj de transfer între registre, **RTL** (**R**egister **T**ransfer **L**anguage).
Instrucţiunea

$$R_2 \leftarrow R_1$$

are următoarea semantică: conţinutul registrului R_1 este transferat(copiat) şi în registrul destinaţie R_2(conţinutul sursă R_1 nu se modifică); săgeata reprezintă transferul şi direcţia acestuia. De exemplu, în Figura 3.15-d în starea A, automatul comandă înscrierea valorii 1 în registrul R şi decrementarea registrului care conţine variabila contor I.

Normal că o operaţie, într-o cale de date, este realizată când este îndeplinită o condiţie logică C, ceea ce se exprimă printr-o primitivă de tipul **if-then**, de exemplu

$$If(C = 1)then(R_2 \leftarrow R_1)$$

Evident că acest transfer $R_2 \leftarrow R_1$, condiţionat de $C = 1$, se realizează numai sincronizat de semnalul de ceas prin frontul activ, deci condiţionarea şi de către semnalul de ceas este implicită. Dar se pot exprima şi transferuri care se realizează în paralel între mai multe registre, ca de exemplu

$$If(C = 1)then(R_2 \leftarrow R_1, R_4 \leftarrow R_3)$$

Alte exemple de instrucţiuni de tip RTL pot fi:

$R_1 \leftarrow R_1 + R_2$; se sumează conţinutul registrelor R_1, R_2, iar rezultatul se înscrie în R_1

$R_3 \leftarrow R_3 + 1$; conţinutul registrului R_3 se incrementează

$R_4 \ll R_4$; conţinutul registrului R_4 este deplasat spre stânga cu o poziţie

$R_5 \leftarrow 0$; este şters conţinutul registrului R_5

Simbolurile, notaţiile şi sintaxa unui limbaj RTL nu sunt standardizate, ceea ce îi oferă o dezvoltare ad-hoc şi o uşurinţă în utilizare. Totuşi, aceste simboluri, notaţii şi sintaxa descrierilor de tip RTL sunt standardizate în cadrul unor limbaje de descriere hardware **HDL** (**H**ardware **D**escription **L**anguage) cum ar fi **VHDL** sau **VERILOG** (prezentate în volumul 2 al acestei lucrări).

Organigrama ASM a fost introdusă pentru descrierea algoritmilor implementabili într-un sistem hardware, care în general, conţine o cale de date şi o cale de control, Figura 3.14. Calea de date are în componenţa sa registre, multiplexoare, decodificatoare, sumatoare iar calea de control este în fond un automat finit ale cărui ieşiri sunt semnalele de control în calea de date. Ca o alternativă la descrierea prin organigrama ASM este cea prin care calea de control este descrisă printr-un graf de tranziţie al stărilor, iar calea de date descrisă prin limbaj RTL. Informaţia pentru trecerea la graful de tranziţie al stărilor este luată din organigrama ASM astfel:

numele stărilor (din cerculeţe) corespund cu cele de lângă dreptunghiurile de starea ale ASM-ului

arcele de transfer dintre stări sunt căile de tranziţie din ASM

expresiile de tranziţie de pe arce sunt expresiile de pe căile (sau din romburile) de tranziţie ale ASM-ului

Iar informaţia, referitoare la operaţiile care se efectuează în calea de date, exprimată RTL, pentru fiecare stare, se obţine din ieşirile de stare (Moore) şi din ieşirile condiţionale(Mealy) ale blocului de stare corespunzătoare.

Exemplul 3.7 Pentru automatul prezentat în Exemplul 3.6, Figura 3.19 să se realizeze o descriere prin graf de tranziţie pentru calea de control şi prin limbaj RTL pentru calea de date.

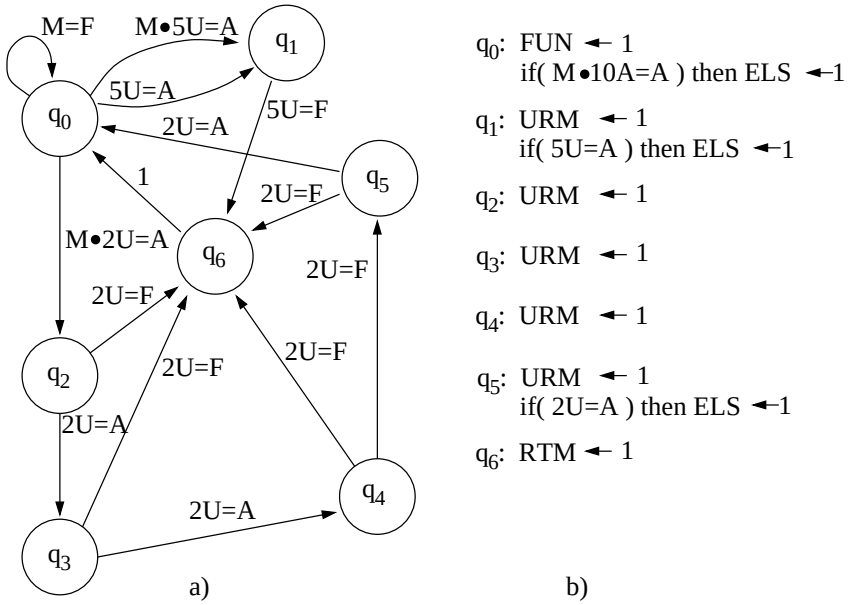


Figura 3.22 Alternativă la descrierea prin organigramă ASM a automatului din figura 3.19-a: a) printr-un graf de tranziţie al stărilor pentru calea de control; b) prin limbaj RTL pentru calea de date.

Soluţie. Aplicând organigramei ASM din Figura 3.19-a regulile prezentate mai sus se obţine graful de tranziţie al stărilor, Figura 3.22-a şi descrierea în limbaj RTL, Figura 3.22-b. Pentru fiecare din cele şapte stări $q_0 \div q_6$, în limbaj RTL, sunt specificate comenziile aplicate în calea de date obţinute ca semnale de ieşire de la calea de control ale automatului(calea de control).

În afara modalităţiilor prezentate de reprezentare a automatelor mai putem aminti: limbaj natural, reţea Petri, HDL. Reprezentarea prin **limbaj natural** este utilă când automatul prezintă un număr foarte mare de stări, de exemplu un automat numărător modul 16 care prezintă $2^{16} = 64536$ stări. **Reţeaua Petri** este tot o structură de graf orientat dar nu urmăreşte o succesiune de stări ci o succesiune de evenimente [Lewin ′92], recomandată mai mult pentru descrierea circuitelor/(proceselor) secvenţiale asincrone decât a celor sincrone. Foarte utilizată reţeaua Petri este pentru sistemele comandate la care sunt active simultan mai multe stări. Limbajele de descriere hardware, HDL (VHDL, Verilog) au devenit modalităţi de descriere aproape exclusive în procesul de proiectare în electronica digitală.

3.2.4 Reducerea numărului de stări

Proiectantul în etapa de construire a grafului de tranziţie al stărilor/ieşirilor, concentrându-se asupra restricţiilor de funcţionarea ale automatului, poate realiza un model (graf) ce conţine stări redundante. Aceasta înseamnă că vor rezulta în tabelul de tranziţie mai multe stări decât cele necesare interpretării funcţionării. Mai multe stări în implementare influenţează în mod direct dimensiunea, deci costul implementării, viteza de funcţionare şi complexitatea automatului. În procesul de proiectare, etapa de eliminare a stărilor redundante este referită prin reducerea stărilor. Efectiv reducerea stărilor înseamnă identificarea stărilor echivalente şi apoi un număr de stări găsite ca echivalente sunt substituite, în funcţionarea automatului, cu o singură stare. Rezultă că în tabelul de tranziţie al stărilor/ieşirilor toate liniile care corespund unor stări echivalente se vor substitui cu o singură linie rezultantă. Noţiunea de stare echivalentă este introdusă prin Definiţia 3.7 . Adăugăm acum trei proprietăţi ale stărilor echivalente:

1. Simetria: dacă $q_i \sim q_j$ atunci şi $q_j \sim q_i$
2. Reflexivitatea: $q_j \sim q_j$ pentru oricare stare
3. Tranzitivitatea: dacă $q_i \sim q_j$ şi $q_j \sim q_k$ atunci $q_i \sim q_k$

Există algoritmi, atât utilizabili sub o formă analitică cât şi sub o formă grafică, prin care se identifică stările echivalente şi apoi sunt eliminate [Yarbrough ′97] [Mano ′97] [Lewin ′92]. Actual, fiecare platformă de proiectare automată în electronică conţine astfel de algoritmi sub forma unor programe interactive. În continuare se va prezenta o metodă grafică de reducere a stărilor echivalente referită prin **mapa implicanţilor** [Yarbrough ′97].

Se consideră un automat cu şase stări: A, B, C, D, E, F, a cărui funcţionare este descrisă prin tabelul de tranziţie al stărilor/ieşirilor prezentat în Figura 3.23-a. O **mapă a implicanţilor** care compară, pentru echivalenţă, fiecare stare cu fiecare stare este o diagramă, Figura 3.23-b, ce are pe orizontală notate toate stările mai puţin ultima,

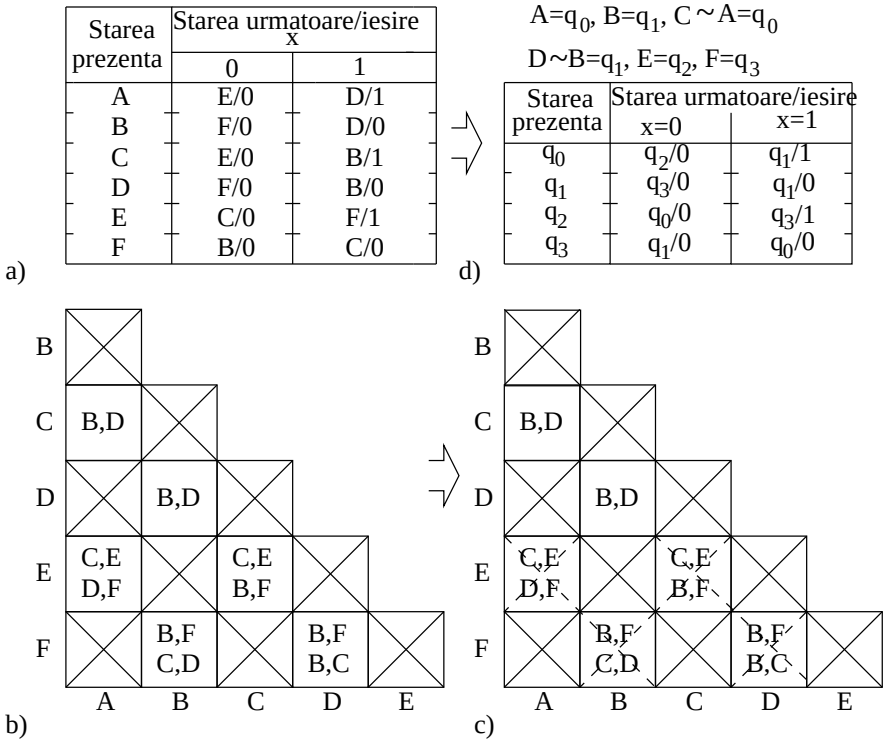


Figura 3.23 Etapele reducerii numărului de stări prin metoda grafică a mapei implicanţiilor: a) tabelul de tranziţie a stărilor/ieşirilor automatului; b) mapa implicanţiilor dedusă din tabelul de tranziţie; c) mapa implicanţiilor după validarea condiţiilor de echivalenţă; d) tabelul de tranziţie al stărilor/ieşirilor pentru automatul echivalent(redus).

în cazul nostru A, B, C, D, E, iar pe verticală toate stările mai puţin prima, în cazul nostru B, C, D, E, F. În căsuţa de coordonate care corespunde intersecţiei celor două stări care se compară, una de pe orizontală şi una de pe verticală, se înscriu condiţiile (stările) necesare pentru echivalenţa celor două stări. Aceste condiţii necesare pentru echivalenţă se deduc din tabelul de tranziţie al stărilor/ieşirilor, comparând fiecare stare cu fiecare, pentru toate configuraţiile de valori ale cuvântului de intrare ce determină cele două stări. Verificarea condiţiilor necesare pentru ca o pereche de stări să fie echivalentă se realizează prin următorii paşi:

1. Se elimină (se diagonalizează) din mapa implicanţiilor acele căsuţe ale căror perechi de stări coordonate au în tabelul de tranziţie al stărilor/ieşirilor ieşiri diferite pentru aceeaşi valoare a cuvântului de intrare(vezi Definiţia 3.7). De exemplu, nu pot fi echivalente perechiile A cu B, A cu D şi A cu F, deoarece au ieşiri diferite, deci căsuţele corespunzătoare intersecţiei acestor perechi se diagonalizează.

2. Pentru perechile de stări prezente la care s-a verificat că au ieşiri identice,

după inspectarea tabelului, se înscrie în căsuţele corespunzătoare din mapa implicanţilor care perechi de stări următoare ar trebui să fie echivalente pentru ca perechea de stări prezente comparate să fie echivalentă. De exemplu, pentru prima coloană din mapă pentru ca $A \sim C$ necesită echivalenţa între B şi D, iar $A \sim E$ necesită $C \sim E$ şi $D \sim F$; pe coloana a doua $B \sim D$ necesită $B \sim D$, $B \sim F$ necesită $B \sim F$ şi $C \sim D$; pe coloana a treia $C \sim E$ necesită $C \sim E$ şi $B \sim F$; iar pe coloana a patra $D \sim F$ necesită $B \sim F$ şi $B \sim C$.

Pentru ca $A \sim E$ necesită $C \sim E$ şi $D \sim F$. Privind în mapa implicanţiilor pentru echivalenţa perechii D, F este necesară $B \sim F$ şi $B \sim C$, iar pentru echivalenţa perechii $C \sim E$ este necesară $C \sim E$ şi $B \sim C$. Să vedem dacă din căsuţele mapei rezultă că sunt îndeplinite simultan aceste patru echivalenţe. Pentru ca $B \sim F$ este necesar ca $B \sim F$ şi $C \sim D$, dar din căsuţa de coordonate C şi D rezultă că stările C şi D nu sunt echivalente. În concluzie , stările A şi E nu sunt echivalente şi căsuţa de coordonate A şi E de pe prima coloană se diagonalizează.

Se aplică această analiză şi pentru celelalte coloane/linii şi se constată că în afară de căsuţa de coordonate B,D, toate celelalte se diagonalizează, rezultând mapa implicanţiilor din Figura 3.23-c. Rezultă că sunt echivalente stările A cu C şi stările B cu D.

3. Pentru fiecare pereche de stări echivalente se păstrează doar una din stări, eventual se redenumesc stările ($A = q_0$, $B = q_1$, $C \sim A = q_0$, $D \sim B = q_1$, $E = q_2$, $F = q_3$) şi se construieşte tabelul simplificat al tranziţiei stărilor/ieşirilor, Figura 3.23-d. Deci automatul iniţial cu şase stări poat fi substituit în funcţionare cu un alt automat(echivalent) care are numai patru stări.

Exemplul 3.8 Pentru automatul Mealy cu graful de tranziţie al stărilor/ieşirilor din Figura 3.24-a să se elimine stările redundante şi apoi să se redeseneze graful de tranziţie

Soluţie. Din graful de tranziţie al stărilor/ieşirilor se obţine tabelul de tranziţie al stărilor/ieşirilor din Figura 3.24-e, iar din tabelul de tranziţie se deduce:

1. Nu sunt echivalente următoarele perechi de stări:A cu B, A cu C, B cu D, B cu E, C cu D şi C cu E
2. Stările: $A \sim D$ numai dacă sunt echivalente B cu C şi D cu C şi A cu B

   $A \sim E$ numai dacă sunt echivalente A cu E şi B cu C

   $B \sim C$ numai dacă sunt echivalente B cu C şi A cu E

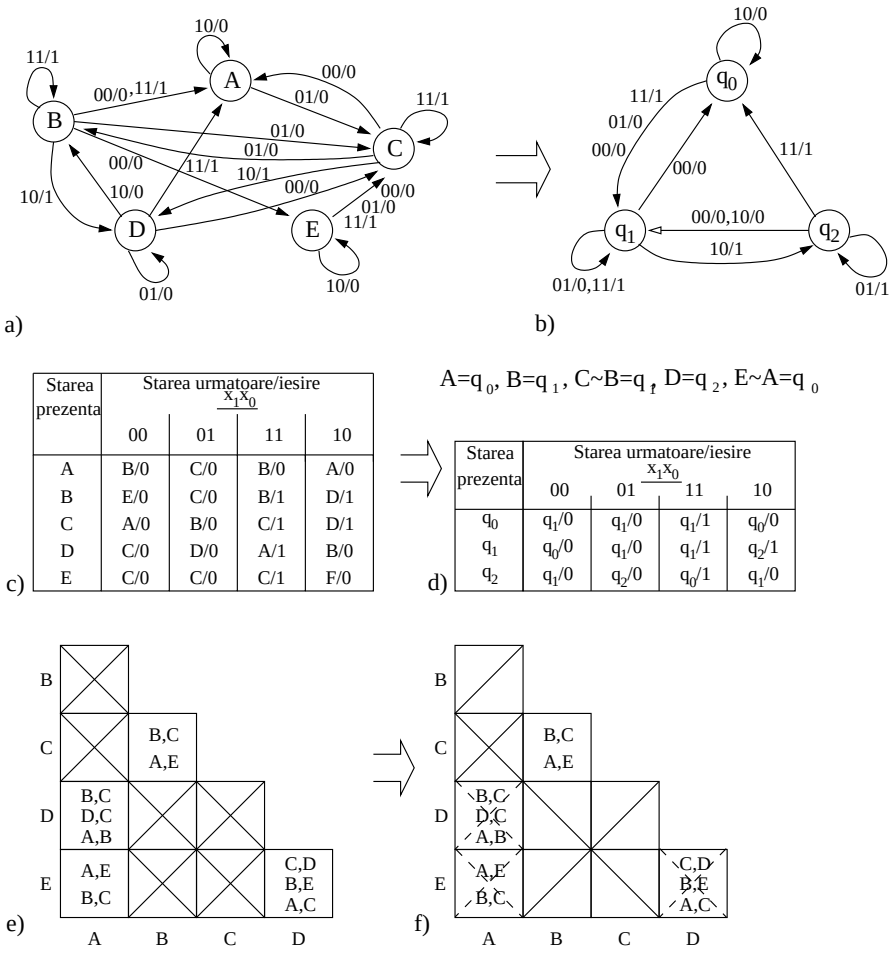
   $D \sim E$ numai dacă sunt echivalente C cu D şi B cu E şi A cu C
3. Se constată că: $A \neq D$ deoarece $A \neq B$ şi $D \neq C$

   $D \neq E$ deoarece $A \neq C, B \neq E$ şi $C \neq D$

   $A \sim E$ numai dacă $B \sim C$, iar $B \sim C$ numai dacă $A \sim E$. Rezultă că $A \sim E$

   şi $B \sim C$(simetrie dacă $q_j \sim q_i$ atunci $q_i \sim q_j$). Mapa implicanţiilor rezultată

   este cea din Figura 3.24-f.
4. Se redenumesc stările în felul următor: $A = q_0, B = q_1, C \sim A = q_0$;



Starea prezenta	Starea urmatoare/iesire x_1x_0 = 00	01	11	10
A	B/0	C/0	B/0	A/0
B	E/0	C/0	B/1	D/1
C	A/0	B/0	C/1	D/1
D	C/0	D/0	A/1	B/0
E	C/0	C/0	C/1	F/0

Starea prezenta	Starea urmatoare/iesire x_1x_0 = 00	01	11	10
q_0	q_1/0	q_1/0	q_1/1	q_0/0
q_1	q_0/0	q_1/0	q_1/1	q_2/1
q_2	q_1/0	q_2/0	q_0/1	q_1/0

Figura 3.24 Explicativă pentru reducerea stărilor redundante ale automatului din Exemplul 3.8

$D = q_2, E \sim A = q_0$ pentru care rezultă tabelul de tranziţie din Figura 3.24-d.

Funcţionarea automatului echivalent(redus) este descrisă prin graful de tranziţie al stărilor/ieşirilor din Figura 3.24-b.

În sinteza circuitelor combinaţionale se întâlnesc cazuri când funcţia de transfer este incomplet specificată, ceea ce în tabelul de adevăr se indică prin semnul indiferent (don′t care). Cazuri similare se întâlnesc şi la circuitele secvenţiale când fie funcţia de transfer, f, fie funcţia de tranziţie, g, nu sunt complet specificate ceea ce se reflectă prin indicarea semnului indiferent respectiv pentru anumite valori ale ieşirii sau anumite stări următoare. Când anumite stări nu sunt specificate automatul nu este predictibil. Este indicat să se evite asemenea cazuri fie prin alegerea numai acelor

configuraţii de intrare care conduc automatul numai prin stări deja specificate sau, fie să se specifice stările nespecificate dacă prin aceasta nu se contravine rezultatului dorit. Odată specificate stările, care iniţial erau nespecificate, automatul nu mai este incomplet specificat.

În procesul de reducere al numărului de stări, specificarea ieşirilor nespecificate poate fi făcută fără a se produce nici un impact asupra secvenţei stărilor automatului final. Este indicat ca ieşirile nespecificate să fie lăsate nespecificat în tabelul de tranziţie al stărilor/ieşirilor cât se poate de mult în procesul de reducere, deoarece prezenţa semnului indiferent duce la o mai mare flexibilitate în compararea stărilor din tabelul de tranziţie. În validarea echivalenţei a două stări, trebuie ca ieşirile să fie identice pentru oricare configuraţie de intrare. Când se compară două stări ale căror ieşiri sunt incomplet specificate în locul noţiunii de echivalenţă se utilizează noţiunea de stări compatibile. Două stări q_i şi q_j sunt stări compatibile dacă, pentru oricare secvenţa de configuraţii aplicate pe intrare, se obţine o aceeaşi secvenţă a ieşirii când ieşirile nespecificate vor fi specificate, indiferent dacă stările q_i, q_j sunt stări iniţiale sau nu. Pentru automatele incomplet specificate există algoritmi(grafici sau analitici) de reducere a numărului de stări [Yarbrough ′97][Lewin ′92].

3.2.5 Asignarea stărilor

După operaţia de reducere a stărilor echivalente şi modificarea corespunzătoare a grafului de tranziţie al stărilor/organigrama ASM şi a tabelului de tranziţie al stărilor se asignează stările. Procesul de asignare al stărilor (codificarea stărilor) unui automat cu s stări constă în alocarea pentru fiecare simbol de stare $q_0, q_1, \ldots, q_{s-1}$ a câte unui cuvânt de cod cu lungimea de minimum k biţi. Valoarea lui k rezultă din relaţia

$$2^{k-1} \leq s < 2^k \tag{3.17}$$

deci $k = \lceil log_2\ s \rceil$

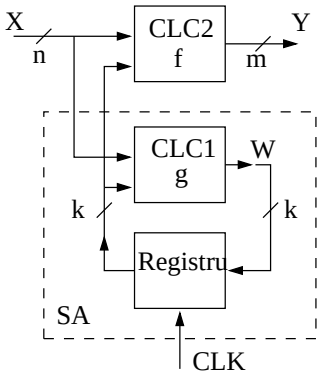


Figura 3.25 Separarea semiautomatului SA în structura unui automat:

Prin asignarea stărilor, automatul definit simbolic este transformat într-un automat cu o structură specificată şi funcţiile de transfer, f, şi de tranziţie, g, sunt fixate.

De fapt, "se personalizează" un semiautomat pentru automatul respectiv, Definiţia 3.8 . În Figura 3.25 este reprezentată o structură generică de automat a cărui structură este separată în partea combinaţională, CLC2, pentru calculul funcţiei de transfer, şi semiautomat, SA. Semiautomatul este compus din circuitul combinaţional CLC1, pentru calculul funcţiilor de excitaţie W, iar în jurul acestuia este închisă bucla printr-un registru(element de memorie). Prin asignare fiecare stare prezentă şi fiecare stare următoare sunt exprimate respectiv prin cuvintele Z şi W cu lungimea de k biţi(din mulţimea de 2^k cuvinte).

În proiectarea automatului se porneşte de la descrierea transferului intrare/ieşire, deci este fixată funcţionarea prin corespondenţă între vectorii X şi Y de la bornele automatului. Proiectantului, pentru o funcţionare dată a unui automat, îi rămâne sarcina de a optimiza realizarea funcţiilor f şi g, ceea ce se reduce la alegerea semiautomatului cel mai potrivit din mulţimea de semiautomate posibile. Ori, această alegere este determinată/fixată prin procesul de codificare al stărilor. Importanţa alegerii semiautomatului cel mai potrivit este evidentă pentru un automat, deoarece această alegere determină direct dimensiunea şi complexitatea automatului. Deoarece automatele în general nu pot fi definite recursiv, deci dimensiunea definiţiei poate fi ridicată, rezultă, în consecinţă, circuite complexe. Un automat complex având şi o dimensiune ridicată va fi mai greu de realizat; este de preferat de realizat un automat de complexitate redusă chiar dacă dimensiunea(circuistica) este ridicată. Rezultă necesară cunoaşterea posibilităţilor de alegere a cuvintelor de cod din mulţimea de 2^k cuvinte, care de fapt este spaţiul stărilor exprimat prin cuvinte de cod. **Codificarea stărilor este cea mai importantă etapă în proiectarea unui automat**.

Tabelul 3.1 Posibilităţi de codificare pentru un automat cu patru stări

Starea	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0
q_0	00	00	00	00	00	00	01	01	01	01	01	01
q_1	01	01	10	10	11	11	10	10	11	11	00	00
q_2	10	11	01	11	01	10	11	00	10	00	10	11
q_3	11	10	11	01	10	01	00	11	00	10	11	10

Starea	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}	C_{20}	C_{21}	C_{22}	C_{23}	C_{24}
	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0	z_1z_0
q_0	10	10	10	10	10	10	11	11	11	11	11	11
q_1	11	11	00	00	01	01	00	00	01	01	10	10
q_2	00	01	11	01	11	00	01	10	00	10	00	01
q_3	01	00	01	11	00	11	10	01	10	00	01	00

Pentru un automat cu s stări din mulţimea de 2^k cuvinte, cu lungimea de k biţi, se pot forma un număr de $2^k!/(2^k - s)!s!$ grupuri distincte, fiecare grup conţinând s cuvinte. Apoi, prin maparea celor s stări ale automatului pe un grup cu s cuvinte rezultă un număr de $s!$ asignări(codificări) diferite. Deci numărul total de codificări, N_c, va fi:

$$N_C = \frac{2^k!}{(2^k - s)!} \tag{3.18}$$

De exemplu, pentru un automat cu patru stări q_0, q_1, q_2 şi q_3(codificabile pe un cuvânt de doi biţi $k = \lceil log_2 4 \rceil = 2$) rezultă $N_C = 2^2!/(2^2 - 4)! = 24$ posibilităţi de asignare, $c_1 \div c_{24}$, reprezentate în Tabelul 3.1. Dar din cele 24 de asignări posibile unele sunt echivalente. Două asignări ale stărilor sunt echivalente dacă pentru cele două implementări ale automatului funcţiile f şi g au aceleaşi mărimi de dimensiune şi complexitate. Se obţin două asignări echivalente în următoarele două cazuri:

1. când codul unei asignări se obţine prin complementarea codului celeilalte. De exemplu, sunt echivalente următoarele perechi de asignări (c_1, c_{24}), (c_2, c_{23}), (c_3, c_{22}),. . .,(c_{12}, c_{13});

2. când codul unei asignări se obţine din codul altei asignări prin interschimbarea coloanelor z_1 şi z_0. Dacă în asignarea c_1 se schimbă valorile biţilor între z_1 şi z_0 se obţine asignarea c_3, la fel c_2 şi c_4 s.a.m.d

Numărul total de asignări neechivalente, N_{cne}, pentru un automat cu s stări şi cuvânt de lungime de k biţi se calculează cu următoarea relaţie

$$N_{cne} = \frac{(2^k - 1)!}{(2^k - s)!s!} \tag{3.19}$$

Valoarea lui N_{cne} creşte dramatic în funcţie de numărul de stări s ale automatului, câteva din aceste valori sunt date în Tabelul 3.2.

Tabelul 3.2 Valori pentru numărul asignărilor neechivalente, Ncne

Tabelul 3.2 Valori pentru numarul asignarilor neechivalente, Ncne

s	k	Ncne	s	k	Ncne
2	1	1	8	3	840
3	2	3	9	4	$10{,}81 \times 10^6$
4	2	3	10	4	$75{,}67 \times 10^6$
5	3	140	16	4	$56{,}48 \times 10^9$
6	3	420	20	5	$143{,}14 \times 10^{21}$
7	3	840			

În proiectarea unui automat efortul de alegere a celui mai bun semiautomat, care determină o dimensiune minimă pentru circuitele combinaţionale, $CLC1$ şi $CLC2$, precum şi o minimizare a complexităţii, este evitat din cauza numărului mare de asignări posibile în spaţiul stărilor care ar trebui analizate. Pentru $s \leq 4$ se pot elabora şi analiza toate soluţiile de semiautomat şi apoi se alege varianta care duce la un minim pentru dimensiune şi complexitate. Pentru cazul când $n \geq 4$ se caută doar găsirea soluţiei care determină un semiautomat ce se aproprie suficient de mult de cel mai bun, am putea spune semiautomatul cel mai potrivit. În acest sens există în literatură indicate anumite reguli care aplicate în procesul de asignare al stărilor pot duce spre o soluţie aproape de cea optimă.

Variabilelor de excitaţie w, care determină starea următoare, se calculează în blocul combinaţional $CLC1$, iar mărimile de ieşire y sunt calculate în blocul combinaţional $CLC2$; în calculul acestor două marimi intervine starea prezentă. În sinteza

celor două blocuri combinaţionale se obţin forme cu atât mai simple cu cât în diagramele V-K se identifică suprafeţe(implicanţi primi esenţiali, IPE) cât mai mari. Ori, pentru a obţine suprafeţe cât mai extinse este indicat ca asignarea stărilor să se efectueze astfel incât codurile stărilor să aibă între ele distanţa unitara şi prin aceasta la maparea pe o diagrama V-K vor corespunde căsuţe adiacente. Pe această observaţie se bazează unele reguli care pot duce la obţinerea de structuri simple pentru cele două blocuri combinaţionale.

Un procedeu simplu de asignare a stărilor care are o probabilitate destul de ridicată pentru a duce la o soluţie apropiată de cea optimă se bazează pe conceptul de locusul stărilor. **Locusul stărilor** este numărul de modificări ale biţilor din cuvântul de stare când pentru un automat se parcurg toate căile de tranziţie. Se va alege acea codificare a stărilor care să realizeze o valoare cât mai mică pentru locusul stărilor. Valoarea minimă s-ar obţine când codurile între două stări succesive pe oricare linie de tranziţie, în afară de buclele la aceeaşi stare, ar diferi doar printr-un singur bit. Această modalitate de codificare, prin care codul unei stări diferă doar printr-un singur bit faţă de codurile stărilor următoare, înspre care există o cale de tranziţie, este referită ca o **codificare cu variaţie minimă**. Codificarea cu variaţie minimă, pe lângă faptul că pot duce duce la suprafeţe mai mai într-o diagramă V-K, prin realizarea unei singure comutaţii în cuvântul de stare la trecerea de la o stare la alta, măreşte siguranţa în funcţionare a automatului, mai ales la cele asincrone.

Un segment de organigramă ASM având o codificare cu variaţie minimă este prezentat în Figura 3.26-a. La o tranziţie din starea q_0, căreia i s-a asignat codul 0000, pe oricare cale de tranziţie înspre stările q_1, q_2, q_3, q_4 în cuvântul de cod se modifică doar un singur bit. Dacă se notează cu nz_3, nz_2, nz_1, nz_0, valorile biţilor din cuvântul de stare următoare, acestea se pot exprima prin relaţia logică ce defineşte calea de tranziţie respectivă.

$$nz_3 = x_1x_0 \qquad nz_2 = x_1\overline{x_0}$$
$$nz_1 = \overline{x_1}\,\overline{x_0} \qquad nz_0 = \overline{x_1}x_0$$

Codificarea cu diferenţa numai de un singur bit, între două stări succesive, nu este totdeauna posibilă. De exemplu, în organigrama ASM din Figura 3.26-b, care conţine pe o buclă trei stări q_0, q_1 şi q_2 acestea nu pot fi toate codificate cu distanţe de cod egale cu unu. Dacă q_1 diferă de q_0 cu un singur bit iar faţă de q_2 tot cu un singur bit, atunci q_2 diferă de q_0 prin cel puţin doi biţi. Aceste relaţii între cuvintele de cod asignate stărilor se pot observa şi modifica mult mai uşor dacă se construieşte o diagramă de tip V-K în care se mapează stările. Într-o astfel de diagramă, două stări succesive care au distanţa de cod egală cu unu, sunt plasate în căsuţe adiacente.

Din diagrama V-K corespunzătoare automatului din Figura 3.26-b rezultă că totdeauna, pentru oricare variantă de asignare cu 2 biţi între cele trei stări q_0, q_1 şi q_2 de pe buclă, există o trecere(pe diagonală) între două căsuţe neadiacente. Uneori, pentru realizarea unei codificări cu distanţa de cod unu între oricare două stări succesive este necesară introducerea unei stări suplimentare, dar această suplimentare de stări pe întreaga organigramă ASM ar putea duce la creşterea numărului de biţi în cuvântul de cod. Prin introducerea stării suplimentare q_4, Figura 3.26-c, cuvântul de cod va fi de trei biţi dar codurile succesive asignate stărilor q_0, q_1, q_2 şi q_4 de pe buclă sunt la distanţa unitară, ceea ce se obţine din plasarea stărilor pe diagrama V-K în poziţii adiacente, (această plasare nu este unică). Această rezolvare pentru codificarea cu

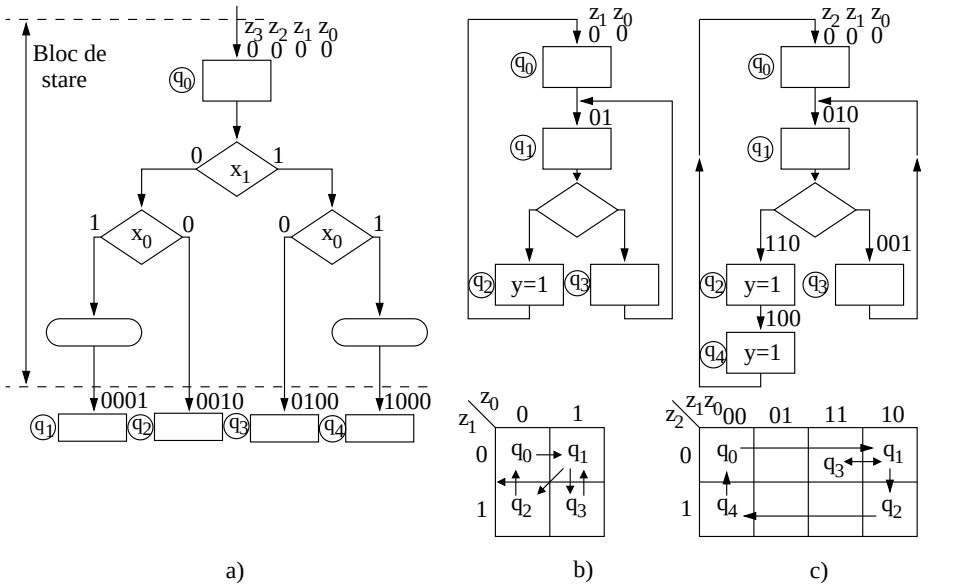


Figura 3.26 Codificare cu variaţie minimă: a) segment de organigramă ASM codificat cu variaţie minimă; c) codificare cu variaţie minimă prin introducerea unei stări suplimentare faţă de organigrama(b).

variaţie minimă implică definirea şi a unei ieşiri în starea suplimentară. Ieşirea dependentă de starea suplimentară poate fi repetarea ieşirii y din starea anterioară sau nici o ieşire NO-OPERATION dacă funcţionarea automatului permite. Generarea ieşirii, $y = 1$, şi în starea suplimentară apare la ieşirea automatului ca o consumare de doi timpi de stare(tacte) în starea q_2.

Pentru un automat valoarea locusului stărilor nu poate fi mai mică decât numărul s de stări. O valoare minimă egală cu s pentru locusul stărilor se poate obţine doar la un automat la care există s tranziţii necondiţionate între stările succesive şi codificarea stărilor este cu variaţie minimă, de exemplu un numărător în cod $Gray$, uzual valoarea minimă a locusului stărilor este mai mare decât s.

Exemplul 3.9 Pentru automatul din Figura 3.27-a să se calculeze locusul stărilor, apoi să se reasigneze stările astfel încât să se obţină valoarea minimă pentru locusul stărilor.

Soluţie. Pentru codurile stărilor înscrise pe organigramă se calculează locusul stărilor, Figura 3.27-b, rezultând o valoare egală cu 10. Din plasarea stărilor pe o diagramă V-K se vede că stările nu au coduri adiacente. Replasând stările pe diagrama V-K, Figura 3.27-c, se face o codificare cu variaţie minimă(cuvintele binare notate în paranteze pe organigramă) pentru care locusul obţine valoarea 7. În această diagramă V-K pe săgeţile care reprezintă căile de tranziţie dintre stări s-a indicat şi distanţa de cod respectivă, dacă această distanţa este mai mare decât unitatea. Pe bucla formată din stările q_1, q_3, q_4 nu se poate realiza o codificare cu variaţie minimă. Asignarea codului 011 pentru q_4 ar duce la o distanţă de cod unitară faţă de q_1 dar ar mări distanţa de cod la doi faţă de q_3, locusul stărilor ar fi tot 7. Se pot încerca şi alte plasări în mapa stărilor. Soluţia ar fi introducerea unei stări suplimentare

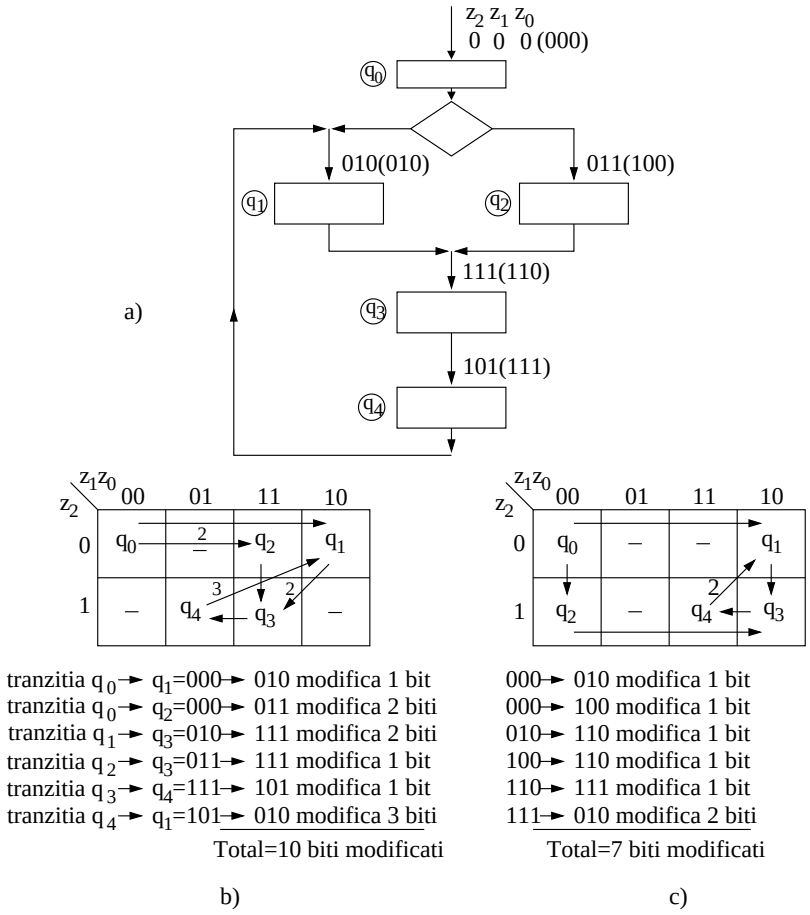


Figura 3.27 Exemplu de plasare a stărilor pe diagrama V-K(b) pentru a minimiza valoarea locusului stărilor(c) la o organigramă dată(a).

pe această bucla.

Exemplul 3.10 Pentru automatul cu tabelul de tranziţie al stărilor/ieşirilor din Figura 3.28-a să se deducă expresia semnalelor de excitaţie.

Soluţie. Deoarece se deduc doar expresiile funcţiilor de excitaţie se vor neglija valorile ieşirilor, deci problema exprimată pentru automat se restrânge numai la sinteza semiautomatului. Se propun două variante de semiautomat.

În Varianta I, Figura 3.28-b, cele şase stări $q_0 \div q_5$ sunt mapate pe diagrama V-K astfel încât să se realizeze într-o măsură cât mai mare variaţie minimă, adică plasare în căsuţe adiacente. În afară de tranziţiile q_3 la q_5 şi q_5 la q_1, care au distanţa de cod egală cu 2, toate celelalte tranziţii au distanţa de cod unitară, rezultă o valoare egală cu 12 pentru locusul stărilor. (Încercaţi o altă plasare care ar duce la o valoare mai mică). Utilizând această asignare rezultă tabelul de tranziţie al stărilor din Figura 3.28-c, în care s-au înscris valorile obţinute pentru biţii stării prezente z_2, z_1, z_0 şi biţii semnalelor de excitaţie w_2, w_1, w_0. Acest

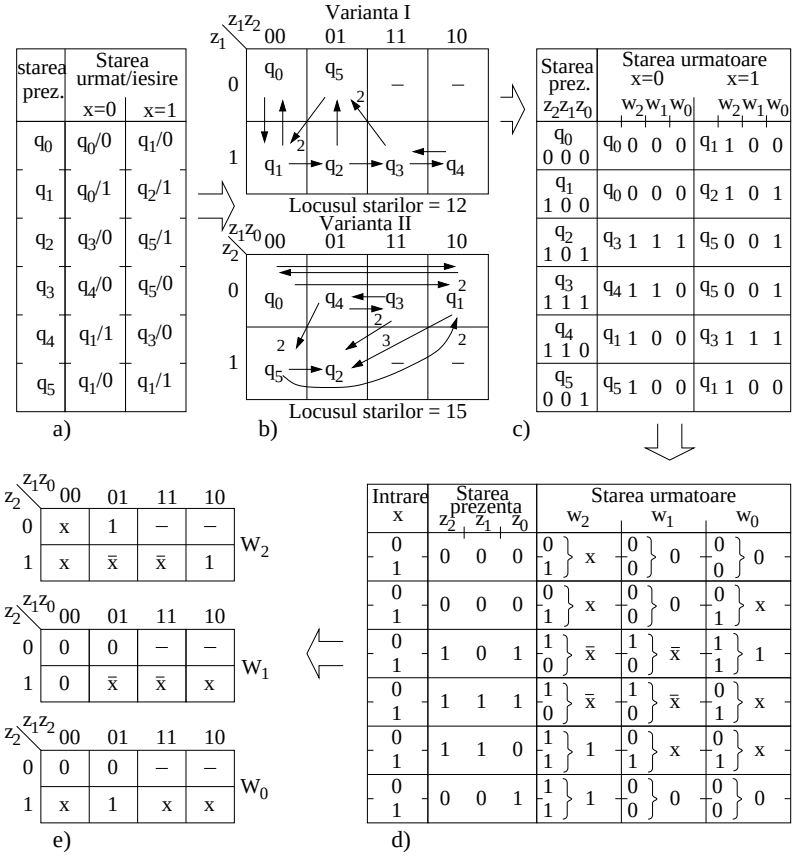


Figura 3.28 Exempificarea etapelor pentru sinteza expresilor semnalelor de excitaţie: a) tabelul de tranziţie al stărilor/ieşirilor; b) variante de mapare a stărilor pe diagrama V-K; c,d) explicitarea tabelului de tranziţie prin introducerea codurilor asigurate; e)diagramele V-k pentru determinarea expresiilor funcţiilor de excitaţie w_2, w_1 şi w_0.

tabel poate fi transformat în tabelul din Figura 3.28-d în care valorile biţilor de excitaţie au fost explicitate în funcţie de biţii stării prezente şi de valorile variabilei de intrare testată. Prin această explicitare, din acest tabel, se poate realiza sinteza expresiilor pentru w_2, w_1, w_0 ca funcţii de patru variabile: z_2, z_1, z_0 şi x. Sinteza se poate simplifica prin introducerea lui x ca variabilă reziduu în valorile pentru w_2, w_1, w_0 (notaţiile după acolade), rezultând diagramele de trei variabile z_2, z_1, z_0 reprezentate în Figura 3.28-e. Expresiile pentru semnalele de excitaţie sunt:

$$\begin{aligned} w_2 &= \overline{z_2} z_0 + z_1 \overline{z_0} + \overline{z_0} x + z_0 \overline{x} \\ w_1 &= z_2 z_0 \overline{x} + z_1 \overline{z_0} x \\ w_0 &= z_2 \overline{z_1} z_0 + z_2 x \end{aligned}$$

Pentru o implementare cu porţi discrete, introducând ca măsură a dimensiunii numărul S de intrări utilizate, rezultă respectiv valorile $11, 7$ şi 6, deci dimensiunea implementării Variantei I este $S_{VarI} = 24$

În Varianta II plasarea stărilor pe diagrama V-K s-a făcut fără a căuta obţinerea unei variaţii minime. S-au obţinut următoarele expresii pentru semnalele de excitaţie.

$$\begin{aligned} w_2 &= z_1x + z_2z_0x \\ w_1 &= \overline{z_2}\ \overline{z_1}z_0 + z_2\overline{z_0} + z_2\overline{x} \\ w_0 &= \overline{z_2}\ \overline{z_1}z_0x + z_1\overline{z_0}x + z_2z_1\overline{x} + z_2z_0\overline{x} \end{aligned}$$

care realizează respectiv dimensiunile $6, 9$ şi 16, iar pe întregul semiautomat $S_{VarII} = 31$.

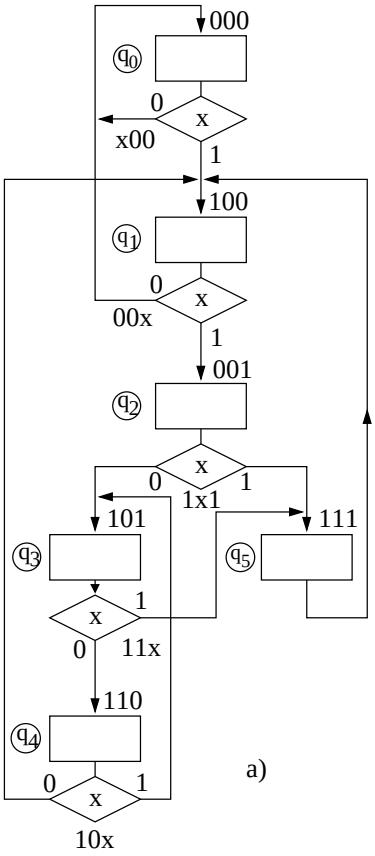
Codificarea cu dependenţă redusă. Selectarea codurilor, luând criteriu locusul stărilor, are în vedere doar modificarea valorilor variabilelor de stare şi efectul lor asupra funcţiilor de excitaţie(valorile variabilelor stării următoare). Dar, valorile variabilelor stării următoare sunt dependente şi de valorile variabilelor de intrare testate în simbolurile de decizie dintr-o organigramă ASM. Modalitatea de asignare a stărilor prin care să se ia în considerare şi contribuţia intrărilor în funcţiile de excitaţie este referită prin codificare cu dependenţă redusă(de variabilele de intrare). Codificarea cu dependenţa redusă presupune ca stările următoare ce se obţin prin tranziţia din aceeaşi stare, în urma testării unor variabile de intrare, să difere între ele printr-un singur bit. Aceasta înseamnă ca expresia celor două stări, care se obţin în urma testării unei singure variabile x_0, să se exprime cât mai simplu în funcţie de aceasta variabilă adică, să fie de forma, de exemplu, $z_4z_3x_0z_1z_0$. Cele două stări următoare sunt $z_4z_30z_1z_0$ şi $z_4z_31z_1z_0$. Dar dacă în acelaşi bloc de stare se testează şi a doua variabilă de intrare x_1 şi cuvintele care se obţin în urma acestei testări trebuie să fie diferite tot printr-un singur bit, de exemplu $z_4x_1z_2z_1z_0$, adică $z_40z_2z_1z_0$ şi $z_41z_2z_1z_0$. Încercarea de a codifica cu dependenţa redusă simultan după două variabile testate x_1, x_0 nu este posibilă. De exemplu, expresia $z_4x_1x_0z_1z_0$ pentru stările următoare poate genera următoarele patru cuvinte $z_400z_1z_0, z_401z_1z_0, z_410z_1z_0$ şi $z_411z_1z_0$ care nu diferă între ele numai printr-un singur bit. Deci codificarea cu dependenţa redusă poate fi realizată doar în funcţie numai de o variabilă testată.

Exemplul 3.11 Pentru automatul cu tabelul de tranziţie al stărilor/ieşirilor din Figura 3.28-a să se realizeze o codificare cu dependentă redusă.

Soluţie. Pentru a fi mai explicită această codificare din tabelul de tranziţie al stărilor s-a desenat organigrama ASM, Figura 3.29-a. În această organigramă după fiecare element de decizie s-a notat expresia codului pentru următoarele două stări în funcţie de valoarea variabilei testate x. De exemplu, în blocul de stare q_1 expresia codurilor stărilor următoare este $00x$ ceea ce impune pentru valoarea lui $x = 0$ codul următor să fie $000(q_0)$, iar pentru $x = 1$ codul următor să fie $001(q_2)$, între aceste două coduri distanţa de cod ste unitară. Pentru această codificare cu dependenţă redusă se obţine tabelul de tranziţie al stărilor din Figura 3.29-b. Urmând aceeaşi succesiune în sinteză ca la Exemplul 3.10 rezultă expresiile pentru semnalele de excitaţie

$$\begin{aligned} w_2 &= z_0 + z_1 + \overline{z_2}x \\ w_1 &= z_2z_0 + \overline{z_1}z_0x \\ w_0 &= z_2\overline{z_1}z_0 + z_2x \end{aligned}$$

pentru care dimensiunea implementării cu porţi discrete, considerând criteriu numărul de intrări, este $S = 18$. Se observă că în expresiile semnalelor de excitaţie w_2, w_1 şi w_0, variabila de intrare x intervine doar o singură dată(dependenţă redusă).



a)

Starea prezenta				Starea urmatoare x=0				Starea urmatoare x=1			
	z_2	z_1	z_0		w_2	w_1	w_0		w_2	w_1	w_0
q_0	0	0	0	q_0	0	0	0	q_1	1	0	1
q_1	1	0	0	q_0	0	0	0	q_2	0	0	1
q_2	0	0	1	q_3	1	0	1	q_5	1	1	1
q_3	1	0	1	q_4	1	1	0	q_5	1	1	1
q_4	1	1	0	q_1	1	0	0	q_3	1	0	1
q_5	1	1	1	q_1	1	0	0	q_1	1	0	0

b)

Figura 3.29 Codificare cu dependenţa redusă: a) organigrama ASM cu indicarea codurilor stărilor în funcţie de variabila de intrare testată, x; b) tabelul de tranziţie al stărilor.

Codificarea de tip unic-activ ("one-hot"). La codificarea de tip unic-activ cuvântul de cod nu mai are lungimea de $\lceil log_2 s \rceil$ ci are lungimea de s biţi, adică atâţia biţi în cuvântul de cod câte stări are automatul. O astfel de codificare pentru automatul cu şase stări, din Exemplele 3.10 şi 3.11, este cea prezentată în coloana a doua a Tabelului 3.3 şi care foloseşte doar şase cuvinte de cod din cele 2^6 cuvinte de cod posibile, fiecare cuvânt de cod având un singur bit egal cu unu("unu-activ"). La prima vedere am fi tentaţi să rejectăm acest tip de asignare din cauza creşterii dimensiunii registrului din calea de reacţie. Acest registru, pentru un automat cu s stări, va avea s celule de memorare(bistabile) şi nu $\lceil log_2\ s \rceil$ celule ca la asignările prezentate până acum. Dar, la o anliză mai atentă, văzând avantajele asignarea de tip unic-activ este preferată în raport cu alte asignări mai ales la implementările integrate.

Avantajul principal pentru asignarea one-hot conastă în simplitatea expresiilor celor s funcţii de excitaţie. Acestă simplitate duce în general la o reducere a dimensiunii circuitului combinaţional $CLC1$, Figura 3.25, care poate compensa într-o

Tabelul 3.3 Modalităţi de asignare a stărilor

Numele starii	Asignare prin : Codul unic activ						Codul unic activ					Codul cu initializare			
	z_5	z_4	z_3	z_2	z_1	z_0	z_4	z_3	z_2	z_1	z_0	z_3	z_2	z_1	z_0
q_0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
q_1	0	0	0	0	1	0	0	0	0	0	1	1	0	0	1
q_2	0	0	0	1	0	0	0	0	0	1	0	1	0	1	0
q_3	0	0	1	0	0	0	0	0	1	0	0	1	0	1	1
q_4	0	1	0	0	0	0	0	1	0	0	0	1	1	0	0
q_5	1	0	0	0	0	0	1	0	0	0	0	1	1	0	1

oarecare măsură creşterea dimensiunii părţii ordonate corespunzătoare registrului. Pe un circuit integrat este de preferat creşterea, intr-o oarecare măsură, a dimensiunii părţii ordonate (registru) în schimbul scăderii dimensiunii părţii neordonate (CLC). În plus, prin scăderea dimensiunii părţii combinaţionale, viteza de răspuns a automatului creşte. De asemenea, acestă simpitate duce la o sinteză mult mai uşoară şi, în depanarea circuitului, necesită un timp mult mai redus. Pentru asignarea one-hot, din Tabelul 3.3, aplicată automatului cu tabelul de tranziţie al stărilor din Figura 3.28-a se obţin următoarele expresii pentru semnalele de excitaţie.

$$w_5 = \overline{\overline{z_2}\ \overline{z_3}}x \qquad w_2 = z_1x$$

$$w_4 = z_3\overline{x} \qquad w_1 = z_5 + z_4\overline{x} + \overline{z_2}z_0x$$

$$w_3 = z_2\overline{x} + z_4x \qquad w_0 = \overline{\overline{z_1}z_2\overline{z_0}}x$$

Există o practică, destul de frecventă, în proiectarea automatelor ca să se introducă şi o stare de aşteptare "idle" care se codifică printr-un cuvânt numai de zero-uri sau mai rar numai din unu-uri. Codificarea numai prin zero-uri este practică deoarece printr-un semnal de RESET-are se poate înscrie acest cuvânt în registrul de stare şi automatul este adus în starea de aşteptare. În această stare de iniţializare/"idle", q_0 din organigrama din Figura 3.29, se ajunge fie prin iniţializare, la punerea sub tensiune, sau fie prin resetare când automatul nu mai are nimic de efectuat. Introducând această starea de iniţializare pentru asignarea de tip one-hot se obţine asignarea de tip one-hot modificat cu o reprezentare ca cea din coloana a treia a Tabelului 3.3. Funcţiile de excitaţie, în număr de $(s-1)$, ale automatului descris în Figura 3.29-a, prin aplicarea asigurării one-hot modificat au următoarele expresii

$$w_4 = (\overline{z_1z_2}x) \qquad w_1 = z_0x$$

$$w_3 = z_2\overline{x} \qquad w_0 = z_4 + z_3\overline{x} + \overline{z_4}\ \overline{z_3}\ \overline{z_2}\ \overline{z_1}\ \overline{z_0}x$$

$$w_2 = z_1\overline{x} + z_3x$$

Al treilea termen din expresia lui w_0 indică faptul că automatul este adus în starea $q_0(= 00\ 000)$ când nici una din celelalte stări nu este activă.

Un exemplu de iniţializare $q_0(= 0\ 000)$ pentru o codificare obişnuită este prezentat în coloana a patra din Tabelul 3.3. Toate stările, în afară de cea de iniţializare, au în codul lor bitul cel mai semnificativ z_3 egal cu 1, deci acest bit prin valoarea 1 indică faptul că automatul nu este în starea de iniţializare. În acest exemplu pentru ceilalţi trei biţi de stare $z_2z_1z_0$ se foloseşte în codificarea stărilor o numărare în binar natural 000, 001, 010 etc.

Pentru toate modalităţiile de asignare a stărilor prezente, în general, numărul de coduri utilizate (valide) s este mai mic decât numărul total de cuvinte de cod posibile. Diferenţa între aceste două numere este numărul de coduri neutilizate (ilegale). Dar ce influenţă negativă pot avea codurile neutilizate? Datorită unor funcţionări defectuoase ale circuitului, unor greşeli de proiectare, modificarea neaşteptată a unei intrări(asincrone) sau la punerea sub tensiune, automatul poate intra într-o stare ilegală. Odata intrat într-o stare ilegală pot exista două scenarii. Primul, funcţie dacă starea ilegală a fost sau nu cuprinsă în procesul de minimizare şi dacă a fost ce valoare a primit în diagrama V-K, după câteva tacturi de clock se poate ajunge intr-o stare legală şi mai departe o funcţionare corectă a automatului. Al doilea, şi cel mai dezastruos, dintr-o stare ilegală se poate trece în alte stări ilegale, inchizându-se un ciclu între acestea iar din această funcţionare "ilegală" se poate iesi numai printr-o oprire şi o repornire, dacă nu se ajunge din nou tot în funcţionare ilegală. În tratarea stărilor ilegale, notate prin simbolul indiferent (don′t care) în diagramele V-K din Figura 3.28-b şi c, pot exista două abordări.

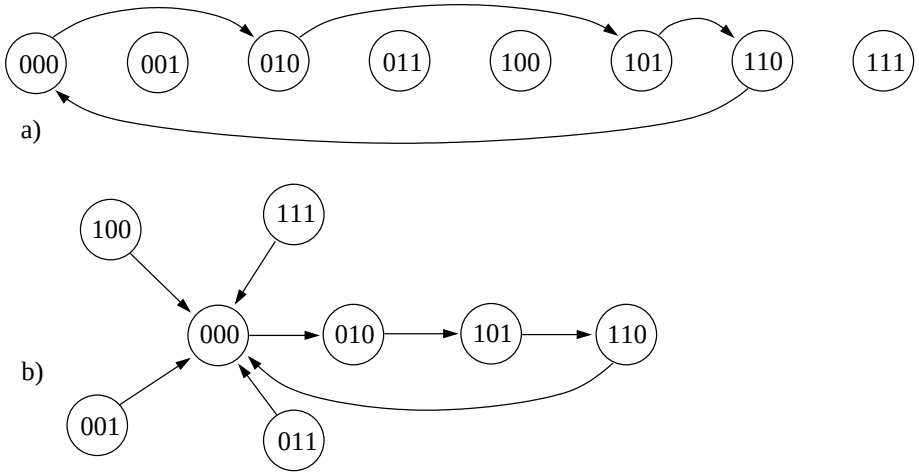


Figura 3.30 Variante de abordare a proiectării automatului în raport cu considerarea stărilor ilegale a) proiectare pentru cost minim; b) proiectare cu risc minim.

Prima abordare a **- costului minim -** consideră că automatul nu va intra niciodată intr-o stare ilegală. Prin urmare, în procesul de sinteza pe diagramele V-K termenii canonici indiferenţi sunt incluşi cu valoarea potrivită împreună cu alţi termeni canonici adiacenţi astfel încât să rezulte implicanţi primi esenţiali de suprafaţă cât mai mare, în consecinţă cost minim de implementare.

A doua abordare **- riscul minim -** evită cantonarea automatului într-o stare

ilegală. Această funcţionare se poate realiza prin proiectarea automatului astfel încât din fiecare stare ilegală să existe o tranziţie într-o stare legală, în general spre starea de iniţializare (de cod $00\ldots00$). De exemplu, automatul din Figura 3.30-a utilizează numai patru din cele opt cuvinte de cod, celelalte patru cuvinte de cod constituie stări ilegale ale automatului. O proiectare cu risc minim va considera graful de tranziţie al stărilor din Figura 3.30-b în care din stările ilegale $001, 011, 100$ şi 111 există tranziţii spre starea de iniţializare 000.

3.2.5.1 Intrări şi ieşiri asincrone

La un automat sincron variaţiile variabilelor de intrare au loc doar în momentul aplicării impulsului de ceas, sunt sincrone (sunt eşantionate/testate) cu semnalul de ceas. Aceste variaţii împreună cu cele ale stării prezente produc starea următoare şi ieşirile. Atât ieşirile cât şi starea următoare vor fi corecte dacă se citesc, respectiv se înscriu în registrul de stare şi de ieşire, numai în perioada regimului stabilizat, adică după consumarea regimului tranzitoriu, vezi Figura 3.7-b. Chiar şi la un automat Mealy imediat se obţin ieşiri corecte dacă sunt citite (utilizate) după consumarea regimului tranzitoriu. Nu la fel se întâmplă când intrările sunt asincrone.

Semnalele asincrone de intrare nu mai au variaţii sincrone cu ceasul automatului, aceste semnale se pot modifica oricând în raport cu frontul activ de sincronizare al ceasului. Semnalele asincrone pe intrare pot proveni de la un alt sistem digital care este comandat cu un ceas diferit sau sunt culese din exterior de la traductori ai unor sisteme mecanice, termice, biologice etc. Variaţia semnalelor asincrone va produce modificări în ieşirile de tip imediat (Mealy) care pot fi încărcate de hazard dacă sunt citite înainte de consumarea regimului tranzitoriu. În schimb, variaţia semnalelor asincrone de intrare nu va produce o funcţionare incorectă a automatului, prin ieşiri încărcate de hazard şi cuvinte de stare următoare eronate, dacă înscrierea (memorarea) în registrul de ieşire şi în registrul de stare se face numai după consumarea regimului tranzitoriu. S-a arătat în secţiunea 3.2.1, Figura 3.6, că înscrierea într-un registru, la momentul iT al aplicării impulsului de ceas, al unui semnal care variază în intevalul interzis Δ $(iT - \tau_{\text{SU}}, iT + \tau_{\text{H}})$, nu poate fi o operaţie deterministă. Dacă în acest interval semnalul are o variaţie de la 0 la 1 sau de la 1 la 0 în registru nu se poate şti exact dacă valoarea înscrisă este 0 sau 1. Această comportare nedeterministă se datorează funcţionării celulelor (bistabile) din care este construit registrul, explicaţiile vor fi date în secţiunea 3.3.1.

În consecinţă, daca variaţia semnalului asincron de intrare produce modificări ale ieşirii şi ale cuvântului de stare următoare chiar în intervalul Δ atunci ieşirea înscrisă în registrul de ieşire poate fi eronată iar codul înscris în registrul de stare poate fi unul ilegal sau poate fi unul care nu mai corespunde unei tranziţii normale. Deci, iată cum o intrare asincronă, prin efectele sale, poate produce o funcţionare incorectă a unui automat. Efectele variaţiei variabilei asincrone în intervalul interzis Δ nu pot fi eliminate dar pot fi atenuate printr-o asignare corespunzătoare a stărilor.

Pentru segmentul de organigramă din Figura 3.31-a, în care se testează şi variabila asincronă x_0^* (asincronismul unei intrări se notează, în general, cu un asterisc), asignarea s-a realizat cu variaţie minimă astfel încât locusul stărilor pentru acest segment are valoarea 3. Expresia cuvântului de cod pentru stările q_2, q_3 şi q_4, în funcţie de variabilele testate, x_1 sincronă şi x_0^* asincronă, este $((\overline{x_1}x_0^*)\ (\overline{x_1}\ \overline{x}_0^*)x_1)$

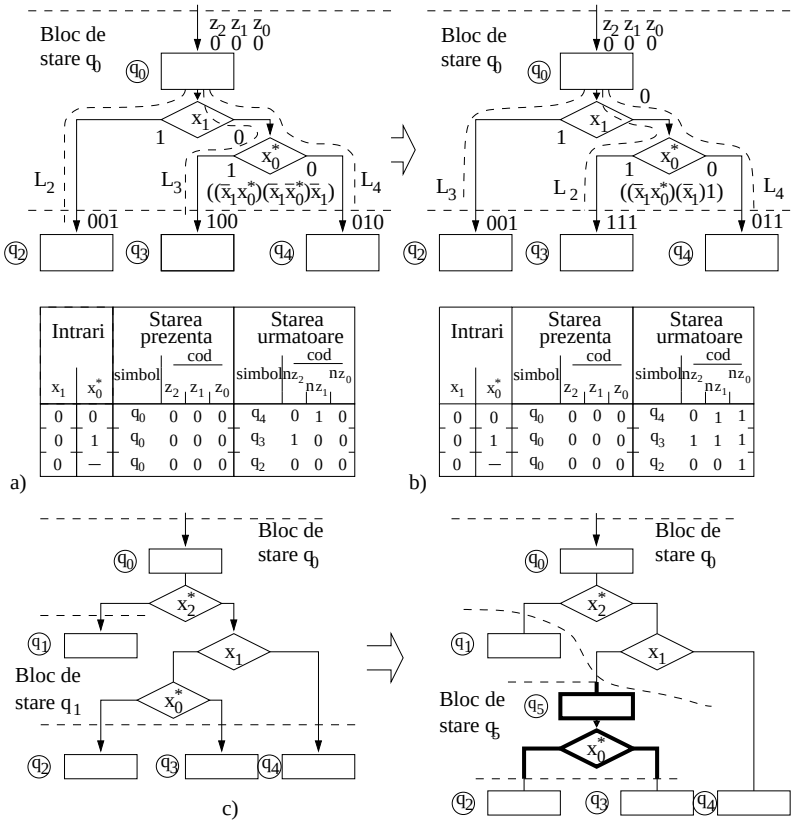


Figura 3.31 Analiza ambiguităţii aspra înscrierii cuvântului de stare următoare generată de testarea unei variabile de intrare asincrona: a) printr-o codificare cu variaţie minimă; b) printr-o codificare cu dependenţa redusă; c) introducerea unui bloc de stare suplimentar pentru eliminarea testării simultane a două variabile asincrone.

adică: $nz_2 = \overline{x}_1 x_0^*$, $nz_1 = \overline{x}_1 \overline{x}_0^*$, $nz_0 = x_1$. Considerăm cazul când $x_1 = 0$, adică o tranziţie din q_0 în q_3 sau q_4, iar cuvântul de cod al acestor două stări se reduce la $x^* \overline{x}_0^* 1$, deci $nz_2 = x_0^*$, $nz_1 = \overline{x}_0^*$ şi $nz_3 = 1$. Să presupunem că în intervalul Δ centrat pe frontul impulsului de ceas din momentul $(i+1)T$, care marchează trecerea din blocul de stare q_0 în unul din blocurile de stare q_3 sau q_4, variabila asincronă x_0^* variază de la valoarea 0 la valoarea 1. Conform celor prezentate anterior, în fiecare din cele două celule (z_2, z_1) ale registrului de stare, pentru biţii cuvântului de stare nz_2 şi nz_1, se poate înscrie fie 0, fie 1 neavând un control determinist asupra acestui proces, în consecinţă pot fi următoarele patru cuvinte înscrise pentru starea următoare: 000, 010, 100 şi 110.

1. Pentru cuvântul 100 înscris în registrul de stare: s-a realizat o comutaţie corectă atât în celula $z_2 = 1$ $(nz_2 = x_0^* = 1)$ cât şi în celula $z_1 = 0$ $(nz_1 = \overline{x}_0^*)$, deci o tranziţie corectă $(L_3 = \overline{x}_1 x_0^* = \overline{0} \cdot 1 = 1)$ într-o stare legală q_3.

2. Pentru cuvântul 010 înscris în registrul de stare: s-a realizat o comutaţie incorectă atât în celula $z_2 = 0$ ($nz_2 = x_0^* = 1$) cât şi în celula $z_1 = 1$ ($nz_1 = \overline{x}_0^* = 0$), deci o tranziţie incorectă în starea q_4 (legală), pe o linie de tranziţie dar care nu a fost activată ($L_4 = \overline{x}_1 \cdot \overline{x}_0^* = \overline{0} \cdot \overline{1} = 0$).

3. Pentru cuvântul 000 înscris în registrul de stare: s-a realizat o comutaţie incorectă în celula $z_2 = 0$ dar o comutaţie corectă în celula z_1 ($nz_1 = \overline{x}_0^* = 0$), deci o tranziţie incorectă în starea q_0, care este o stare legală a automatului dar pentru care nu există nici o linie de tranziţie, din L_0 în L_0.

4. Pentru cuvântul 110 înscris în registrul de stare: s-a realizat o comutaţie corectă în celula $z_2 = 1$ dar o comutaţie incorectă în celula $z_1 = 1$ ($nz_1 = \overline{x}_0^* = 0$), deci o tranziţie incorectă într-o stare care nici nu este legală pentru automat.

În concluzie, o codificare cu variaţie minimă nu poate atenua efectele produse de modificarea unei variabile asincrone în intervalul Δ centrat pe un front al semnalului de ceas.

Codificarea stărilor cu dependenţă redusă faţă de o intrare asicronă poate atenua influenţa pe care o are aceasta asupra comutaţiei stării automatului. În Figura 3.31-b s-a realizat segmentul de organigramă ASM, analizat anterior, dar de data aceasta printr-o codificare cu dependenţa redusă faţă de variabila asicronă x_0^*. Evident, nu se poate realiza simultan o codificare cu dependenţă redusă după două variabile testate, adică atât după x_1 cât şi după x_0^* (vezi secţiunea 3.25). Pentru $x_1 = 0$ rezultă tranziţiile la stările q_3 şi q_4 ale căror cuvinte de cod se pot exprima prin expresia $(\overline{x}_1 x_0^*)(\overline{x}_1)(1) = x_0^* 11$, deci numai un singur bit $nz_2 = x_0^*$ este dependent de valoarea variabilei asincrone testate. Considerând ca anterior, variaţia variabilei x_0^* de la 0 la 1, rezultă că pot fi înscrise în registrul de stare următoarele două cuvinte de cod 011 şi 111. Când se înscrie 111 este o tranziţie la starea q_3 pe o cale de tranziţie corectă (L_3), iar când se înscrie 011 este o tranziţie la q_4 pe o cale de tranziţie incorectă, L_4 (dar nu ca în cazul codificării cu variaţie minimă când apăreau căi de tranziţie inexistente sau chiar stări ilegale). Evident, tranziţia la q_4 este incorectă dar care are consecinţe atenuante. Dacă tranziţia a fost greşită la q_4 şi dacă până la următorul impuls de clock $(i+2)T$ intrarea x_0^* revine la valoarea 0, se consideră că a fost un spike în această intrare, tranziţia a fost corectă (neglijând spike-ul). Dar dacă variabila x_0^* se menţine la valoarea 1 acum se testează această valoare în blocul de stare al lui q_4 deci evenimentul pe intrare a fost sesizat cu o întârziere de un tact. Uneori, se poate realiza achiziţia semnalelor de intrare încât acestea să nu se modifice la intervale de timp mai scurte decât două perioade ale impulsului de ceas.

Dar dacă într-un bloc de stare se testează două sau mai multe variabile asincrone cum se pot atenua efectele modificării simultane a acestor variabile asupra ambiguităţii care poate apărea la înscrierea cuvântului de stare următoare? Pentru aceste cazuri, deoarece nu se poate realiza o codificare cu dependenţă redusă simultan după două sau mai multe variabile de intrare testate în acelaşi bloc de stare, soluţia este introducerea de stări suplimentare şi testarea într-un bloc de stare doar a unei singure variabile asincrone. În Figura 3.31-c se testează în blocul de stare q_0 două variabile de intrare asincrone x_2^* şi x_0^*. Prin introducerea stării suplimentare q_5 se transferă testarea variabilei asincrone x_0^* în blocul de stare corespunzător lui q_5. Trebuie analizat ce ieşire se va genera în blocul de stare q_5 pentru ca să nu se modifice funcţionarea

automatului(de dorit să se genereze aceeaşi ieşire ca şi în blocul de stare q_0). Apare a doua întrebare referitoare la existenţa mai multor variabile asincrone testate, prin secvenţializarea lor în blocuri de stare succesive nu este pierdută semnificaţia lor? Răspunsul este negativ deoarece variabilele asincrone nu sunt percepute în automat, prin testare, ca formând un cuvânt de cod ci numai ca semnale cu interpretare independentă (fără o corelare intre ele).

Codificarea cu dependenţă redusă, pentru evitarea ambiguităţii tranziţiei stărilor, în procesul de sinteză a automatului se va reflecta într-o anumită structură obţinută pentru circuitul combinaţional CLC1, Figura 3.25, care calculează funcţiile de excitaţie. Dar aspecte de funcţionare incorectă pot apare nu numai pentru calculul stării următoare ci şi pentru calculul ieşirilor care se realizează cu partea combinaţională notată prin CLC2 pe Figura 3.25.

Pentru un automat Mealy imediat, Figura 3.25, 3.8-a, chiar şi când intrările sunt sincrone, se obţin aproape totdeauna ieşiri care sunt încărcate cu hazard. La ieşirea circuitului CLC2, realizat pe două niveluri logice, nu se produc semnale încărcate cu hazard numai dacă în configuraţia de intrare se modifică cel mult un bit. Dar configuraţia de intrare este definită pe produsul cartezian $X \times Q$ ceea ce înseamnă că, uzual, se modifică doi biţi unul pe intrare şi unul în cuvântul de stare. Dacă există şi intrare de tip asincron variaţiile acesteia se transmit direct în ieşiri ce pot fi încărcate cu hazard pe durata tranzitorie. Funcţionarea incorectă a automatului Mealy imediat poate fi evitată dacă acesta este transformat într-un automat Mealy cu întârziere, Figura 3.8-b. Deoarece la automatul Mealy cu întârziere ieşirile sunt disponibile (după înscrierea în registrul de ieşire R_2 când s-a consumat regimul tranzitoriu) numai la urmatorul impuls de clock; trebuie ţinut cont că efectul unei intrări se manifestă la ieşire cu o întârziere de un tact. Dar dacă există o intrare asincronă aspectul de ambiguitate discutat anterior,la înscrierea biţilor în registrul de stare, apare şi aici la înscrierea(memorarea) ieşirilor în registrul de ieşire R_2. Această ambiguitate apare atunci când intrarea asincronă are o modificare în intervalul interzis al impulsului de ceas aplicat registrului de ieşire R_2. În concluzie pentru un automat Mealy cu întârziere şi cu cel puţin o intrare asincronă nu sunt garantate semnale de ieşire corecte. Ambiguitatea pe ieşire la un Mealy cu întârziere, determinată de intrări asincrone, poate fi eliminată dacă automatul este transformat într-un automat Moore imediat echivalent.

Automatul Moore imediat, Figura 3.8-c, generează, la fel ca şi Mealy cu întârziere, ieşiri care sunt dependente de intrarea aplicată în tactul anterior. Intrările asincrone la automatul Mealy imediat nu generează ambiguitate pe ieşiri deoarece acestea nu sunt cuplate direct cu intrările, modificarea intrărilor se transmite la ieşire prin intermediul stării automatului. Dar automatul Moore imediat poate genera ieşiri încărcate cu hazard. Hazardul se datorează circuitului combinaţional CLC2 de pe ieşire, ce produce hazard atunci când în configuraţia sa de intrare, care este cuvântul de stare prezenta, se modifică mai mult de un bit. Se poate elimina hazardul din ieşire prin două modalităţi de implementare. Prima, se elimină circuitul CLC2, adică ieşirile sunt identice cu starea $Y \equiv Q$ (caz uzual la circuitele numărător). A doua modalitate constă în transformarea automatului Moore imediat într-un automat Moore cu întârziere, Figura 3.8-d. Dar la automatul Moore cu întârziere efectul modificării intrării se manifestă la ieşire cu o întârziere de două tacturi.

3.2.6 Proiectarea automatelor sincrone

Definirea unui automat cu s stări necesită descrierea funcţionării automatului în raport cu fiecare din aceste stări, deci, evident, complexitatea automatului $C_{automat}$ - dimensiunea definiţiei - are ordinul de mărime $C_{automat} \in O(s)$. Pentru un automat cu s stări cea mai mică valoare a numărului de intrări secundare este egală $k = \lceil log_2\ s \rceil$. Pentru n intrări principale dimensiunea circuitelor CLC1 şi CLC2, din Figura 3.25, depinde de numărul total de intrări $n + k$. Neglijând numărul de intrări principale, $(n = 0)$, în cazul general dimensiunea circuitelor combinaţionale ale automatului sunt proporţionale cu 2^k (dar $s = 2^k$) deci $C_{automat} \in (2^k)$. În concluzie, rezultă că pentru un automat complexitatea sa este în acelaşi ordin de mărime cu dimensiunea circuitului combinaţional asociat. Această concluzie sugerează faptul că proiectarea unui automat de complexitate redusă implică realizarea unei structuri reduse de circuit combinaţional.

Reducerea dimensiunii părţii combinaţionale a unui automat, ce face parte dintr-un sistem digital, direcţionează proiectarea spre divizarea într-o cale de date şi o cale de control (automat de control). Dar chiar şi prin această reducere poate rezulta pentru un automat de control, sau pentru automatele funcţionale - care implementează diferite funcţii, de o complexitate destul de ridicată. Soluţia pentru reducerea acestei complexităţi este partajarea în mai multe automate de complexitate mai redusă şi apoi interconectarea acestora, Figura 3.20 şi 3.21

În proiectarea unui automat sincron se parcurg următoarele etape:

1. Descrierea funcţionării automatului. Această descriere poate fi a unui automat complex dar care prin partajare duce la mai multe automate fiecare definit apoi separat sau poate fi descrierea unui automat autonom. Descrierea poate fi direct sub forma unei organigrame ASM, a unui graf de tranziţie al stărilor/ieşirilor sau se transpune sub una din aceste forme dintr-o descriere verbală.

2. Construirea tabelului de tranziţie al stărilor/ieşirilor şi reducerea stărilor redundante.

3. Codificarea stărilor şi construirea tabelului de tranziţie asignat pentru stări şi ieşiri.

4. Sinteza funcţiilor de excitaţie (starea următoare) şi a ieşirilor.

5. Implementarea, într-o anumită tehnologie, şi apoi testarea.

6. Elaborarea documentaţiei.

Operaţiile cuprinse în aceste etape au fost prezentate în secţiunile anterioare acum vor fi reluate în cadrul unui exemplu:

Exemplul 3.12 Să se realizeze sinteza automatului sincron descris prin organigrama ASM din Figura 3.32-a.

Soluţie. Automatul prezintă trei intrări x_2, x_1, x_0 şi cinci ieşiri y_1, y_{2_L}, y_4 ieşiri imediate condiţionate(de tip Mealy) şi y_3, y_5 ieşiri imediate dependente de stare(de tip Moore). Codul celor cinci stari $q_0 \div q_5$ rezultă prin plasarea acestora pe o diagrama V-K, Figura 3.32-b. Se descompune diagrama ASM în 6 blocuri de stare $B_0 \div B_5$ în care se identifică 11 căi de tranziţie $L_1 \div L_{11}$. În Figura 3.32-c se calculează valoarea locusului stărilor, care este

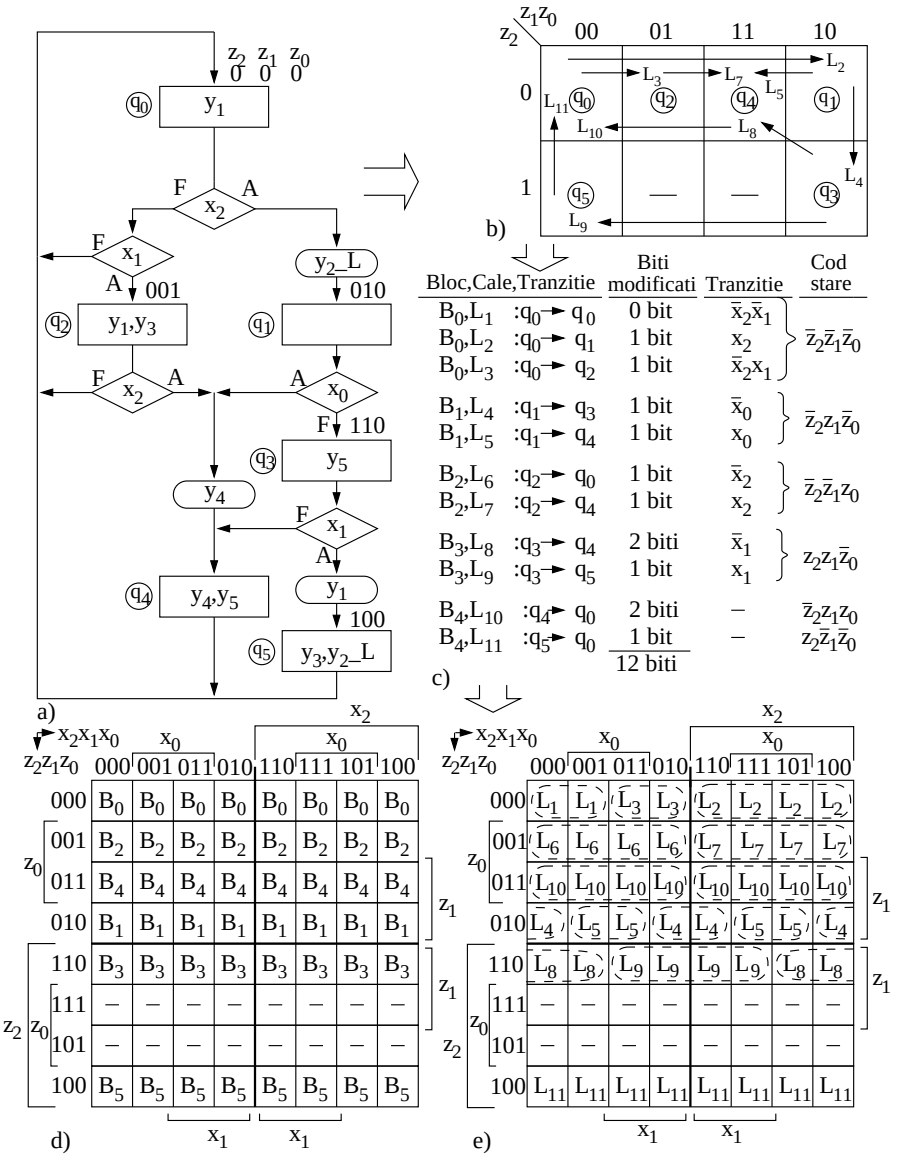


Figura 3.32 Exemplu de extragere dintr-o organigrama ASM (a) a blocurilor de stare, a căilor de tranziţie cu expresiile logice corespunzătoare (c) şi plasarea stărilor pe o diagramă V-K (b) pentru obţinerea unui minim pentru locusul stărilor. Contribuţia blocurilor de stare (d) şi a căilor de tranziţie (e) la descrierea automatului.

egală cu 12, şi pentru fiecare cale de tranziţie şi pentru fiecare stare este prezentată expresia logică. Apoi, pentru o sistematizare şi explicitare a procesului de sinteză în Figura 3.32-d şi Figura 3.32-c sunt prezentate respectiv contribuţiile blocurilor şi căilor de tranziţie la descrierea automatului. Tabelul de tranziţie al stărilor/ieşirilor (de tip Mealy) cu 6 stări şi 3 intrări ar cuprinde $6 \cdot 2^3 = 48$ de elemente. Dar pentru că nu există tranziţii între stări pentru toate configuraţiile de intrare posibile, tabelul de tranziţie (complet) se va substitui cu o forma simplificată de tabel care cuprinde 11 linii, câte una pentru fiecare cale de tranziţie, şi două coloane (una pentru starea prezentă şi alta pentru starea următoare, similar se va realiza un tabel şi pentru ieşiri). Tabelele se vor construi sub formă simbolică şi apoi sub formă asignată.

Tabelul de tranziţie al stărilor sub formă simbolica şi asignată este prezentat în Figura 3.33-a. Pentru ieşiri se disting două tabele unul pentru ieşiri dependente numai de stare (de tip Moore) şi unul pentru ieşiri condiţionate (de tip Mealy). Tabelul pentru ieşirile de tip Moore y_3, y_5, Figura 3.33-b, cuprinde doar stările şi valorile corespunzătoare ale ieşirilor. În schimb, tabelul pentru ieşirile de tip Mealy, Figura 3.33-c, este construit, ca şi cel al stărilor, pentru toate căile de tranziţie şi stările respective. Dar ieşirile condiţionate y_1, y_{2_L}, y_1, figurate în simbolurile de ieşiri condiţionate, sunt prezente şi în simbolurile de stare: y_1 în q_0 şi q_2, y_{2_L} în q_5 şi y_4 în q_4. O ieşire de tip Moore, figurată într-o stare, poate fi convertită formal într-o ieşire condiţionată dacă acea ieşire este introdusă pe fiecare cale de tranziţie care porneşte din acea stare. Procedând în acest fel ieşirile y_1, y_{2_L} şi y_4 vor fi tratate ca fiind active pe toate căile de tranziţie care pornesc respectiv din blocurile de stare B_0, B_2, B_5 şi B_4, în consecinţă se introduc în tabel pe toate aceste căi. Deci, pentru sinteză, aceste ieşiri nu mai sunt reprezentate şi de tip Mealy şi de tip Moore, ci numai de tip Mealy. Concentrând în acelaşi tabel atât tranziţia stărilor cât şi ieşirile condiţionate şi de stare se obţine forma reprezentată în Figura 3.34. Utilizând valorile din acest tabel, transcrise pe diagrama V-K, se pot deduce expresiile pentru ieşiri şi pentru funcţiile de excitaţie (biţii stării urmatoare).

Pentru ieşirile dependente de stare y_3 şi y_5 expresiile logice se obţin dintr-un tabel V-K, cu opt căsuţe, de variabile z_2, z_1 şi z_0.

$$y_3 = \overline{z_1}(z_2 + z_0)$$
$$y_5 = z_1(z_2 + z_0)$$

Funcţiile de excitaţie w_2, w_1, w_0 şi ieşirile condiţionate y_4, y_{2_L}, y_1 sunt dependente de şase variabile, $x_2, x_1, x_0, z_2, z_1, z_0$ deci se obţin din tabele V-K cu 64 căsuţe. Deoarece acestea au fost explicitate doar în funcţie de 11 căi de tranziţie expresiile lor se pot deduce mai uşor ca sumă logică numai de acele căi pentru care au valoarea 1, iar pentru minimizare se utilizează diagrama V-K din Figura 3.32-e unde s-au mapat deja contribuţiile fiecărei căi de tranziţie. Se obţin următoarele expresii:

$w_2 = L_4 + L_9 = \overline{z_2} z_1 \overline{z_0} x_0 + z_2 z_1 x_1$

$w_1 = L_2 + L_4 + L_5 + L_7 + L_8 = \overline{z_2}\ \overline{z_1} x_2 + \overline{z_2} z_1 \overline{z_0} + z_2 z_1 \overline{x_1}$

$w_0 = L_3 + L_6 + L_7 + L_8 = \overline{z_2}\ \overline{z_1}\ \overline{z_0}\ \overline{x_2} x_1 + \overline{z_2}\ \overline{z_1} z_0 + z_2 z_1 \overline{x_1}$

$y_4 = L_5 + L_7 + L_{10} = \overline{z_2} z_1 x_0 + \overline{z_1} z_0 x_2 + z_1 z_0$

$y_{2\_L} = L_1 + L_3 + L_4 + L_5 + L_6 + L_7 + L_8 + L_9 + L_{11} = \overline{L_2 + L_{10}} = \overline{\overline{z_2}\ \overline{z_1}\ \overline{z_0} x_2 + z_1 z_0}$

$y_1 = L_1 + L_2 + L_3 + L_6 + L_7 + L_9 = \overline{z_2}\ \overline{z_1} + z_2 z_1 x_1$

În acelaşi mod se pot deduce şi expresiile pentru y_3 şi y_5

$y_3 = L_6 + L_7 + L_{11} = \overline{z_1} z_0 + z_2 \overline{z_1} = \overline{z_1}(z_2 + z_0)$

$y_5 = L_8 + L_9 + L_10 = z_2 z_1 + z_1 z_0 = z_1(z_2 + z_0)$

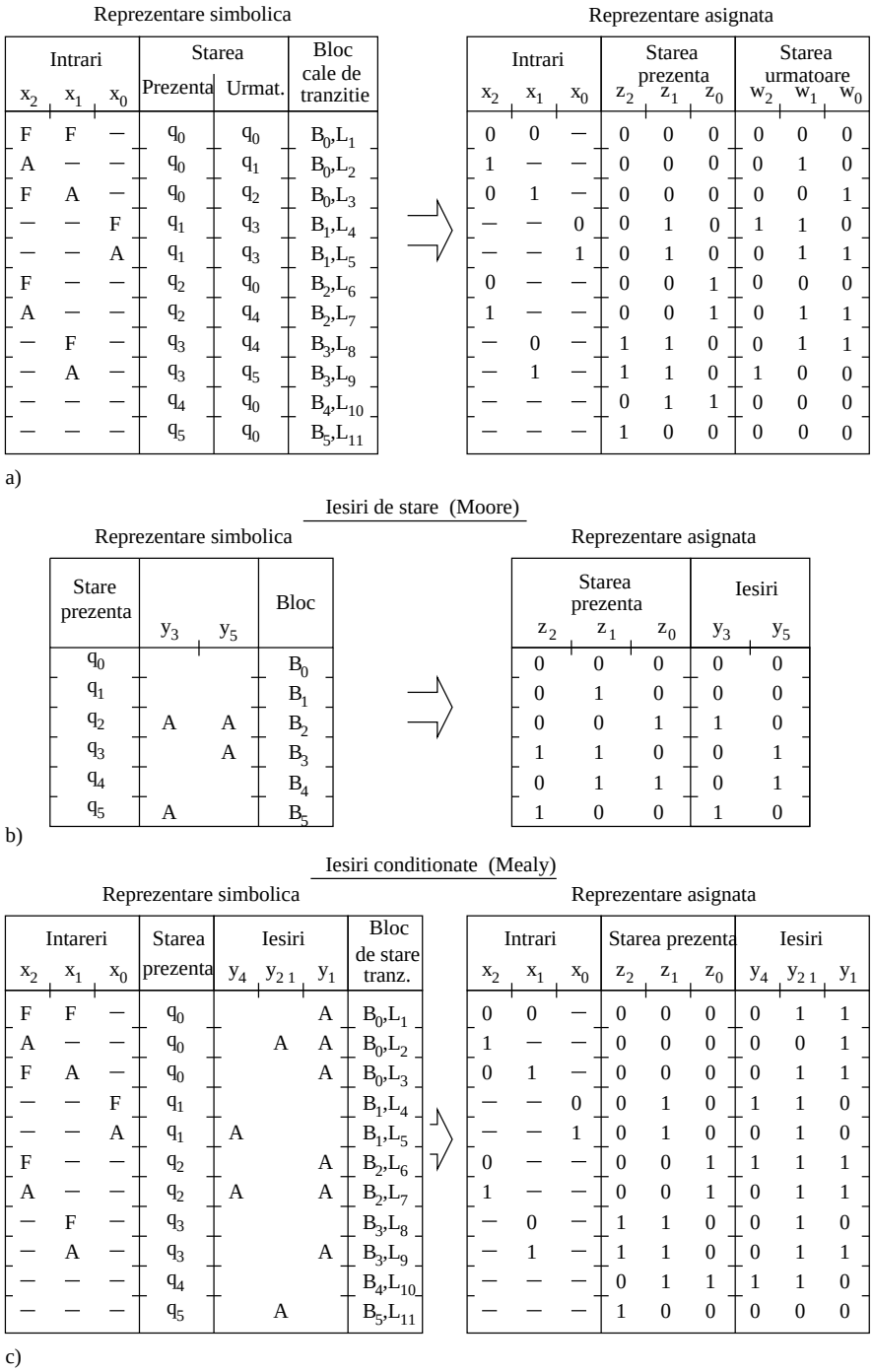


Figura 3.33 Tabelele de tranziţie în reprezentare simbolică şi asignată pentru stări (a) pentru ieşirile dependente doar de stare (b) şi pentru ieşirile condiţionate (c)

calea de tranz.	Intrari			Starea prezenta				Starea urmatoare				Iesiri dependente de stare		Iesiri conditionate		
	x_2	x_1	x_0	simbol	z_2	z_1	z_0	simbol	w_2	w_1	w_0	y_3	y_5	y_4	$y_{2\,1}$	y_1
L_1	0	0	–	q_0	0	0	0	q_0	0	0	0	0	0	0	1	1
L_2	1	–	–	q_0	0	0	0	q_1	0	1	0	0	0	0	0	1
L_3	0	1	–	q_0	0	0	0	q_2	0	0	1	0	0	0	1	1
L_4	–	–	0	q_1	0	1	0	q_3	1	1	0	0	0	0	1	0
L_5	–	–	1	q_1	0	1	0	q_4	0	1	1	0	0	1	1	0
L_6	0	–	–	q_2	0	0	1	q_0	0	0	0	1	0	0	1	1
L_7	1	–	–	q_2	0	0	1	q_4	0	1	1	1	0	1	1	1
L_8	–	0	–	q_3	1	1	0	q_4	0	1	1	0	1	0	1	0
L_9	–	1	–	q_3	1	1	0	q_5	1	0	0	0	1	0	1	1
L_{10}	–	–	–	q_4	0	1	1	q_0	0	0	0	0	1	1	0	0
L_{11}	–	–	–	q_5	1	0	0	q_0	0	0	0	1	0	0	1	0

Figura 3.34 Tabelul combinat pentru stări, ieşiri dependente de stare şi ieşiri condiţionate.

Aceste expresii pot fi implementate prin oricare din variantele de realizare a unui circuit logic combinaţional: porţi logice, DMUX+nivel exterior de SAU adăugat, MUX, circuite ROM sau circuite PLA. Organizarea automatului, în care s-au separat circuitele pentru: calculul funcţiei de tranziţie CLC1, pentru calculul ieşirilor de tip Mealy CLC2 şi pentru calculul ieşirilor de tip Moore, este prezentată în Figura 3.35-a. Expresiile obţinute anterior prin minimizare necesită în total realizarea a 16 produse logice distincte de două până la cinci variabile, dar numai trei produse logice din cele 19 sunt utilizabile de două ori ($z_2 z_1 x_1$, $z_2 z_1 \overline{x_1}$ şi $z_1 z_0$).

Implementarea automatului pe o matrice PLA, reprezentată generic în Figura 3.35-b, poate fi realizată, mai simplu, fără minimizarea expresiilor logice. În această implementare fiecare poartă AND realizează un termen produs logic dintre expresia logică a unei căi de tranziţie şi expresia logică a cuvântului de cod al stării din care porneşte calea de tranziţie respectivă, Figura 3.32-c. De exemplu, în blocul de stare B_2 expresia căii de tranziţie L_7 este x iar expresia cuvântului de stare este $\overline{z_2}\,\overline{z_1} z_0$, deci pe coloana notată cu L_7 din matricea programabilă AND a circuitului PLA se programează produsul logic $x\overline{z_2}\;\overline{z_1} z_0$. Procedând în acest mod, fără a mai fi necesară minimizarea expresiilor, pentru căile de tranziţie $L_1 \div L_{11}$ sunt necesare 11 porţi AND din circuitul PLA faţă de 16 porţi AND când pentru sinteză s-a utilizat minimizarea expresiilor pe diagrame V-K. Dar, atât ieşirile de tip Mealy (y_1, y_{2_L} şi y_4) cât şi cele de tip Moore (y_3 şi y_5) fiind imediate suferă de inconvenientul acestor automate cu ieşiri imediate adică, sunt afectate de hazard. Hazardul se produce când cel puţin două variabile de intrare în CLC2 sau în CLC3 se modifică. De exemplu, pe calea de tranziţie L_{10}, când starea comută de la 011 la 000, dacă prin celula de registru de stare z_1 comutaţia de la 1 la 0 se propagă mai repede decât comutaţia de la 1 la 0 prin celula z_0 atunci pe ieşirea y_3, prin termenul $\overline{z_1} z_0$, se generează un glitch 1. Soluţia pentru eliminarea hazardului pe ieşiri este cea generală - ieşirile să fie convertite din ieşiri imediate în ieşiri întârziate prin introducerea de registre pe ieşire.

Prezentarea până acum a automatelor a fost lipsită de explicaţiile necesare realizării registrelor de stare şi de ieşire precum şi a celulelor acestora. Acestea vor fi

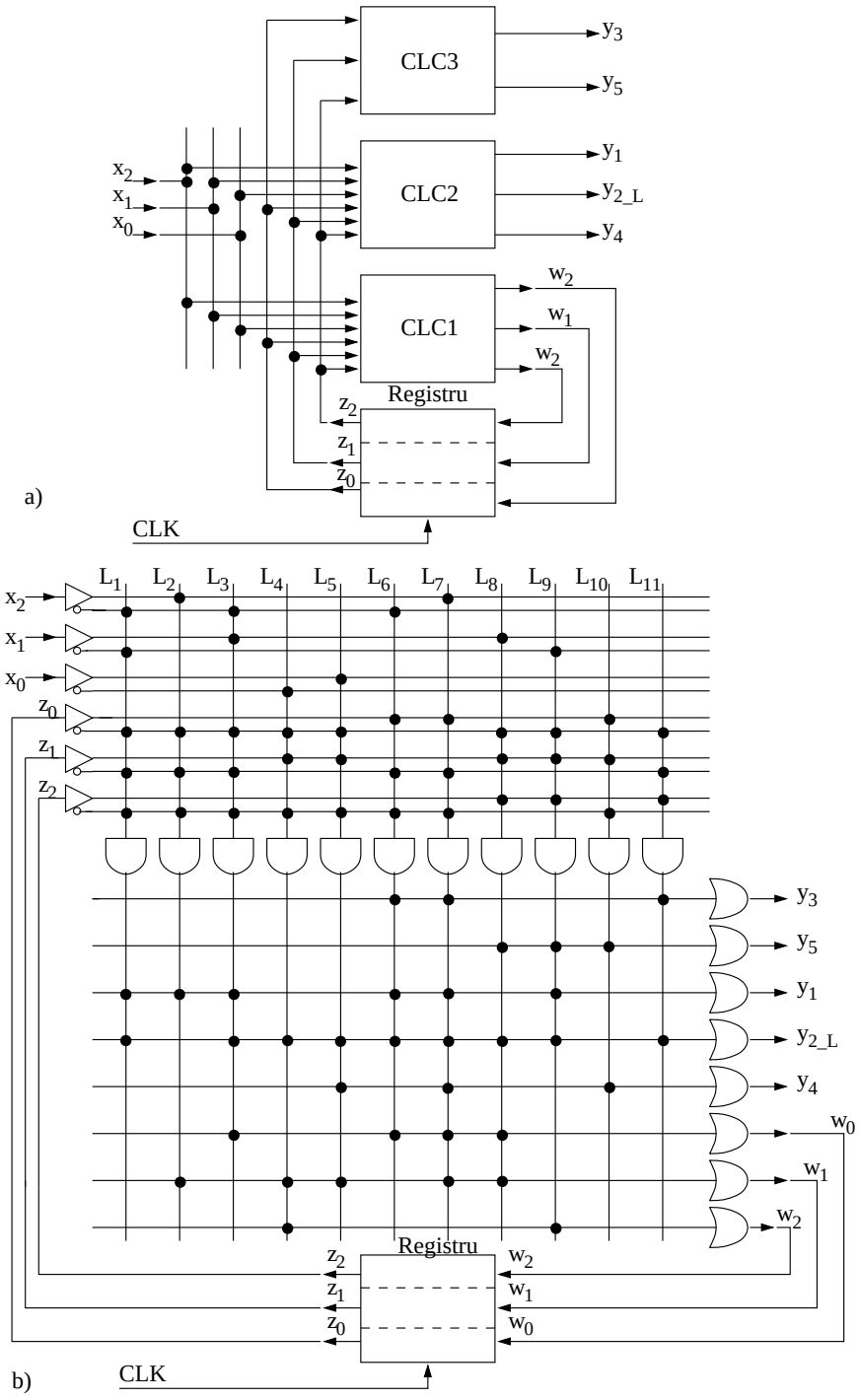


Figura 3.35 Explicativă pentru Exemplul 3.12 a) structurarea pe subcircuite a automatului; b) implementarea(fără minimizare) a automatului

clarificate în cadrul secţiunilor următoare de bistabile şi de registre.

Şi în capitolul de CLS, similar ca în capitolul de CLC, s-a pornit cu prezentarea noţiunilor fundamentale de organizare şi funcţionare a circuitului logic secvenţial, s-au expus modalităţi de exprimare pentru funcţia de tranziţie g şi de transfer f, trecând apoi la descrierea unei metode prin care se poate face sinteza unui circuit logic secvenţial (automat)). Evident, cu aceste cunoştinţe şi instrumente se poate proiecta un CLS. Dar, aşa cum s-a afirmat, dacă sistemul este de complexitate ridicată acesta se va partaja în automate de complexitate mai redusă. În general, se caută să rezulte prin partajare automate componente de largă utilitate care sunt nişte circuite, de facto, standard. Aceste circuite de facto standard există deja implementate şi obtenabile comercial, sarcina proiectantului reducându-se la a selecta circuitul şi a-i folosi optim toate facilităţile în sistem mai complex în care îl introduce. În continuare, în subcapitolele care urmează, se vor prezenta unele din aceste circuite secvenţiale cu aplicabilitate foarte extinsă în sistemele digitale.

3.3 CIRCUITE BASCULANTE

Structurarea cea mai simplă pentru un circuit secvenţial asincron din Figura 3.1 se obţine când partea combinaţională se reduce doar la un singur inversor. Un astfel de circuit, care a fost descris în Figura 3.3-a din Exemplul 3.2, nu prezintă nici o stare stabilă, deci este un oscilator. Acest circuit va prezenta două stări stabile dacă partea combinaţională va fi compusă din două inversoare (sau un număr par de inversoare) înseriate, Figura 3.4-b, din acelaşi exemplu. Dar, acest circuit cu două inversoare poate fi privit ca un circuit cu un singur inversor a cărui reacţie se realizează prin celălalt inversor, deci fiecare inversor are o reacţie - ieşirea este adusă la intrare - prin celălalt inversor. Circuitul astfel rezultat, cu două reacţii - câte una pentru fiecare inversor, constituie celula pe care se bazează întreaga clasă de cicuite basculante. Se pot diferenţia aceste circuite basculante în funcţie de modul de realizare, în curent continuu sau în curent alternativ, a celor două conexiuni de reacţie. Conform acestei diferenţieri rezultă următoarele tipuri de cicuite basculante:

1. **Circuitele basculante bistabile**, care reduse la celula de bază, prezintă pentru cele două conexiuni de reacţie legături galvanice simetrice. Acestea sunt **circuitele latch** şi **circuitele basculante bistabile (triggere)** care realizează suportul fizic pentru funcţia de memorare a unui bit. Există şi o variantă în care cele două legături galvanice de reacţie nu mai sunt simetrice, această variantă cunoscută sub denumirea de **trigger Schmitt** şi a cărei utilizare este de circuit discrimitator de nivel(releu de amplitudine).

2. **Circuitul basculant monostabil**, care prezintă o conexiune de reacţie în curent continuu iar cealaltă în curent alternativ (conexiune prin condensator). Această neidentitate a celor două conexiuni de reacţie determină ca cicuitul să aibă o stare stabilă **(monostabil)** iar cealaltă stare instabilă. Utilitatea monostabilului este aplicaţii de generare de intervale de timp(releu de timp).

3. **Circuitul basculant astabil**, care prezintă ambele conexiuni de reacţie în curent alternativ (conexiuni prin condensator). Cele două conexiuni fiind active

doar la variaţii îi imprimă cicuitului o funcţionare de oscilator, deci nici o stare stabilă **(astabil)**.

Prezenţa cuvântului basculant în denumirea acestor circuite semnifică faptul că trecerea între cele două stări se face prin **basculare**. Comutarea ieşirii la un CLC se poate produce la modificarea valorii unei variabile de intrare, iar ieşirea calculată este menţinută la nivelul logic respectiv doar atât timp cât este aplicată valoarea variabilei de intrare. Bascularea este tot un proces de comutare a ieşirii, iniţiată de o modificare a valorii unei variabile de intrare, dar ieşirea este menţinută la nivelul logic respectiv chiar dacă după un interval de timp nu mai rămâne aplicată valoarea variabilei de intrare. Rezultă că bascularea este o comutaţie care trebuie doar iniţiată, după care se autocomandă, acest efect de automenţinere este o consecinţă a reacţiei (pozitive) din structura circuitului. Un efect similar apare la comanda dispozitivului tiristor, unde este suficientă amorsarea pentru ca apoi printr-o reacţie pozitivă, în interiorul dispozitivului, să se menţină starea de conducţie. Deşi din punct de vedere fenomenologic există deosebire între comutaţie şi basculare în exprimarea curentă la circuitele basculante este foarte uzual să se substituie exprimarea corectă – circuitul basculează – cu exprimarea uzuală – circuitul comută.

3.3.1 Circuitul latch

Circuitul secvenţial asincron din Figura 3.36-a, identic cu cel din Figura 3.4-b de la Exemplul 3.2, unde s-a demonstrat că prezintă două stări stabile, poate fi redesenat sub o formă mult mai uzuală, pentru analiză, ca în Figura 3.36-b. În această formă se scot în evidenţă conexiunile de reacţie realizate încrucişat, de la ieşirea inversorului 1 la intrarea inversorului 2, $V_{in2} = V_{01}$, şi de la ieşirea inversorului 2 la intrarea inversorului 1, $V_{in1} = V_{02}$. O implementare a acestei organizări poate fi cu inversoare CMOS ca în Figura 3.36-c (unde prin linie întreruptă s-au identificat cele două inversoare).

Considerând caracteristica statică de transfer $V_0 = f(V_{in})$ a unui inversor, cu reprezentarea din Figura 1.14-b, prin conectarea încrucişată a acestuia cu un al doilea inversor identic, pentru semnalul de intrare V_{in1}, se obţine $V_{in1} = V_{02} = f(V_{in2}) = f(V_{01}) = f(f(V_{in1}))$. Aceasta înseamnă că semnalul de intrare V_{in1} după o inversare (180^0) pe primul inversor, $V_{01} = f(V_{in1})$, şi după încă o inversare (180^0) pe al doilea inversor, $V_{02} = f(V_{01})$, se aplică în fază peste V_{in1}, întărindu-l. La fel se poate deduce $V_{in2} = f(f(V_{in2}))$.

Deoarece între ieşirile celor două inversoare există un decalaj de 180^0 a semnalelor evident că ieşirea unui inversor este în nivelul H şi se notează prin Q, iar ieşirea celuilalt inversor va fi în nivelul L şi se notează prin $\overline{Q}$, Q_N sau Q_L. La conectarea tensiunii de alimentare, printr-un fenomen de concurs, deoarece practic cele două inversoare nu pot avea timpi de propagare identici, una din ieşiri va fi forţată în H iar cealaltă în L. Punctul de funcţionare al circuitului se obţine la intersecţia caracteristicilor statice de transfer ale celor două inversoare. Cele două caracteristici statice sunt trasate în Figura 3.36-d în care pentru inversorul 1 pe abscisă este fixată V_{in1} iar pe ordonată V_{01}, iar pentru inversorul 2, deoarece $V_{in2} = V_{01}$ şi $V_{02} = V_{in1}$, tensiunea de ieşire V_{02} este fixată pe abscisă şi V_{in2} pe ordonată. Se observă că se obţin trei puncte de funcţionare, două puncte stabile S_1, S_2 şi un punct instabil M.

Punctele S_1 şi S_2, în unul din acestea ajungând prin concurs la conectarea tensiunii, sunt puncte stabile pentru că în fiecare din acestea amplificarea totală $A = A_1 \cdot A_2$

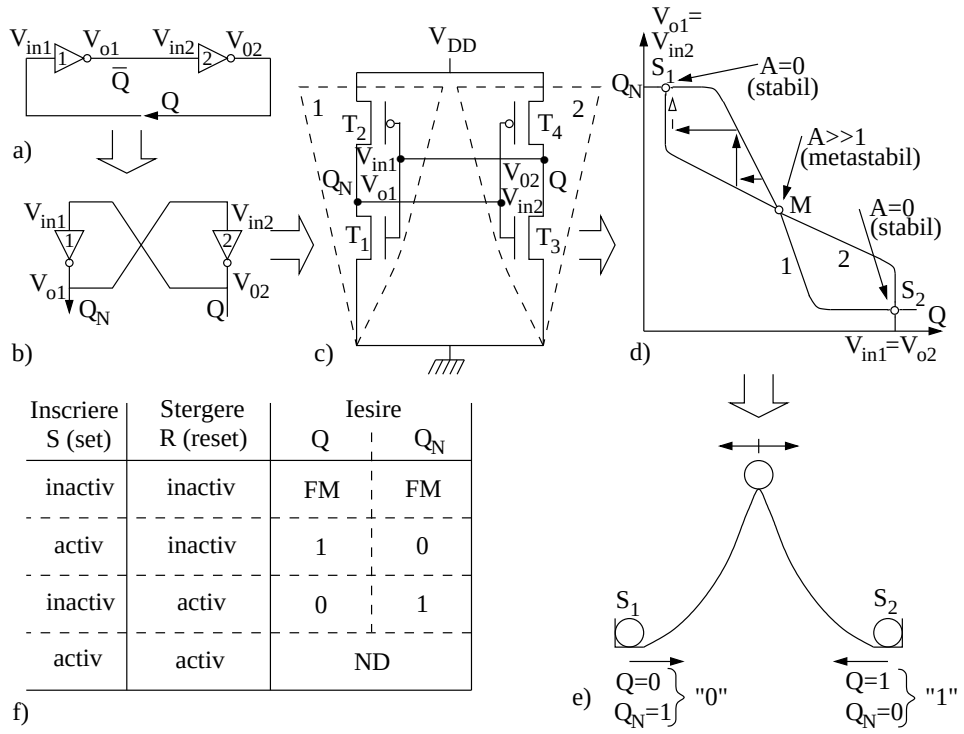


Figura 3.36 Celula elementară pentru circuitele bistabile: a,b) circuitul dublu inversor cu conexiuni de reacţie încrucişate; c) implementare cu inversoare CMOS; d) determinarea punctelor de funcţionare prin intersecţia caracteristicilor statice(S_1, S_2-stabile, M-instabil); e) analogie mecanică pentru punctele de funcţionare stabile; f) tabelul caracteristic pentru comanda celulei elementare.

pe bucla formată din cele două amplificatoare inversoare este nulă deoarece $A = dV_{01}/dV_{in1} = 0$, $A_2 = dV_{01}/dV_{in2} = 0$. Unul din puncte, prin convenţie, poate fi considerat că reprezintă starea "1" a circuitului $Q = 1, Q_N = 0$ iar celălalt starea "0" a circuitului $Q = 0, Q_N = 1$.

Al treilea punct de intersecţie M, care apare în zona liniară a caracteristicilor statice, zonă interzisă pentru funcţionarea în digital, vezi Figura 1.14-a, este denumit **punct de metastabilitate** şi are valorile pentru coordonatele V_{in} şi V_0 situate în afara nivelurilor logice H şi L, deci este fără utilitate în aplicaţiile digitale. Teoretic, circuitul poate sta un timp indefinit în punctul M dar, practic, după un anumit timp, determinat de variaţii de tensiune întâmplătoare, iese din M deplasându-se spre S_1 sau S_2. În zona punctului M coeficientul de amplificare total pe buclă are o valoare mare deoarece atât $A_1 \gg 1$ cât şi $A_2 \gg 1$; de exemplu, pentru circuitul din Figura 3.36-c toate patru tranzistoare sunt în regim de conducţie determinând o pantă foarte abruptă pentru caracteristica statică a fiecărui amplificator inversor. Dacă circuitul se află în punctul de metastabilitate şi întâmplător apare un mic zgomot care micşorează valoarea lui V_{in1}, această variaţie ΔV_{in1} propagată succesiv în bucla încrucişată prin

$\Delta V_{01}, \Delta V_{in2}$ şi ΔV_{02}, pentru că bucla are o amplificare mare, ajunge să producă o variaţie importantă asupra lui V_{in1} în sensul în care a produs-o zgomotul iniţial. Apoi, din nou, variaţia propagată succesiv prin buclă va duce la o şi mai pronunţată scădere a lui V_{in}, şi tot aşa repetat în buclă până când V_{in1} scade la nivelul L iar punctul de funcţionare ajunge în S_1. Această deplasare din M spre S_1 este figurată prin traseul în zig-zag între caracteristica inversorului 1 şi a inversorului 2.

O analogie mecanică pentru cele trei puncte de funcţionare ale cicuitului este reprezentată în Figura 3.36-e. Bila (punctul de funcţionare al cicuitului!) poate staţiona pe un plan orizontal în poziţiile S_1 sau S_2(puncte stabile) între care există un "deal" pe al cărui vârf bila se poate sprijini doar într-un singur punct (punctul de metastabilitate). Când bila se află în vârf foarte uşor oricare "adiere de vânt va sufla" bila fie înspre S_1 sau S_2. Când bila se află în S_1 sau S_2 cu o forţă (semnal) puternică se poate trece peste deal în poziţia opusă S_2 sau S_1. Critică apare situaţia (de nedeterminare, ND) când bila se află în S_1 sau S_2 iar o forţă de valoare mai redusă aplicată asupra ei poate să o deplaseze înspre vârful dealului, de unde revine în poziţia iniţială sau reuşeşte să o deplaseze chiar până în vârf unde staţionează şi de unde după un timp nedefinit poate reveni în poziţia stabilă iniţială sau poate trece în cealaltă poziţe stabilă.

Starea de metastabilitate poate apare la aplicarea tensiunii de alimentare sau la o comandă de trecere între cele două poziţii stabile, de la 1 la 0 sau de la 0 la 1 logic, dar semnalul de comandă este insuficient (nu are amplitudinea necesară sau are amplitudinea necesară dar durata de aplicare este prea scurtă). Starea de metastabilitate (punctul de funcţionare în M) trebuie evitată în circuitele digitale deoarece generează niveluri de tensiune în afara celor de 1 logic sau 0 logic şi, în plus, din starea de metastabilitate nu se poate determina în care stare următoare trece circuitul în cea de 0 sau în cea de 1 logic. Circuitul format din cele două inversoare cu conexiuni încrucişate, care este celula de bază pentru circuitele basculante, în structura prezentată până acum, nu poate fi comandat (se fixează într-o stare la conectarea tensiunii). Pentru a fi comandat trebuie să i se adauge intrări de comanda.

Intrările de comandă sunt: 1 - intrarea de înscriere, notată cu **S (Set)**, care prin activare va înscrie ieşirea în **starea logică 1,** $Q = 1$**,** $Q_N = 0$, 2 - intrarea de ştergere, notată cu **R (Reset)**, care prin activare va înscrie ieşirea în **starea logică 0,** $Q = 0, Q_N = 1$. Evident, cu cele două intrări se pot realiza patru comenzi de intrare, Figura 3.36-f:

1. Nu se activează nici una din intrări, **S=inactiv**, **R=inactiv** ceea ce corespunde unei comenzi de tipul nici o operaţie **NOP** (**N**o **OP**eration) adică ieşirea circuitului rămâne fără modificări, **FM**;

2. Se activează doar înscrierea **S=activ**, **R=inactiv**, ieşirea se înscrie în starea logică 1, $Q = 1, Q_N = 0$;

3. Se activează doar ştergerea **S=inactiv**, **R=activ**, ieşirea se înscrie în starea logica 0, $Q = 0, Q_N = 1$;

4. Se activează simultan atât înscrierea S=activ, cât şi ştergerea R=activ. Aceasta este o comandă absurdă! deoarece i se cere circuitului să treacă simultan atât în starea 1 ($Q = 1, Q_N = 0$) cât şi în starea 0 ($Q = 0, Q_N = 1$), **starea circuitului**

este nedeterminată , **ND**. Această comandă de înscriere şi ştergere simultană este evitată în circuitele digitale.

Conform clasificării circuitelor pe baza buclelor de reacţie incluse una în alta, prezentate în secţiunea 3.3, circuitul latch care conţine o singură buclă de reacţie este un circuit de ordinul 1, SO-1.

3.3.1.1 Latch-ul SR

La circuitul latch din Figura 3.36-b se pot adăuga cele două intrări de date S (înscriere) şi R (ştergere) dacă inversoarele sunt realizate cu porţi NOR, Figura 3.37-a, sau cu porţi NAND, Figura 3.38-a. Pentru structura obţinută, latch SR, la activarea (se consideră convenţia de logică pozitivă) unei intrări S sau R se obţine ieşirea in stare H dar nu la poarta NOR la care s-a aplicat intrarea activată ci la cealaltă poartă NOR (ieşirea unei porţi NOR este totdeauna L atunci când una sau mai multe intrări sunt H). Dacă intrarea de înscriere devine activă, $S = 1$, şi intrarea de ştergere este menţinută inactivă, $R = 0$, după timpul de propagare τ_p ieşirea porţii NOR_1 comută în starea L, $Q_N = 0$, care aplicată prin legătura de reacţie pe intrarea porţii NOR_2 va comuta, după timpul de propagare τ_p, ieşirea porţii NOR_2 în starea H, $Q = 1$. După un interval de timp $2{\cdot}\tau_p$, în sprijinul semnalului de înscriere $S = 1$ aplicat pe una din intrările porţii NOR_1, vine pe cealaltă intrare a potii NOR_1 un semnal de reacţie $Q = 1$ de la ieşirea porţii NOR_2, deci semnalul de înscriere poate fi dezactivat, $S = 0$, deoarece starea logică 1 ($Q = 1$, $Q_N = 0$) este menţinută în continuare de către semnalul de reacţie $Q = 1$. Dacă semnalul de înscriere $S = 1$ nu respectă această condiţie critică de timp, ca durata sa să fie $T_W \geq 2 \cdot \tau_p$, se poate ca latch-ul să nu se înscrie în starea logică 1 sau, mai rău, să intre în starea de metastabilitate unde, teoretic, poate sta indefinit, dar practic după un interval de timp se îndreaptă fie spre starea logică 1 ($Q = 1$, $Q_N = 0$), fie spre starea logică 0 ($Q = 0$, $Q_N = 1$). Intrarea în metastabilitate se poate genera şi în cazul în care semnalul aplicat pe intrare nu are o amplitudine suficientă, deci se situează în intervalul interzis între 1 si 0 logic. Dacă ulterior se activează din nou intrarea de înscriere, $S = 1$, odată sau de mai multe ori consecutiv, latch-ul rămâne înscris în aceeaşi stare logică 1, deci numai primul impuls dintr-o succesiune de înscriere are efectul de înscriere. Acest mod de funcţionare este similar cu cel de operare al unui **zăvor** (latch in limba engleză), care se închide la prima împingere a sa. Pentru deschiderea (ştergerea) zăvorului este necesară o împingere în sens contrar dar tot numai odată. Similar, se pot descrie secvenţele de propagare prin porţile NOR la înscrierea latch-ului în starea logică 0 când pe intrările de date se aplică $S = 0$, $R = 1$.

Tabelul caracteristic al latch-ului SR, Figura 3.37-b, care exprimă ieşirea în funcţie de cele patru cuvinte aplicate pe intrările de date, se obţine din tabelul din Figura 3.36-f adaptat pentru o convenţie de logică pozitivă ($activ = H$, $inactiv = L$). Din acest tabel caracteristic se poate deduce tabelul de excitaţie, Figura 3.37-c. **Tabelul de excitaţie**, care este în fond tabelul de evoluţie al stărilor, vezi Figura 3.2-b, prescrie pentru oricare comutaţie impusă latch-ului între două stări care trebuie să fie cuvântul SR aplicat pe **intrările de date**. Liniile tabelului de excitaţie se deduc din analiza liniilor din tabelul caracteristic în felul următor:

1. Comutaţia din $Q = 0$ în starea următoare $Q^+ = 0$ se realizează pentru $SR = 00$ din linia întâia a tabelului caracteristic (fără modificare, FM) sau pentru

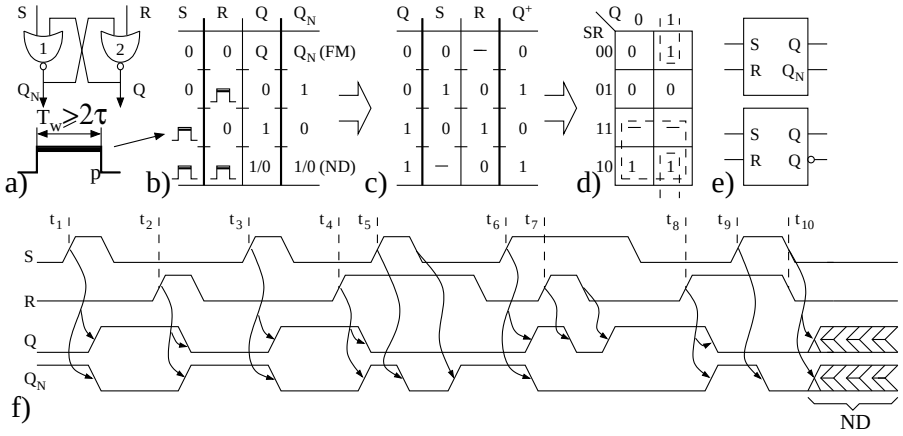


Figura 3.37 Analiza latch-ului SR: a) structură de latch realizat cu porţi NOR; b,c,d) tabelul caracteristic, tabelul de excitaţie şi diagrama VK pentru latch-ul SR; e) simboluri de reprezentare pentru latch-ul SR; f) explicarea funcţionării cu ajutorul diagramelor de timp.

$SR = 01$ din linia a treia (din starea logică 0 se aplică comanda de înscriere tot în starea logică 0), deci cuvântul necesar aplicat pe intrările de date trebuie să fie $SR = 0-$.

2. Pentru comutaţia din $Q = 0$ în $Q^+ = 1$ corespunde numai linia a treia din tabelul caracteristic, deci $SR = 10$.

3. Pentru comutaţia din $Q = 1$ în $Q^+ = 0$ corespunde numai linia a doua din tabelul caracteristic, deci $SR = 10$.

4. Comutaţia din $Q = 1$ în starea următoare $Q^+ = 1$ se realizează pentru $SR = 00$ (fără modificare, FM) sau pentru $SR = 10$ din linia a doua (din starea logică 1), deci cuvântul aplicat pe intrările de date trebuie să fie $SR = -0$.

Tabelul de excitaţie se poate transpune într-o diagramă de tipul V-K, Figura 3.37-d. Din această diagramă se poate deduce ecuaţia logică a latch-ului SR, care are expresia

$$Q^+ = S + \overline{R}Q \tag{3.20}$$

Această ecuaţie se poate deduce direct şi din structura latch-ului SR din Figura 3.37-a scriind relaţia pentru poarta NOR_1 în felul următor $Q_N^+ = \overline{S + \overline{R + Q_N}}$ care devine $Q^+ = S + \overline{R}Q$ deoarece $Q_N = \overline{Q}$.

Funcţionarea unui latch poate fi descrisă şi prin diagrame de semnale ca în Figura 3.37-f; săgeţile leagă fiecare modificare pe intrările de date cu efectele (comutaţiile) pe care le produce pe ieşiri. De exemplu, pentru modificarea intrărilor de date din momentele t_1, t_2, t_3, t_4 latch-ul se înscrie respectiv în stările 1, 0, 1 şi 0. La momentul t_5, atât timp cât $SR = 11$, ieşirile vor fi $Q_N = 0$ şi $Q = 0$, la fel se intâmplă în

momentul t_7 şi în momentul t_9. Diferenţa între t_5, t_7 şi t_9 apare prin faptul că la primele două intrările S şi R nu sunt dezactivate simultan, deci latch-ul trece într-o stare logică, pe când la t_9 intrările SR din 11 sunt dezactivate simultan t_{10} în 00 ceea ce provoacă intrarea în starea de metastabilitate, iar din această stare după un interval de timp latch-ul poate ajunge fie în 0 fie în 1. Cuvântul de intrare de date $SR = 11$ determină ca ambele ieşiri să fie în 0 şi se spune că această comandă produce o stare de nedeterminare (ND, linia a patra în tabelul caracteristic), pentru că la dezactivarea simultană a intrărilor de date (când ieşirile Q, Q_N au valori între 0 şi 1 logic)şi după consumarea intervalului de metastabilitate, se va trece fie în starea $Q = 0$, $Q_N = 1$, fie în starea $Q = 1$,$Q_N = 0$ fără a exista un control asupra acestei tranziţii.

Exemplul 3.13 Folosind circuitul latch să se formeze semnalul de la un comutator pentru a fi aplicat la intrarea unui sistem digital.

Soluţie. La închiderea unui contact, lamela acestuia, datorită elasticităţii, poate produce câteva oscilaţii înainte de a se opri pe poziţia comandată. Astfel că semnalul cules care să sesizeze poziţia închis a contactului, înainte de a ajunge la valoarea stabilizată, este sub forma unor impulsuri între o valoare V a tensiunii şi o valoare zero. În Figura 3.38-c tensiunea culeasă V_O înainte de a ajunge la valoarea zero, corespunzătoare contactului închis pe poziţia 2, are oscilaţii ca cele reprezentate în diagrama de semnale. Un sistem digital la care se aplică un astfel de semnal ar sesiza închiderea contactului de mai multe ori. Se poate obţine un semnal doar cu o singură comutaţie (la închiderea pe poziţia 2) dacă se utilizează un latch, deoarece acesta are proprietatea: este activat doar de prima activare dintr-un şir aplicat pe o intrare.

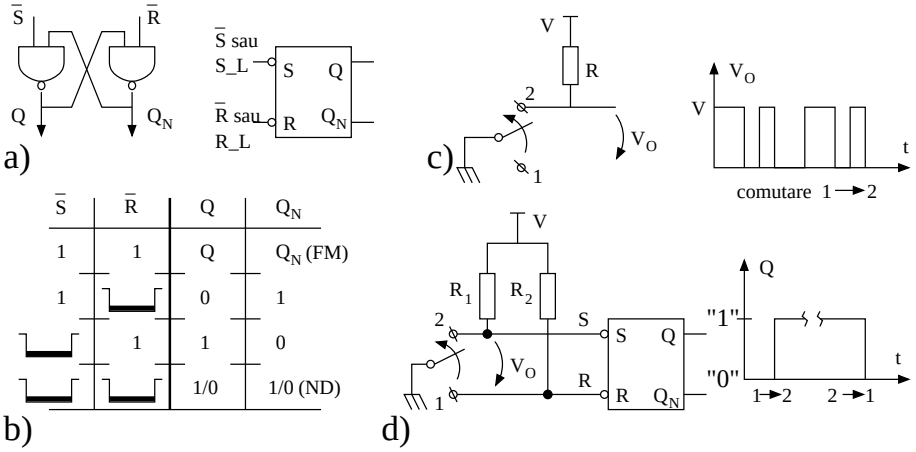


Figura 3.38 Explicativă pentru Exemplul 3.13: a,b) structură şi tabelul caracteristic pentru un latch cu intrările active în L (latch $\overline{S}\,\overline{R}$); c) oscilaţiile în semnalul obţinut la închiderea unui contact (datorită vibraţiilor lamelei contactului); d) circuit, cu un latch $\overline{S}\,\overline{R}$, pentru "curăţirea" unui semnal obţinut la închiderea/deschiderea unui contact.

O structură de latch cu porţi NAND este cea din Figura 3.38-a care se activează pe intrările de date prin semnale în stare L, de unde denumirea de latch $\overline{S}\,\overline{R}$. Tabelul carac-

teristic al latch-ului $\overline{S}\,\overline{R}$, Figura 3.38-b, se obţine din tabelul caracteristic din Figura 3.36-f aplicând convenţia de logică negativă (activ=L , inactiv=H).

Aplicând semnalul cules V_O, de pe comutator, la intrarea unui latch $\overline{S}\,\overline{R}$, ca în Figura 3.38-d, se obţine un semnal "curat" ca cel din diagrama de semnale alăturată. La închidere, din poziţia 1 în poziţia 2, latch-ul fiind comandat pe înscriere $\overline{S} = 0$, ieşirea sa comută în $Q = 1$, $Q_N = 0$, iar la deschidere din poziţia 1 în 2, latch-ul este comandat pe ştergere $\overline{R} = 0$, ieşirea comută in $Q = 0$, $Q_N = 1$.

În sistemele digitale, pentru o procesare controlată, este necesar să existe un control asupra felului/modului procesării (**cum?**), momentul procesării(**când?**) şi în care punct (adresă) să fie procesată (**unde?**). Aceste trei acţiuni, exprimate sintetic prin tripla interogaţie unde? când? şi cum?, apar ca o primă deficienţă a latch-ului $\overline{S}\,\overline{R}$ sau SR deoarece nu le poate realiza. În plus latch-ul mai prezintă starea de nedeterminare (ND) când sunt activate simultan cele două intrări de date.

Latch-ul cu ceas. La circuitul latch cuvântul aplicat pe intrările de date determină care va fi starea în care comută (cum?), dar acelaşi cuvânt aplicat determină şi momentul când comută (când?). Pentru a separa cele două acţiuni este necesar ca latch-ul să fie înzestrat cu intrare distinctă de cum? faţă de când?. Pentru modul de comutare se păstrează intrările de date $\overline{S}, \overline{R}$ sau S,R, iar pentru prescrierea momentului comutaţie se introduce o intrare pe care se aplică un semnal de ceas. Accesul cuvântului de date la intrările de date ale latch-ului se va face prin porţi care sunt validate de semnalul de ceas, deci modificarea stării circuitului va fi posibilă doar în momentele marcate de ceas (pe palier sau pe front). Ecuaţia logică a latch-ului SR cu ceas, similară cu relaţia 3.15, are expresia:

$$Q(t+1) = S + \overline{R}Q(t) \tag{3.21}$$

În relaţia 3.20 starea următoare a fost notată prin Q^+, dar în relaţia 3.21 a fost substituită prin Q(t+1) ca să indice faptul că starea următoare se înscrie, pornind de la starea prezentă Q(t), doar la validarea următoare $(t+1)$ prin semnalul de ceas.

În Figura 3.39 este prezentat un latch SR cu ceas ce se obţine din latch-ul din Figura 3.37-a căruia i s-au aplicat cele două porţi AND pe intrare, care validează accesul datelor pe intrările S şi R doar pe durata T_W a palierului pozitiv al ceasului. La fel, structura latch-ului cu ceas în tehnologie CMOS, Figura 3.39-b, se obţine din cea cu două inversoare din Figura 3.36-b (delimitată prin linie întreruptă în Figura 3.39-b), la care s-a adăugat în reţeaua de tip n un tranzistor pentru intrarea S înseriat cu un tranzistor pentru validarea de ceas, CLK, precum şi tranzistoarele complementare din reţeaua de tip p. Tabelul caracteristic şi tabelul de excitaţie, Figura 3.39-c şi d, pot fi considerate aceleaşi ca cele din Figura 3.37-c şi d; diferenţă existând în tabelul caracteristic, care prin linia a cincea exprimă faptul că dacă semnalul CLK nu este activ, nu se primeşte nici o comandă pe intrările de date, deci latch-ul este nevalidat.

Decuplarea între când şi cum se realizează numai în cazul când cuvântul de date este aplicat pe intrările de date şi numai după ce aceste date sunt stabilizate se aplică un semnal de ceas care validează transferul prin cele două porţi adăugate la intrare. Pe durata T_W a palierului activ de ceas oricare modificare pe intrările de date S,R este resimţită în ieşirile Q şi Q_N ale latch-ului ceea ce se referă prin: **latch-ul este transparent pe durata palierului activ al ceasului**. Deci, realizarea decuplării

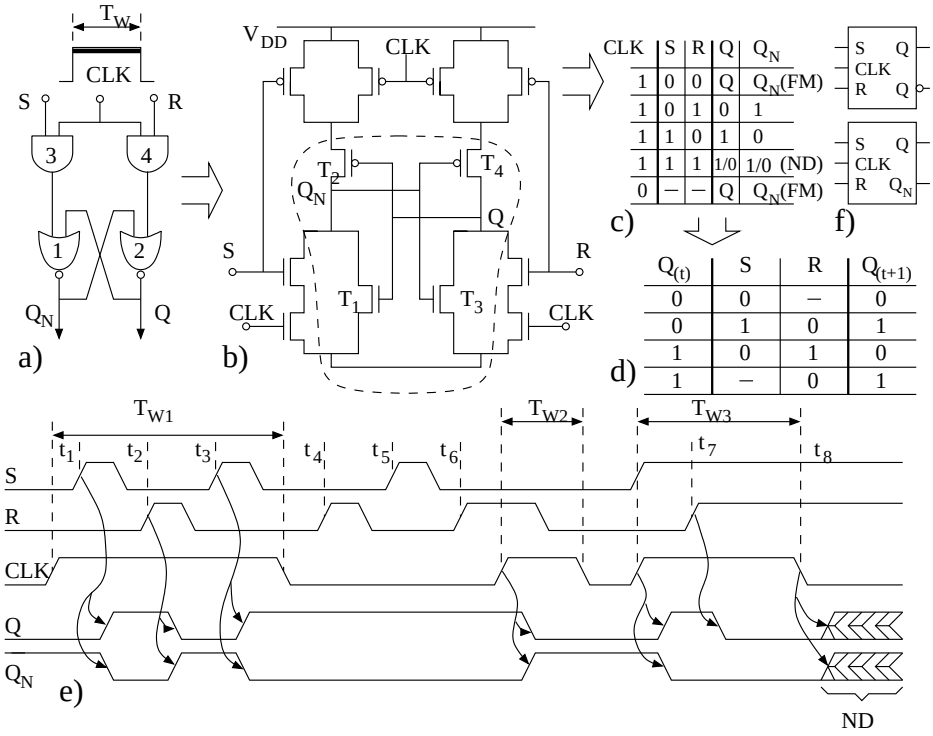


Figura 3.39 Latch-ul SR cu ceas: a) organizarea unui latch SR cu ceas şi implementarea sa (b) în tehnologie CMOS; c,d) tabelul de tranziţie şi tabelul de excitaţie; e) exemplu de diagrame de semnal; f) simboluri de reprezentare.

presupune că este îndeplinită condiţia prin care cuvântul de date este aplicat şi stabilizat pe intrările de date în momentul apariţiei palierului activ de ceas şi aceste date rămân nemodificate pe toată durata palierului.

Funcţionarea latch-ului cu ceas poate fi prezentată intuitiv prin diagrama de semnale ca în Figura 3.39-e. Pe durata palierelor active ale ceasului $T_{W_1}, T_{W_2}, T_{W_3}$ latch-ul este transparent, modificările de pe intrările de date se propagă spre ieşire, ieşirea este cuplată cu intrarea, iar pe durata de neactivare a ceasului modificările pe intrare (în momentele t_4, t_5, t_6) nu produc nici o schimbare pe ieşire. În momentul t_7 ambele intrări sunt activate $S = R = 1$ care comută ieşirile in $Q_N = 0$, $Q = 0$, deoarece ceasul este activ, dar în momentul t_8, când ceasul devine inactiv, latch-ul intră în metastabilitate, iar după un timp (cât?) se indreaptă fie spre starea logică 1 fie spre starea logică 0.

Deci, latch-ul cu ceas (latch-ul sincron) poate separa acţiunea "când" de "cum", dacă datele pe intrare rămân stabile pe o durată T_{W_1} dar nu elimină interdicţia de a nu fi activate simultan semnalele S şi R. Obtenabil ca circuit integrat discret este circuitul 74LS279 care conţine 4 latch-uri $\overline{S}\,\overline{R}$ pe capsulă.

3.3.1.2 Latch-ul D

În tabelul caracteristic al latch-ului SR sau $\overline{S}\,\overline{R}$, Figura 3.39-c, dacă se elimină linia întâia şi a patra se obţine tabelul caracteristic din Figura 3.40-c care exprimă funcţionarea latch-ului de tip D. Fizic, eliminarea celor două cuvinte de intrare, ambele intrări neactivate şi ambele intrări activate, se poate realiza prin conectarea celor două intrări prin intermediul unui inversor ca în Figura 3.40-a, adică totdeauna $R = \overline{S}$. Latch-ul rezultat, referit latch-ul D, având o singură intrare de date, notată cu D, nu mai prezintă restricţia, de la latch-ul SR, de a nu se activa simultan cele două intrări de date. Ca structurare latch-ul D se poate obţine din circuitul din Figura 3.39-a prin conectarea unui inversor pe intrarea R.

Variante de latch D foarte des utilizate în tehnologia unipolară sunt cele din Figura 3.40-b bazate pe circuite dinamice (Figura 1.54) şi porţi de transmisie. Bucla între inversoarele 1 şi 2 se închide prin poarta de transmisie PT2, iar introducerea (înscrierea) intrării D în buclă se face prin poarta de transmisie PT1; cele două porţi de transmisie sunt comandate în opoziţie prin semnalul de ceas. Pe palierul activ de ceas, CLK=1, PT1 este dechisă, PT2 este blocată iar capacitatea (parazită) de pe intrarea amplificatorului inversor 1 se încarcă ajungând la valoarea tensiunii existentă pe intrarea D. Pe durata palierului inactiv de ceas, CLK=0, PT1 este blocată, PT2 este deschisă realizându-se închiderea buclei între inversoarele 1 şi 2 ceea ce permite încontinuu încărcarea/menţinerea capacităţii parazite la valoarea de tensiune la care a fost încărcată.

Tabelul de excitaţie din Figura 3.40-d, dedus din tabelul caracteristic, arată că totdeauna ieşirea Q devine egală cu intrarea D la aplicarea frontului activ de ceas, deci ecuaţia care exprimă funcţionarea este

$$Q(t+1) = D \tag{3.22}$$

Conform acestei relaţii rezultă că latch-ul D poate stoca/memora data aplicată pe intrare, de unde şi denumirea de latch D (de la Data). Dar pe durata palierului activ de ceas latch-ul D are o funcţionare transparentă, oricare modificare pe intrare se propagă la ieşire, nu există o decuplare a ieşirii de intrare. Această transparenţă a latch-ului D se poate urmări pe diagrama de semnale din Figura 3.40-e pe intervalele T_{W_1}, T_{W_2} şi T_{W_3}, când frontul activ de ceas validează cele două porţi AND de pe intrare, în schimb modificările intrării D în momentele t_1, t_4, t_5 şi t_8 nu produc nici o comutaţie a ieşirii deoarece sunt în palierul inactiv de ceas.

Într-un sistem, funcţionarea corectă a unui latch D impune respectarea unor parametrii de timp. Timpii de propagare, daţi în cataloagele producătorilor, sunt definiţi pe diagramele de timp din Figura 3.40-f [Wakerly ′00]; abreviaţiile DQ indică propagarea de la intrarea de date la ieşire, iar CQ propagarea de la intrarea de ceas la ieşire. De exemplu, pentru circuitul integrat 74LS75, 4× latch D/capsulă, se dau următoarele valori: $\tau_{\text{PLH(DQ)}} = 11ns$ (tipic), $19ns$ (maxim); $\tau_{\text{PHL(DQ)}}$=$9ns$ (tipic), $17ns$ (maxim); $\tau_{\text{PLH(CQ)}}$=$10ns$ (tipic), $18ns$ (maxim); $\tau_{\text{PHL(CQ)}}$=$10ns$ (tipic), $18ns$ (maxim).

Modificarea semnalului de intrare D, pe durata palierului activ de ceas, produce doar o comutaţie corespunzătoare a ieşirii, pe când o modificare pe intrarea D care coincide cu durata fronturilor semnalului de ceas introduce latch-ul în metastabilitate, de exemplu la sfârşitul intervalului de transparenţă T_{W_3}. Pentru a evita metastabilitatea se impune ca intrarea de date să fie menţinută stabilă pe intervalul τ_{SU} înainte

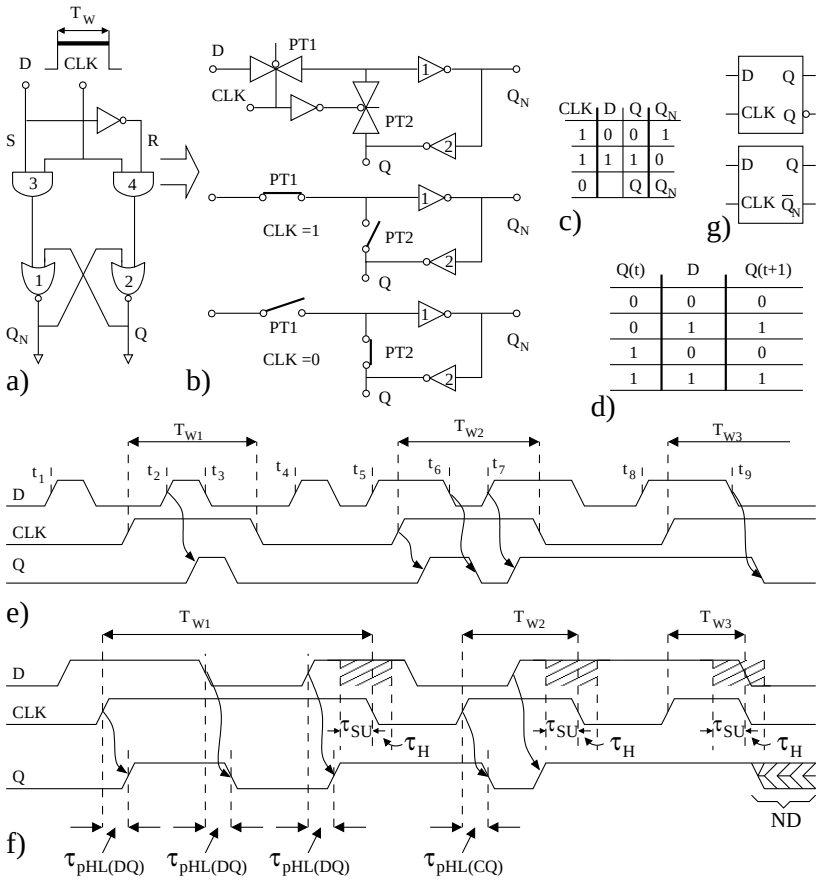


Figura 3.40 Latch-ul D: a) organizarea unui latch D şi implementarea sa (b) în tehnologie CMOS cu celule dinamice şi porţi de transmisie; c,d) tabelul de tranziţie şi tabelul de excitaţie; e) exemplu de diagrame de semnal; f) definirea timpilor de propagare; g) simboluri de reprezentare.

de frontul semnalului de ceas.

Exemplul 3.14 **Latch-ul Early**. Acest tip de latch, pe lângă funcţia existentă la latch-ul D, de a memora bitul aplicat pe intrarea D, poate realiza şi o a doua funcţie, cea de procesare pe două niveluri logice (AND-OR). În consecinţă, timpul de propagare necesar realizării celor două funcţii este egal numai cu timpul de propagare pe două niveluri logice. Structura latch-ului Early este cea desenată în Figura 3.41-a. Informaţia de un bit aplicată pe intrarea D, pe palierul CLK=1, se obţine pe ieşirea Q după ce s-a propagat prin porţile 3 şi 1. Apoi, prin legătura de la Q la poarta 4 se realizează reacţia, adică întărirea

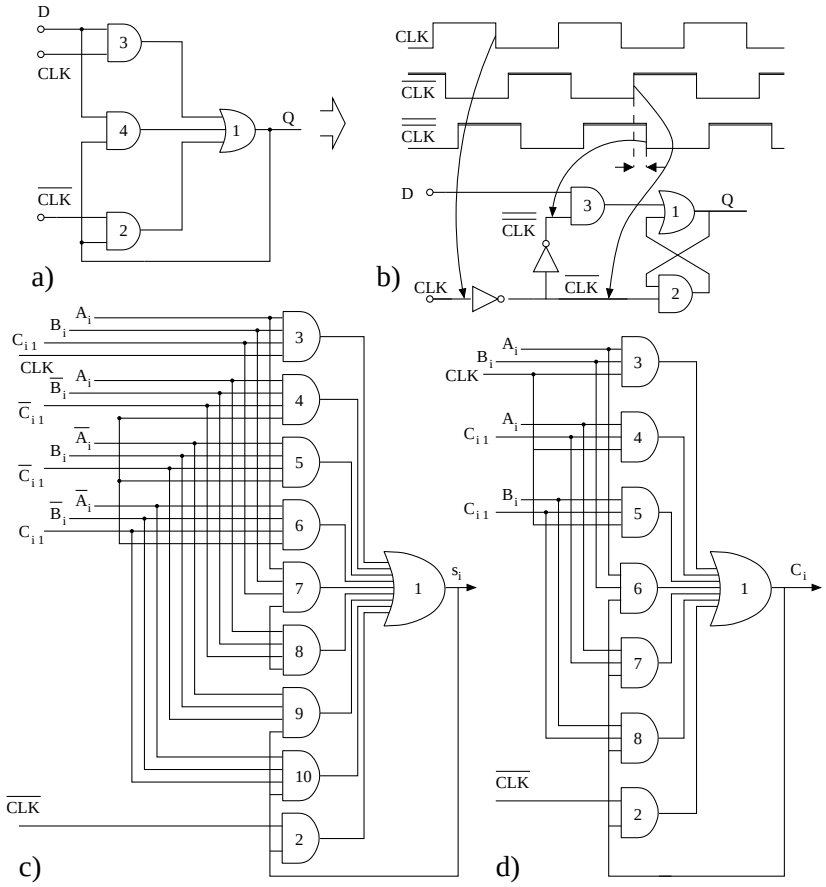


Figura 3.41 Latch-ul Early: a) structură de principiu; b) structură la care s-a eliminat poarta de menţinere pentru evitarea hazardului (prin comanda cu semnale de ceas defazate); c,d) calculul şi memorarea funcţiilor (s_i) şi (C_i) pentru un sumator cu transport progresiv pe o structură de tip latch Early.

comenzii aplicate pe intrarea de date, ieşirea Q fiind menţinută în continuare atât prin poarta 3 cât şi prin poarta 4. La dezactivarea semnalului CLK şi activarea palierului pentru semnalul $\overline{CLK}$=1 conexiunea de la ieşirea porţii 1 se închide prin poarta 2, deci menţinerea reacţiei se face în continuare prin polaritatea (inversă) a semnalului de ceas. Pentru a nu apare discontinuitate (hazard) în menţinerea valorii de ieşire Q la schimbarea semnalului de

ceas, adică la comutarea menţinerii de la poarta 3 la poarta 2, a fost introdusă poarta (de menţinere) 4.

Structuri de calcul şi memorare pentru expresiile 2.17 ale sumei s_i şi transferului următor C_i, la un sumator cu transportul anticipat, sunt prezentate în Figurile 3.41-c şi 3.41-d. Aceste structuri sunt extensii ale structurii de latch Early obţinute prin repartizarea a câte unei perechi de porţi, notate cu 3 şi 4 în Figura 3.41-a, pentru fiecare termen produs din relaţiile 2.17. Dacă implementarea acestor relaţii se face pe un circuit combinaţional pe două niveluri logice (AND-OR) urmat de un latch D (considerat că are propagarea $2\tau_p$ rezultă o propagare totală egală cu $4\tau_p$. Pentru implementarea cu latch Early propagarea totală fiind numai de $2\tau_p$; propagarea pe latch fiind "ascunsă" în propagarea părţii combinaţionale.

Avantajul implementării de tipul AND-OR plus memorare pe latch Early poate fi însoţită, cum este cazul sumatorului, şi de următoarele dezavantaje: semnalele de date şi de clock trebuie să aibă fan-out ridicat; poarta OR trebuie să suporte un fan-in mare, iar numărul de porţi AND poate fi de asemenea ridicat. Se pot diminua aceste dezavantaje dacă porţile de menţinere (7,8,9,10 în Figura 3.41-c şi 6,7,8 în Figura 3.41-d) sunt eliminate. Dar după această eliminare de porţi, pentru a asigura o funcţionare fără hazard a circuitului la comutaţia CLK→0 şi $\overline{CLK}$ →1, este necesar un defazaj între palierele active ale semnalului de ceas. Va trebui ca $\overline{CLK}$ →1 să devină activ cu Δt înainte ca CLK să fie dezactivat, ceea ce se poate obţine prin două buffere inversor pe intrări pentru semnalul de ceas, Figura 3.41-b.

Latch-ul D, cu restricţia impusă de a menţine nemodificată data de intrare pe durata palierului activ de ceas, poate separa acţiunea "când" de "când", de asemenea elimină posibilitatea activării simultane a două intrări dar, totuşi, până acum nu prezintă şi facilitatea de acţiune "unde". Evident, introducerea acţiunii "unde" presupune existenţa unui cuvânt (de adresă) care, pentru o configuraţie particulară, să selecteze (deci un decodificator) un anumit punct (locaţie) dintre mai multe unde există stocată informaţia (deci mai multe latch-uri D). O structură care realizează această funcţionare, referită prin latch adresabil, este reprezentată în Figura 3.42.

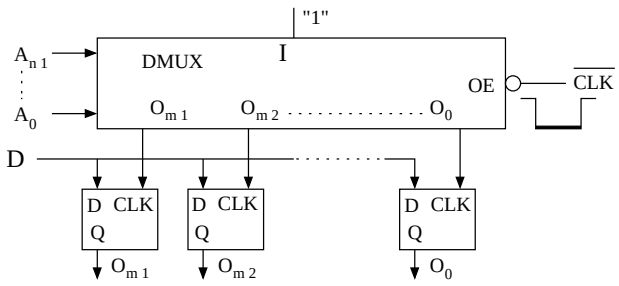


Figura 3.42 Organizarea de principiu pentru latch-ul adresabil.

Structura de latch adresabil se obţine printr-o conectare în paralel de m latch-uri D, având toate în comun aceeaşi intrare de date D. Selectarea pentru înscriere a datei, aplicată pe intrarea D, în unul din cele m latch-uri D se face prin activarea semnalului de ceas care se obţine ca o ieşire din cele $2^n (m = 2^n)$ ale DMUX 1:2^n. La DMUX pe intrarea de validare a ieşirii OE se aplică semnalul de ceas $\overline{CLK}$, iar pe

intrările de selectare se aplică un cuvânt de adresă $A_{n-1}A_{n-2}, ..., A_0$. Ieşirea fiecărui latch D, pentru citire, este disponibilă permanent fără selectare.

Înscrierea într-un anumit latch (adresă), pentru o funcţionare corectă, trebuie să respecte anumite relaţii temporale. Pentru latch-uri, se aplică întâi semnalul pe intrarea D, care trebuie să fie stabil pe palierul activ al semnalului aplicat pe o intrare CLK. Pentru DMUX întâi se aplică cuvântul de selectare şi numai după stabilizarea acestui, pentru a evita hazardul pe ieşiri, se activează semnalul de ceas $\overline{CLK}$ (=OE). Structura de latch adresabil la care se adaugă selectarea (unde) şi pe ieşire (pentru citire) se transformă în structura celulei de bază pentru implementarea unei locaţii de memorie.

3.3.2 Circuite Basculante Bistabile (Triggere)

Pentru eliminarea deficienţelor prezentate de structura de latch elementar (nediferenţierea între acţiunile "când" şi "cum", inexistenţa adresării "unde" şi restricţia activării simultane a comenzii de înscriere şi de citire) s-au realizat diferite tipuri de latch-uri cu ceas. Totuşi, aceste latch-uri cu ceas mai prezintă un inconvenient: transparenţa – ieşirea este cuplată cu intrarea (pe durata palierului activ de ceas). Circuitele care realizează o izolare a ieşirii de intrare sunt referite prin termenul de bistabile/triggere (flip-flop).

3.3.2.1 Principiul master-slave

Principiul Master-Slave (M-S) exprimă o modalitate de structurare a oricărui tip de circuit basculant. Această modalitate de structurare constă în înserierea a două latch-uri cu ceas, fiecare fiind transparent pe palierul H al semnalului de ceas. Considerăm un bistabil SR structurat M-S, ca în Figura 3.43-a, din două latch-uri SR, primul - master - comandă pe cel de al doilea - slave. Datele SR ale bistabilului se aplică pe intrările de date ale latch-ului master, ale cărui ieşiri Q_M, Q_M_L se aplică pe intrările SR ale latch-ului slave, iar ieşirile bistabilului Q, Q_N sunt cele ale latch-ului slave. Cele două latch-uri sunt validate, pentru transferul datelor de la intrare spre ieşire, pe paliere defazate cu 180 ale semnalului de ceas, latch-ul master este comandat pe palierul CLK=1 iar, apoi, latch-ul slave pe palierul $\overline{CLK} = 1$ (datorită inversorului pe semnalul de ceas). Pe durata palierului pozitiv, CLK=1, latch-ul master fiind transparent datele de intrare (sau orice modificare a acestora) sunt transferate la ieşire,Q_M, Q_M_L, dar latch-ul slave este blocat. În schimb, pe durata palierului negativ, $\overline{CLK} = 1$, latch-ul master este blocat iar latch-ul slave devine transparent şi transferă datele de la ieşirile masterului Q_M, Q_M_L, care sunt intrări la slave, spre ieşirile bistabilului, Q, Q_N. De fapt, latch-ul slave are ca date de intrare datele de la ieşirea latch-ului master în momentul comutării semnalului de ceas de la 1 la 0, deci bistabilul apare ca fiind comandat de către frontul posterior. Bistabilul are izolată ieşirea de intrare, deoarece permanent unul din latch-uri este blocat, iar bascularea apare ca fiind efectuată în momentul frontului negativ de ceas. Evident, ca bistabilul să-şi modifice ieşirea conform datelor aplicate pe intrările SR acestea trebuie să rămână stabile pe toată durata de transparenţă a latch-ului master, CLK=1, până în momentul apariţiei frontului posterior care va determina transferul în slave. Această întârziere, din momentul aplicării datelor până în momentul bas-

culării ieşirii, este specificată în simbolul de reprezentare al structurilor master-slave, Figura 3.43-d, prin simbolul ⌉ aplicat pe ieşirile Q şi Q_N.

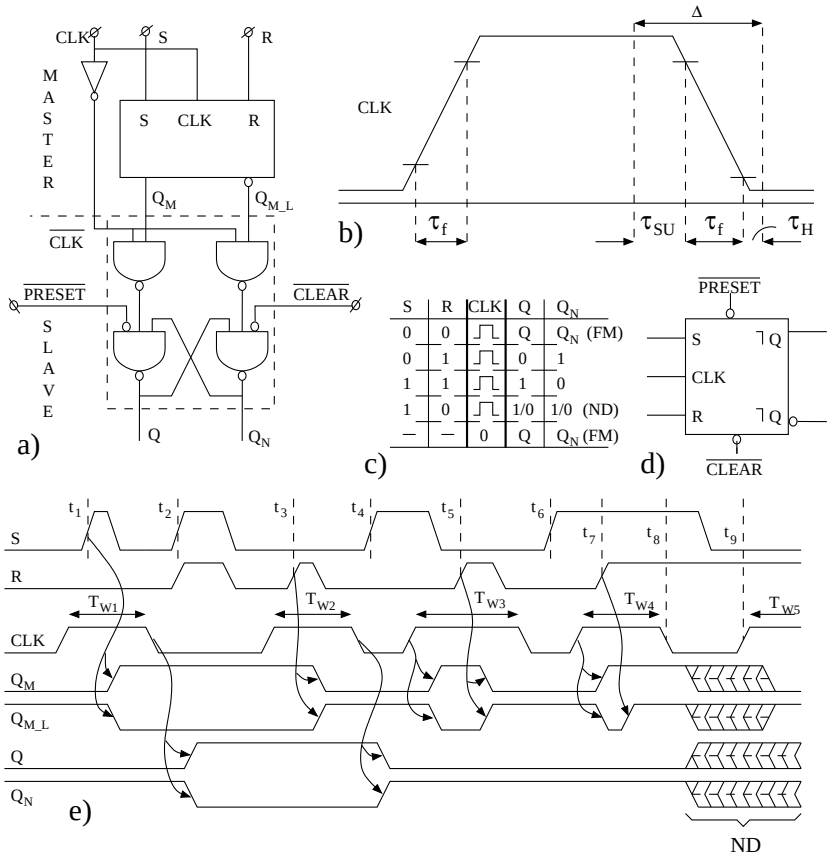


Figura 3.43 Principiul master slave: a) organizarea de tip master-slave; b) reprezentarea intervalelor de timp pentru o tranziţie corectă a bistabilelor comandate pe front (în cazul acesta pe frontul posterior); c,d) tabelul de tranziţie şi simbolul de reprezentare pentru un bistabil SR master-slave; e) diagrame de semnal ubţinute în funcţionarea unui bistabil SR master-slave.

Transparenţa latch-ului master pe durata palierului pozitiv de ceas este un dezavantaj deoarece un glitch, posibil să apară în sistemele digitale, poate modifica datele aplicate pe intrări, înscrie o valoare eronată în latch-ul master şi care, apoi, la apariţia frontului posterior se transferă în latch-ul slave spre ieşirea bistabilului.

Din punct de vedere logic bistabilul SR master-slave este identic cu latch-ul SR cu ceas, ceea ce rezultă şi din identitatea tabelelor caracteristice din Figurile 3.43-c şi 3.39-c, au acelaşi tabel de excitaţie, Figura 3.39-d, deci aceeaşi ecuaţie logică, relaţia 3.20. Funcţionarea bistabilului SR este descrisă şi în diagrama de semnale din Figura 3.43-e. În intervalele de transparenţă ale masterului T_{W_1}, T_{W_2}, T_{W_3}, T_{W_4} şi T_{W_5}, când ceasul este activ, apariţia unui impuls pe una sau ambele intrări de date

poate modifica starea latch-ului master, stare deja fixată de configuraţia de date (care exista la începutul intervalului respectiv de transparenţă). De exemplu, în momentul t_1 latch-ul master este comutat din 0 în 1, în t_3 din 1 în 0 şi în t_5 din 1 în 0 iar schimbările configuraţiei de date din momentele t_2, t_4 şi t_6 sunt ignorate de latch-ul master deoarece nu este activat de ceas. Se observă că modificarea stării latch-ului slave poate apare doar pe frontul posterior al semnalului de ceas. În momentul t_8, pe frontul posterior al semnalului de ceas T_{W_4}, deoarece $Q_M Q_M\_L = 11$ ($SR = 11$), ambele latch-uri intră în metastabilitate, iar din această stare poate ieşi după un anumit timp înscriindu-se fie în starea 1 fie în starea 0. Latch-ul master va fi înscris determinist, şi scos din starea de metastabilitate, dacă încă nu s-a consumat această stare, în momentul t_9 când este înscris prin SR=01 în starea 0, iar latch-ul slave poate fi înscris determinist în starea 0 pe frontul posterior al palierului T_{W_5}.

Rezultă, din explicaţiile anterioare, că starea de metastabilitate la un latch apare când ambele sale ieşiri sunt în 1 la dezactivarea semnalului de ceas. Dar, metastabilitate poate apare şi când intrările de date se modifică pe durata frontului de înscriere/(activare) a semnalului de ceas, Figura 3.43-b (se consideră frontul posterior când se înscriu datele în latch-ul slave). Ca durată a frontului de înscriere se consideră un interval Δ compus din următoarele trei subintervale:

- τ_{SU} **-timpul de prestabilire (Set-Up)**, care este intervalul de timp dinainte de începerea tranziţiei active (frontului) a ceasului în care intrările de date trebuie să fie stabile;
- τ_{f} **-durata frontului activ de ceas**;
- τ_{H} **-timpul de menţinere (Hold)**, care este intervalul de timp după tranziţia frontului activ în care intrările de date trebuie să mai fie încă menţinute stabile.

Deci intervalul Δ, denumit **fereastră de decizie**, axat pe frontul activ de tranziţie are durata

$$\Delta = \tau_{\mathrm{SU}} + \tau_{\mathrm{f}} + \tau_{\mathrm{H}} \tag{3.23}$$

Este referit prin fereastră de decizie deoarece în acest interval Δ bistabilul testează datele aplicate pe intrare şi decide care va fi starea la ieşire.

Pentru circuitele TTL valori uzuale sunt: $\tau_{\mathrm{SU}}=(5 \div 20)ns$, $\tau_{\mathrm{H}}=(5 \div 0)ns$; la circuitele mai rapide aceşti parametri tind spre valori care se situează înspre limitele inferioare ale intervalelor. La implementarea unui crcuit, într-o anumită tehnologie, se recomandă ca perioada T a semnalului de ceas să fie faţă de timpul de propagare τ_{p} pe o poartă în relaţia $T \geq (40 \div 50)\tau_{\mathrm{p}}$. Lăţimea ferestrei de decizie, în raport cu durata unui palier activ de ceas este foarte unică, deci se poate considera că tranziţia pe frontul semnalului de ceas a bistabilelor se realizează punctual (într-un moment de timp). La toate circuitele bistabile, a căror structură se bazează pe celule latch, pentru o funcţionare corectă trebuie respectată această condiţie de nemodificare a datelor în intervalul Δ, axat pe frontul activ al semnalului de ceas.

Circuitele basculante pentru înscriere, pe lângă intrările de date validate (sincronizate) cu semnalul de ceas, în general, mai prezintă încă două **intrări asincrone**, de regulă active în stare low, notate prin $\overline{PRESET}$ (înscriere în starea $Q = 1$, $Q_N = 0$) şi $\overline{CLEAR}$ (înscriere în starea $Q = 0$, $Q_N = 1$). Aceste intrări asincrone se aplică, Figura 3.43-a, la bistabil pe celula latch care generează starea bistabilului Q,

Q_N; având acces direct, fără validare prin semnal de ceas, la fixarea stării bistabilului, rezultă că **intrările asincrone realizează o comandă prioritară în raport cu intrările de date**.

Deşi organizarea de tip master-slave a bistabilelor realizează o izolare a ieşirii de intrare faptul că datele de intrare trebuie să fie menţinute constante pe durata palierului activ de ceas apare ca o restricţie destul de supărătoare în anumite aplicaţii. Soluţia ar fi: scurtarea acestui palier doar la un impuls care să apară fie pentru frontul pozitiv fie pentru frontul negativ al semnalului de ceas. Astfel se obţin **bistabile cu comutaţie pe front**. Generarea unui impuls pe unul din fronturile semnalului de ceas se bazează pe producerea intenţionată a hazardului de propagare, Figura 2.24.

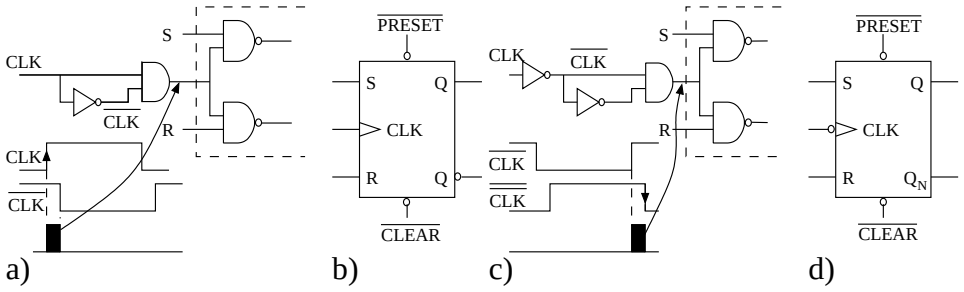


Figura 3.44 Bistabile cu comutaţia pe front: a,c) circuite pentru generarea de impulsuri pentru comutaţia bistabilelor pe frontul pozitiv, respectiv negativ; b,d) simbolurile de reprezentare ale bistabilelor SR cu comutaţie pe frontul pozitiv, respectiv negativ.

În Figurile 3.44-a şi 3.44-b sunt prezentate două circuite care produc impulsuri, cu lăţimea τ_{p} egală cu timpul de propagare printr-un inversor, respectiv pe frontul pozitiv sau negativ. Un astfel de impuls de ceas, care să comande comutaţia bistabilului, trebuie să aibă durata egală sau mai mare cu timpul de tranziţie în bistabil a semnalelor de date spre ieşire. Pentru semnalele de date restricţia de nemodificare se reduce doar la intervalul de τ_{SU} înainte de front şi τ_{H} după front, adică pe intervalul Δ, ceea ce este cu mult mai redus decât lăţimea unui palier a semnalului de ceas. Comanda unui bistabil pe frontul pozitiv se simbolizează prin semnul $\rhd$, ca în Figura 3.44-b iar pentru frontul negativ prin semnul $\circ\!\!\rhd$ ca în Figura 3.44-d. Cele mai utilizate bistabile, mai ales pentru implementări sub formă de circuite integrate discrete, sunt cele cu comutaţie pe front. Bistabilele cu comutaţie pe front rezolvă cu acurateţea unui front problema decuplării acţiunii "cum" de "când".

3.3.2.2 Bistabilul D

Acest tip de bistabil, similar din punct de vedere logic latch-ului D, se poate obţine dintr-un bistabil SR, căruia i se forţează datele de intrare să îndeplinească relaţia $\overline{R} = S$, prin utilizarea unui inversor pe intrări. În Figura 3.45-a este prezentată modalitatea de transformare a unui bistabil SR master-slave într-un bistabil de tip cu comutaţie pe frontul negativ (transferul din latch-ul master în latch-ul slave se

realizează la sfârşitul palierului activ de ceas CLK=1). Se poate realiza un bistabil D cu comutaţie pe frontul pozitiv dacă se utilizează încă un inversor pe intrare, obţinându-se deschiderea master-ului pe palierul negativ al ceasului iar al slave-ului pe palierul pozitiv. Evident că indiferent de tipul frontului semnalul aplicat pe intrarea D trebuie să respecte restricţiile de nemodificare pe durata intervalelor τ_{SU} şi τ_{H}, iar înscrierea se face conform datei eşantionate în momentul frontului.

O structură de bistabil D cu comutaţie pe frontul negativ, realizată prin înserierea a două celule CMOS dinamice, este reprezentată în Figura 3.45-b. Celula latch master este validată pe palierul pozitiv de ceas, iar celula latch slave este validată pe palierul negativ de ceas. Când CLK=1, poarta de transmisie PT1 este dechisă iar PT2 închisă, ieşirea Q_M urmăreşte intrarea D iar intrarea în celula slave nu este validată deoarece poarta de transmisie PT3 este închisă (doar PT4 deschisă pentru a menţine valoarea ieşirii Q înscrisă anterior).În momentul frontului negativ, când semnalul de ceas devine $\overline{CLK}$=1, poarta PT1 este închisă iar PT2 deschisă, celula master este devalidată , în schimb celula latch slave devine transparentă, poarta de transmisie PT3 se deschide iar PT4 se închide, şi înspre ieşirea Q se transmite valoarea logică Q_M memorată în momentul frontului negativ. Succesiunea aceasta de transfer se repetă în următoarea perioadă a semnalului de ceas; porţile PT1 şi PT4 sunt dechise simultan când CLK=1, iar în opoziţie, când $\overline{CLK} = 1$, sunt comandate simultan PT2 şi PT3.

Tabelul caracteristic al bistabilului D, Figura 3.45-c, este similar cu cel al latch-ului D, Figura 3.40-c cu deosebirea că la primul tranziţia este pe front iar la al doilea pe palier. Ca tabel de excitaţie pentru bistabilul D se poate utiliza cel de la latch-ul D, Figura 3.40-d la fel şi ecuaţia logică a bistabilului D este cea exprimată prin relaţia 3.22. Diferenţa în funcţionare între latch-ul D şi bistabilul D se poate evidenţia din analiza diagramelor de semnal din Figura 3.45-d. Se observă că pe intervalele $CLK = 1$ (T_{W_1}, T_{W_2}, T_{W_3}, T_{W_4}) latch-ul fiind transparent ieşirea Q urmăreşte variaţiile intrării D ceea ce nu se întâmpla cu ieşirea Q a bistabilului, acesta va memora numai valorile lui D din momentele fronturilor negative; deci bistabilul poate elimina transferul spre ieşirea sa a semnalelor de zgomot. Dacă perioada semnalului de ceas este cu mult mai mică decât cele mai mici intervale între variaţiile semnalului aplicat pe intrare atunci ieşirea Q a bistabilului poate fi identică cu intrarea D, dar cu o întârziere de propagare faţă de aplicarea frontului de tranziţie activ.

Un tip de bistabil D, foarte uzual în aplicaţii, este cel cu validarea (semnalului de ceas), prezentat în Figura 3.45-e. Se compune dintr-un bistabil D cu MUX2:1 pe intrarea D, iar selectarea multiplexorului se realizează cu semnalul de validare, E. La intrarea de date 0 a multiplexorului se aplică ieşirea Q a bistabilului D iar pe intrarea 1 se aplică data de intrare la bistabilul D cu validare. Pentru semnalul de validare inactiv, E=0, pe fiecare front pozitiv de ceas se reînscrie în bistabilul D valoarea Q (bistabilul rămâne în aceiaşi stare); bistabilul cu validare apare ca un bistabil D la care semnalul de ceas nu este validat (nu se aplică). Pentru semnalul de validare activ, E=1, data de intrare se înscrie, prin MUX2:1, în bistabilul D. În aplicaţii acest tip de bistabil înscrie condiţionat data în funcţie de un alt semnal care se aplică pe intrarea de validare. O altă variantă de bistabil D cu validare, dar cu facilităţi extinse, este prezentată în Figura 3.72-c.

Aplicaţiile bistabilului D sunt: memorarea unui bit aplicat pe intrare, conform relaţiei $Q(t + 1) = D$, la fel ca latch-ul D; sincronizarea semnalelor asincrone înainte de a fi aplicate unui sistem sincron ("aducerea în acelaşi timp" cu ceasul sistemului

sincron). Atât pentru memorarea unui bit cât şi pentru sincronizarea unui semnal asincron nu apar probleme dacă semnalul aplicat pe intrarea D nu are modificări în fereastra de decizie Δ. Dacă semnalul aplicat pe intrarea D este o dată deja sincronizată (nu se modifică în interiorul ferestrei de decizie), deci respectă timpii de prestabilire τ_{SU} şi de menţinere τ_H, atunci această dată este transferată la ieşire, cu înârzierea de propagare τ_{pD} faţă de frontul activ, şi va fi valoarea ieşirii bistabilului pe următoarea perioadă de ceas; în Figura 3.46-a este prezentat acest mod de funcţionare.

Probleme apar când semnalul aplicat pe intrarea D este un semnal asincron care îşi modifică valoarea în interiorul ferestrei de decizie (modificarea valorii unui semnal asincron poate avea loc oricând), Figura 3.46-b.

Eşantionarea unui semnal asincron D, care variază pe durata ferestri de decizie (se modifică în $\overline{D}$), poate crea două situaţii de evitat în funcţionarea bistabilului D:

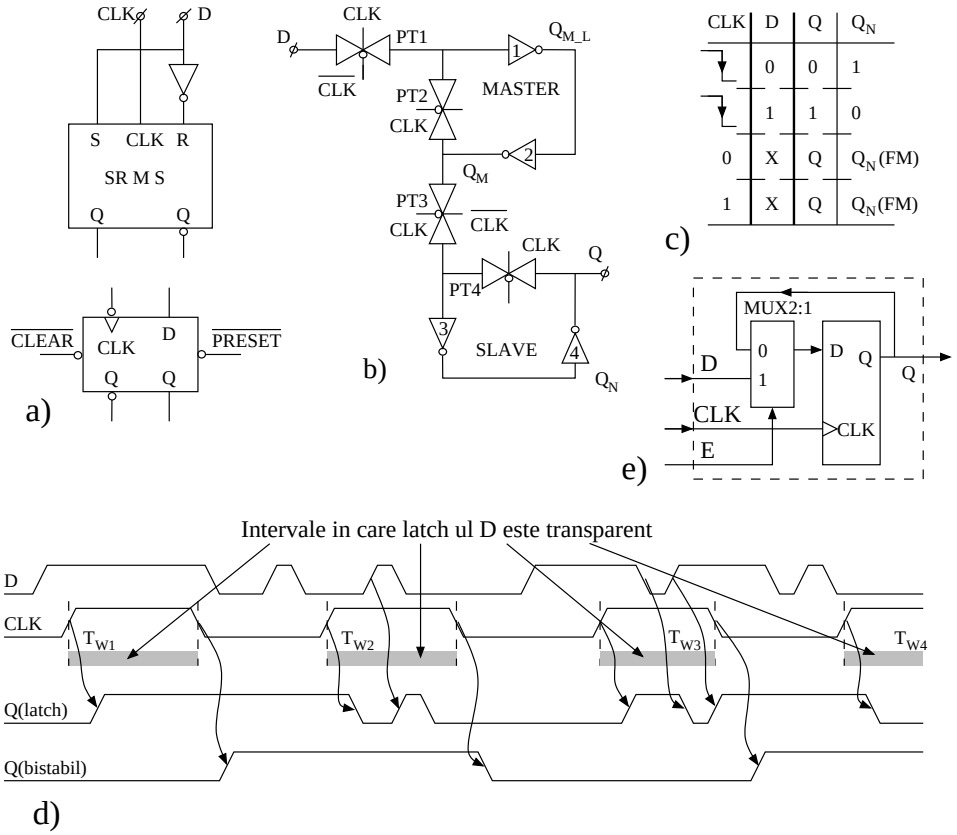


Figura 3.45 Bistabilul D: a) transformarea unui bistabil SR master-slave într-un bistabil D cu comutaţie pe frontul negativ; b) structură de bistabil D în tehnologie CMOS cu celule dinamice; c) tabelul caracteristic; d) comparaţia formelor de semnal obţinute la ieşirile unui latch D şi un bistabil D şi un bistabil D pentru acelaşi semnal aplicat pe intrare.

1. Ieşirea bistabilului, după timpul de propagare τ_{pD}, se înscrie nedeterminist, fie cu valoarea D aplicată pe intrare, fie cu valoarea complementară $\overline{D}$. Dacă se înscrie cu valoarea $\overline{D}$ sincronizarea a fost reuşită, iar dacă se înscrie cu valoarea D se poate considera că eşantionarea s-a realizat puţin înainte de modificare, când semnalul aplicat pe intrare într-adevăr avea valoarea D. La următoarea eşantionare (următorul front activ de clock), dacă semnalul asincron nu îşi mai modifică valoarea, în bistabil se înscrie valoarea corectă, adică $\overline{D}$; pentru sistemul care utilizează semnalul sincronizat apare că semnalul a fost eşantionat (sincronizat) cu întârzierea de o perioadă de ceas.

2. Bistabilul, după timpul de propagare τ_{pD}, oscilează sau intră în metastabilitate, la ieşire se generează un semnal între 0 şi 1 logic, deoarece funcţionarea (latch-ului de bază al bistabilului) este în regim liniar. Iar dacă ieşirea acestui bistabil se aplică pe intrarea altor bistabile/porţi unele din acestea pot sesiza intrarea pe nivel logic 1 iar altele pe nivel logic 0 sau unele bistabile pot intra în metastabilitate. După un timp, τ_{m} - timp de metastabilitate - care are o determinare probabilistică [Wakerly ′00], bistabilul iese din starea de metastabilitate înscriindu-se, nedeterminist, fie în valoarea logică 1, fie 0, Figura 3.46-b. Scoaterea bistabilului din metastabilitate se obţine prin două mobilităţi. Prima, se forţează într-o stare validă prin aplicarea unei intrări care respectă parametrii de τ_{SU} şi τ_{H}. A doua, după τ_{m} (care are o durată?!) starea de metastabilitate se consumă.

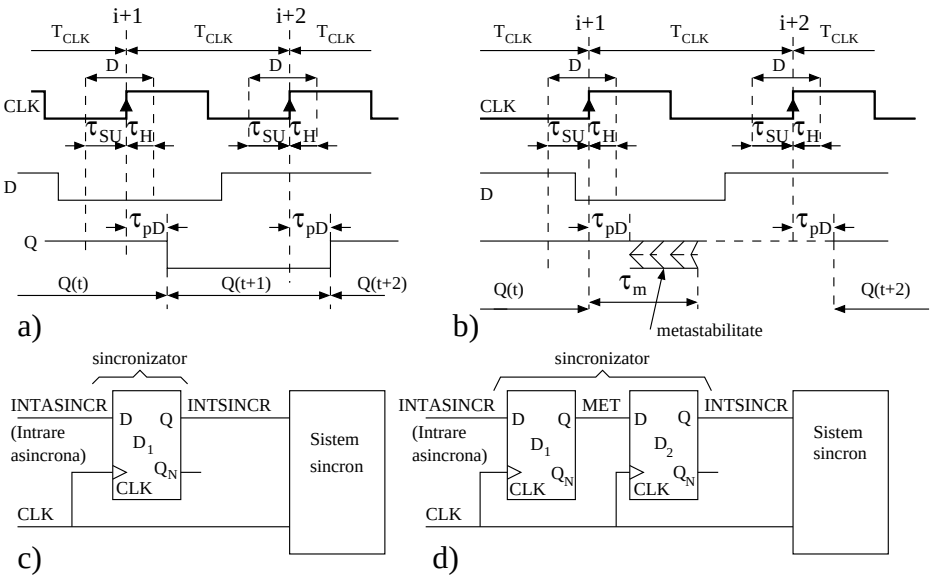


Figura 3.46 Utilizarea bistabilului D pentru sincronizarea datelor. Tranziţia datelor prin bistabil când este îndeplinită condiţia de nemodificare în fereastra de decizie (a) când nu este respectată condiţia de nemodificare; c) structură de sincronizator cu un bistabil D (d) cu două bistabile înseriate.

Un circuit pentru obţinerea unei intrări sincrone, INTSINCR, aplicate apoi la un sistem sincron, prin eşantionarea unui semnal asincron, INTASINCR, este prezentat în Figura 3.46-c. Elementul sincronizator este un bistabil D care produce o sincronizare corectă în afară de cazul în care semnalul INTASINCR îşi modifică valoarea în fereastra de decizie. Când INTASINCR se modifică în fereastra de decizie variaţia semnalului de ieşire obţinut, INTSINCR, poate fi încadrată în una din cele două situaţii de evitat, în funcţionare ale bistabilului D, descrise anterior (aplicarea unei intrări nedefinite, fie 1, fie 0, la sistemul sincron sau o sincronizare ratată – adică aplicarea unei valori situate între 0 şi 1 logic). Evitarea acestor funcţionări incorecte poate fi realizată de către varianta de sincronizator cu două bistabile D_1, D_2 înseriate ca în Figura 3.46-d. Semnalul INTSINCR va fi generat corect de bistabilul D_2 dacă în fereastra lui de decizie semnalul MET, produs de bistabilul D_1, nu are modificări. Chiar şi când semnalul MET, produs de D_1, este nedeterminist, fie 1, fie 0, acesta devine stabil după τ_{pD}, deci următorul front de ceas îl găseşte stabilizat pe intrarea bistabilului D_2. Dar dacă semnalul MET este produs de starea de metastabilitate a lui D_1, şi această stare nu s-a consumat până la apariţia noului front de ceas , $\tau_{\text{m}} > \tau_{\text{CLK}}$, atunci şi D_2 intră în metastabilitate. Relaţia de temporizare care să impună ca D_2 să nu intre în metastabilitate, deci ca această variantă de sincronizator să nu producă sincronizări ratate, are expresia $\tau_{\text{m}} \leq T_{CLK} - \tau_{\text{SU}}$. Din această relaţie se deduc două modalităţi de a evita sincronizări ratate: 1- mărirea perioadei T_{CLK} a semnalului de eşantionare (în general, semnalul asincron are o frecvenţă de modificare mult mai mică decât frecvenţa de ceas); 2- micşorarea timpului de prestabilire τ_{SU} (în general, circuitele moderne, mai rapide, realizează valori mai mici, chiar sub 5 ns).

În concluzie, există posibilitatea de a realiza o sincronizare aproape sigură pe timpul de funcţionare al unui echipament. Structuri de sincronizatoare care să realizeze pentru timpul mediu între două (eventuale) sincronizări ratate, **MTBF** (**M**ean **T**ime **B**etween **S**ynchronizer **F**ailures), valori ce depăşesc timpul de viaţă de funcţionare al unui echipament pot fi găsite în [Wakerly ′00].

3.3.2.3 Bistabilul JK

Toate circuitele prezentate până acum, prin completarea structurii latch-ului din secţiunea 3.3.1, au avut ca finalitate eliminarea unora din deficienţele latch-ului. Totuşi, a rămas nesoluţionată comanda contradictorie de activare simultană a înscrierii şi ştergerii. Aceasta comandă a fost eliminată, dar nu soluţionată, la latch-ul D şi bistabilul D prin comasarea acestei comenzi cu cea de neactivare simultană a intrărilor (No-OPeration), dar circuitul s-a redus la o singură intrare de date. Soluţionarea se poate obţine prin realizarea unui automat cu două intrări **J-înscriere** şi **K-ştergere**, Figura 3.47-a, având ca element de memorare, fie un latch cu ceas, fie un bistabil, care pe lângă comenzile normale (înscriere JK=10, ştergere JK=01, No-OPeration JK=00) să aibă şi **comanda JK=11 pentru care să basculeze în starea opusă**.

Funcţionarea acestui automat - bistabilul JK - este redată în tabelul caracteristic din Figura 3.47-b. Se obţine tabelul de excitaţie, Figura 3.47-c, sau forma concentrată, Figura 3.47-d, dacă în tabelul caracteristic se analizează pentru fiecare din cele patru tranziţii posibile, între cele două stări, care din linii asigură tranziţia respectivă. De exemplu, tranziţia între stările $Q(t) = 1$ şi $Q(t+1) = 0$ poate fi realizată de linia treia, JK=01, sau linia a patra, JK=11, adică JK=-1. Mapând acest tabel de excitaţie

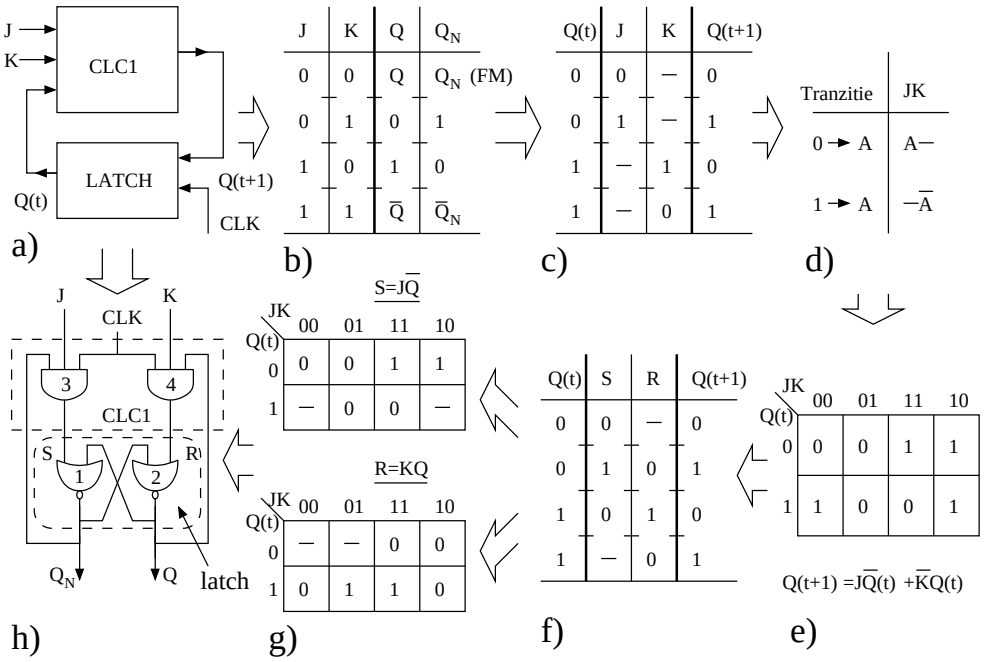


Figura 3.47 Etapele sintezei structurii de bistabil JK: Pornind de la structura de automat (a) cu tabelul caracteristic (b) şi tabelul de excitaţie (c,d) se obţine tabelul de tranziţie al stărilor sub forma unei diagrame V-K (e). Folosind tabelul de excitaţie al latch-ului SR(f) se obţin funcţiile de excitaţie $S = J\overline{Q}(t)$ şi $R = \overline{K}Q$ (g) rezultând structura de tip JK (h).

într-o diagramă V-K, având intrările $Q(t)$ şi JK, Figura 3.47-e, se deduce ecuaţia logică pentru funcţionarea bistabilului JK.

$$Q(t+1) = J \cdot \overline{Q}(t) + \overline{K} \cdot Q(t) \tag{3.24}$$

Alegând pentru elementul de memorare al automatului JK un latch SR cu ceas este necesar a se determina ecuaţia logică pentru CLC1 care generează semnalele S şi R. Diagramele V-K pentru sinteza funcţiilor de excitaţie S, R, Figura 3.47-g, se obţine prin conversia diagramei V-K a bistabilului JK utilizând tabelul de excitaţie al latch-ului SR, Figura 3.47-f. De exemplu, când bistabilul JK este în starea $Q(t) = 0$ şi se aplică JK=00 bistabilul comută în starea $Q(t+1) = 0$, căsuţa de coordonate $Q(t)JK = 000$. Dar pentru că această tranziţie (0→ 0) este realizată de către latch-ul SR se observă, din tabelul de excitaţie al acestuia, că valorile aplicate pe intrările sale trebuie să fie $S = 0$, $R = -$, valori care se introduc în cele două căsuţe de coordonate $Q(t)$ JK=000 ale diagramelor V-K din Figura 3.47-g. Iar când bistabilul comută din $Q(t) = 1$ în $Q(t+1) = 0$, căsuţa de coordonate $Q(t)$ JK=111, valorile aplicate pe intrările latch-ului trebuie să fie $S = 0$ şi $R = 1$, valori care se introduc respectiv în căsuţele de coordonate $Q(t)$ JK=111 ale diagramelor V-K pentru funcţiile S şi R. Rezultă cele două funcţii de excitaţie $S = J \cdot \overline{Q}(t)$ şi $R = \overline{K}Q(t)$ cu implementarea

din Figura 3.47-h.

Structura de bistabil JK obţinută, corectă din punct de vedere logic, nu este practic funcţională când palierul activ al semnalului de ceas are durata mai mare decât timpul de propagare prin latch. De exemplu, dacă bistabilul este în starea logică 0 ($Q = 0$, $Q_N = 1$) şi pe intrările de date se aplică $JK = 11$ poarta 3 va genera un semnal $S = 1$ care va înscrie bistabilul în starea 1 ($Q = 1$, $Q_N = 0$), iar dacă se menţine $JK = 11$ se va deschide acum poarta 4 ce va genera un semnal $R = 1$ care va înscrie bistabilul în starea 0 ş.a.m.d, dacă se menţine $JK = 11$. Perioada de oscilaţie este egală cu dublul timpului de propagare printr-o poartă AND de pe intrare plus timpul de propagare prin latch-ul SR. De fapt, când durata palierului activ de ceas este mai mare decât timpul de propagare în automat, acesta are o funcţionare de automat asincron (transparenţă), care pentru $JK = 11$ nu îndeplineşte niciodată condiţia de stabilitate, starea prezentă să fie identică cu starea viitoare calculată, deci automatul este cu oscilator.

Eliminarea posibilităţii de intrare în oscilaţie a bistabilului JK, din Figura 3.47-h, când JK=11, se poate realiza prin izolarea intrării de ieşire adică substituind latch-ul SR cu o structură de bistabil SR master-slave, Figura 3.48-a, obţinându-se **bistabilul JK master-slave** cu simbolul de reprezentare din Figura 3.48-c şi tabelul caracteristic din Figura 3.48-b.

Dar şi acest tip de bistabil este afectat de ceea ce se numeşte captarea de (zgomote) 1 sau 0 datorită transparenţei latch-ului master pe palierul activ de ceas. De exemplu, dacă bistabilul este în starea 1 ($Q = 1$, $Q_N = 0$) iar pe intrările de date se aplică JK=00 atunci în momentul apariţiei palierului activ de ceas latch-ul master se înscrie în $Q_M = 1$ şi $Q_M\_L = 0$ deci bistabilul va rămâne tot în starea 1. Dar dacă pe durata palierului activ de ceas pe intrarea K apare un glitch, posibil în sistemele digitale, acesta va înscrie ($K \cdot Q = 1$) latch-ul master în 0, $Q_M = 0$ şi $Q_M\_L = 1$, care se transferă apoi la ieşire ca stare 0 a bistabilului chiar dacă pe intrare a dispărut zgomotul şi se revine la JK=00 (captare zero). Similar se întâmplă dacă bistabilul este în starea 0 ($Q = 0$, $Q_N = 1$), intrarea este JK=00 şi pe intrarea J apare un glitch, va rezulta ($J \cdot Q_N = 1$) la ieşire starea 0 ($Q = 1$, $Q_N = 1$) chiar dacă zgomotul a dispărut şi se revine la JK=00 (captare 1).

Se obţine un bistabil JK, fără posibilitatea captării de 1 sau 0, dacă structura de bistabil master-slave este substituită cu un bistabil D cu comutaţie pe front, Figura 3.48-d al cărui simbol de reprezentare este desenat în Figura 3.48-f, iar tabelul caracteristic este descris în Figura 3.48-e. Un exemplu de diagrame de semnal pentru **bistabilul JK cu comutaţie pe frontul pozitiv** este dat în Figura 3.48-g; variaţia de semnale pe intrări trebuie să respecte condiţia de nemodificare pe intervalele τ_{SU} şi τ_{H} în jurul frontului de comutaţie.

Dintr-un bistabil JK se poate obţine simplu un bistabil D prin comasarea liniilor unu şi patru, Figura 3.48-e, în una singură prin utilizarea unui inversor pe intrare. Un bistabil JK, având două bucle de reacţie suprapuse, este un sistem de ordinul doi SO-2 pe când un bistabil D este un sistem de ordinul unu SO-1. Utilizarea unui bistabil JK pentru modelarea funcţionării unui bistabil D ar însemna irosirea autonomiei unui SO-2 faţă de un SO-1; realizarea unui bistabil D dintr-un bistabil JK este o soluţie numai când nu există un bistabil D.

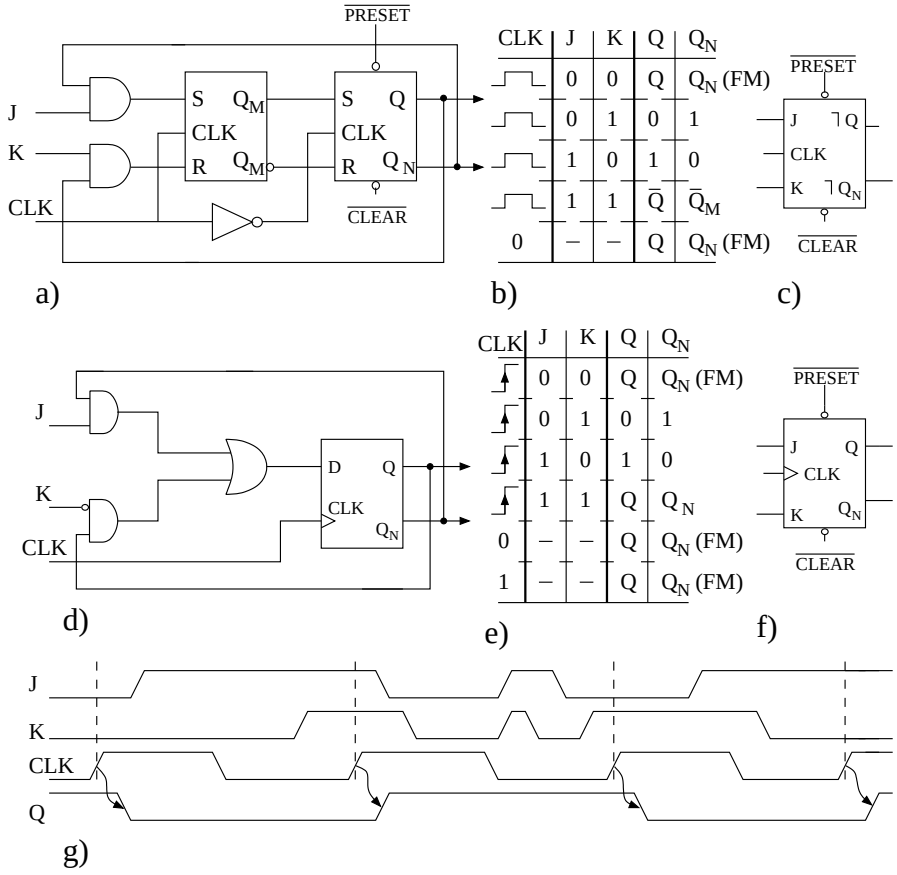


Figura 3.48 Bistabilul JK: Structura (a) tabelul caracteristic (b) şi simbolul de reprezentare (c) pentru un bistabil JK master-slave; structura (d), tabelul caracteristic (e) simbolul de reprezentare (f) şi diagrame de semnal (g) pentru un bistabil JK cu comutaţie pe front pozitiv.

3.3.2.4 Bistabilul T

Conectând împreună cele două intrări JK, nu printr-un inversor ca pentru modelarea funcţionării bistabilului D, ci direct se obţine bistabilul de tip T. În practică funcţionarea unui bistabil T totdeauna este modelată pe un bistabil JK, sau pe celelalte tipuri, deoarece nu există un circuit integrat discret care să conţină bistabile T. Prin conectarea împreună a intrărilor tabelul caracteristic din Figura 3.48-e devine tabelul caracteristic al bistabilului T reprezentat în Figura 3.49-b. Din acesta, notând J=K=T, se obţine tabelul de excitaţie, Figura 3.49-c, şi tabelul de tranziţie al stărilor reprezentat prin diagrama V-K din Figura 3.49-d din care rezultă ecuaţia de funcţionarea bistabilului T:

$$Q(t+1) = \overline{T} \cdot Q(t) + t \cdot \overline{Q}(t) = T \oplus Q(t) \tag{3.25}$$

iar pentru $T = 1$ aceasta devine:

$$Q(t+1) = \overline{Q}(t) \tag{3.26}$$

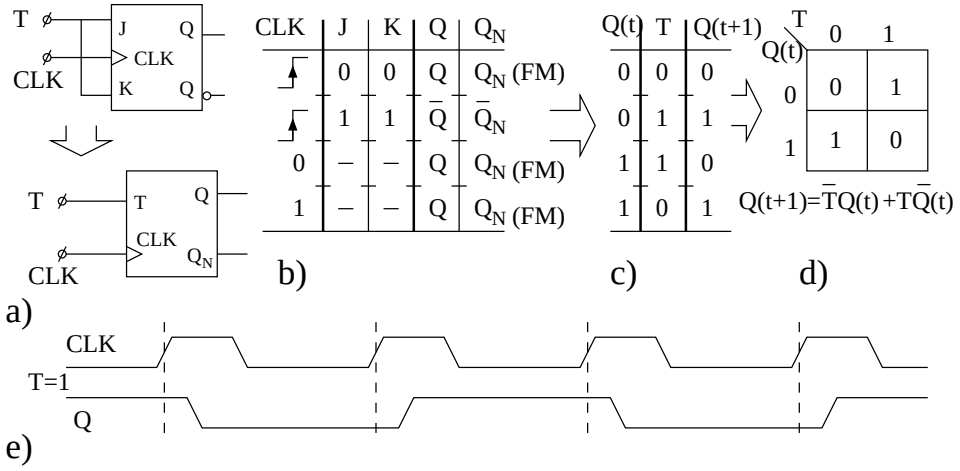


Figura 3.49 Bistabilul T: a) modalitatea de a obţine un bistabil T dintr-un bistabil JK (J=k); b) tabelul caracteristic; c,d) tabelul de excitaţie; e) diagrame de semnal (semnalul Q are perioada dublă faţă de semnalul de ceas).

Relaţia 3.26 arată că la menţinerea intrării la valoarea T=1, bistabilul T la fiecare impuls de ceas comută/(basculează) în starea opusă de unde şi denumirea de **basculă** (toggle). Această simplă comportare de basculă are două aplicaţii foarte frecvent utilizate în sistemele digitale:

1. **Numărător modulo 2**. Deoarece tot la al doilea front (impuls de ceas) aplicat pe intrare bistabilul revine în aceeaşi stare ($QQ_N \rightarrow \overline{Q}\,\overline{Q}_N \rightarrow QQ_N \rightarrow \overline{Q}\,\overline{Q}_N \ldots$) ceea ce este echivalent cu identificarea claselor de resturi modulo 2, $\hat{0}$ şi $\hat{1}$, din numărul de impulsuri aplicat pe intrare, deci un numărător în bază 2. Înseriind două bistabile T starea acestora, Q_1Q_0, revine în aceiaşi valoare după patru impulsuri de ceas, adică identifică clasele de resturi modulo 4: $00 = \hat{0}$, $01 = \hat{1}$, $10 = \hat{2}$, $11 = \hat{3}$ pentru că 4 modulo $4 = 0 = \hat{0}$. Se poate deduce că un numărător modulo 2^n se obţine din înserierea a n circuite bistabile T.

2. **Divizor de frecvenţă**. Pentru că la fiecare front în acelaşi sens al semnalului de ceas, adică la un interval de o perioadă T_{CLK}, bistabilul basculează alternativ de la $1 \rightarrow 0$, $0 \rightarrow 1$, $1 \rightarrow 0 \ldots$ rezultă că perioada semnalului Q este dublă în raport cu perioada semnalului de ceas $T_Q = 2T_{CLK}$. În plus, pentru că Q comută doar la fronturile în acelaşi sens ale ceasului nu depinde de factorul de umplere al semnalului de ceas. În Figura 3.49-e factorul de umplere al semnalului de ceas este sub 50% pe când frontul de umplere al semnalului generat pe Q este de 50% (există o mică abatere faţă de această valoare cu diferenţa între timpii de propagare τ_{pHL} şi τ_{pLH}). În concluzie, semnalul obţinut pe Q este cu un factor de umplere 50% şi are frecvenţa jumătate din cea a semnalului de ceas.

Tabelul 3.4 Descrierea sintetică a bistabilelor SR, D, JK şi T

Tipul de bistabil	Tabelul Caracteristic	Tabelul de excitare si ecuatia de functionare	Graful de tranzitie	Simboluri de reprezentare si exemple de circuite obtenabile comercial
SR	CLK S R Q Q_N ↑ 0 0 Q Q_N(FM) ↑ 0 1 0 1 ↑ 1 0 1 0 ↑ 1 1 1/0 1/0(ND) 0 – – Q Q_N(FM)	Q(t) S R Q(t+1) 0 0 – 0 0 1 0 1 1 0 1 0 1 – 0 1 Q(t+1)= S + R$\bar{Q}$(t)	Q=0 Q_N=1; 0 01; 10 Q=1 Q_N=0; 0	PRESET S Q CLK R Q_N CLEAR 74xx71
D	CLK D Q Q_N ↑ 0 0 1 ↑ 1 1 0 0 – Q Q_N(FM) 1 – Q Q_N(FM)	Q(t) D Q(t+1) 0 0 0 0 1 1 1 0 0 1 1 1 Q(t+1)=D(t)	Q=0 Q_N=1; 0 0; 1 Q=1 Q_N=0; 1	PRESET D Q CLK Q_N CLEAR 74xx74;74xx174;74xx374
JK	CLK J K Q Q_N ↑ 0 0 Q Q_N(FM) ↑ 0 1 0 1 ↑ 1 0 1 0 ↑ 1 1 $\bar{Q}$ $\bar{Q}_N$ 0 – – Q Q_N(FM) 1 – – Q Q_N(FM)	Q(t) J K Q(t+1) 0 0 – 0 0 1 – 1 1 – 0 0 1 – 1 1 Tranzitia JK 0 → A A 1 → A $\bar{A}$ Q(t+1)=$\bar{J}$Q(t)+KQ(t)	Q=0 Q_N=1; 0 0; 1 Q=1 Q_N=0; 1	PRESET J Q CLK Q_N CLEAR 74xx76;74xx112;74xx109
T	CLK J K Q Q_N ↑ 0 0 Q Q_N ↑ 1 1 $\bar{Q}$ $\bar{Q}_N$ 0 – – Q Q_N(FM) 1 – – Q Q_N(FM) J= K=T	Q(t) T Q(t+1) 0 0 0 0 1 1 1 1 0 1 0 1 Q(t+1)=T$\oplus$ Q(t) pt T=1; Q(t+1)=Q(t)	Q=0 Q_N=1; 0 1; 1 Q=1 Q_N=0; 1	PRESET T Q CLK Q_N CLEAR

În Tabelul 3.4 sunt sintetizate informaţiile principale despre cele patru tipuri de circuite basculante prezentate.

Exemplul 3.15 Deoarece bistabilul T nu exista sub formă de circuit integrat independent să se modeleze funcţionarea sa cu circuite bistabile de tip SR, JK şi D.

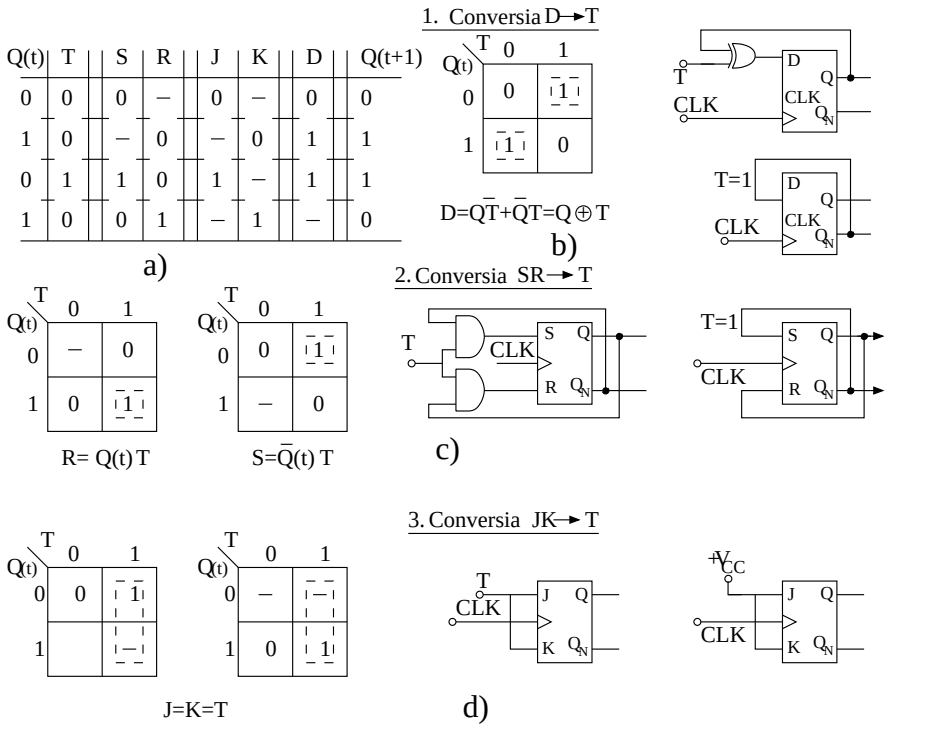


Figura 3.50 Conversia bistabilelor: a) tabelul de excitaţie pentru conversia SR, JK şi D în T; b) conversia $D \to T$; c) conversia $RS \to T$; d) conversia $JK \to T$.

Soluţie. Rezolvarea acestei probleme este similară cu cea pentru sinteza bistabilului JK prin modelarea funcţionării pe un automat pe bază de bistabil SR, Figura 3.47. Concret, trebuie realizate sinteze de automate pe bază de bistabile SR, JK şi D care modelează funcţionarea bistabilului T. Funcţionarea bistabilului T este descrisă prin tabelul său de excitaţie redat în coloanele 1, 2 şi 8 din Figura 3.50-a. Coloanele 3-4, 5-6 şi 7 sunt valorile respectiv pentru intrările SR, JK şi D, ale bistabilelor SR, JK şi D, care produc aceeaşi comutaţie între $Q(t)$ şi $Q(t+1)$ ca cea realizată de bistabilul T; aceste valori se citesc pentru fiecare bistabil din tabelele de excitaţie prezentate în Tabelul 3.4. Apoi, se face sinteza funcţie de excitaţie pentru fiecare din intrările S, R, J, K şi D pe o diagramă V-K având ca variabile de intrare pe $Q(t)$ şi T.

1. Conversia D→T, Figura 3.50-b. Rezultă funcţia de excitaţie pe intrarea D de forma $D = Q(t) \cdot \overline{T} + \overline{Q}(t) \cdot T = Q(t) \oplus T$, care este identică cu cea din relaţia 3.25, iar pentru $T = 1$ (basculă) se obţine relaţia $D = \overline{Q}(t)$. Este evident, că pentru funcţionarea bistabilului D ca basculă pe intrarea acestuia trebuie aplicată valoarea de pe ieşire dar negată, ceea ce se poate realiza printr-o reacţie fie de pe Q, printr-un element inversor comandat(XOR), fie direct de pe Q_N.

2. Conversia SR→T, Figura 3.50-c. Pentru funcţiile de excitaţie rezultă expresiile $S = \overline{Q}(t)T$; $R = Q(t)T$, iar pentru $T = 1$ se obţine $S = Q_N(t)$ şi $R = Q(t)$. Structurile de circuit obţinute conform acestor relaţii pot fi considerate şi ca fiind particularizări ale structurii de bistabil JK cu J=K=T, din Figura 3.47-h.

3. Conversia JK→T, Figura 3.50-d. Funcţiile de excitaţie sunt $J = K = T$, adică intrările bistabilului JK sunt conectate împreună generând astfel intrarea bistabilului T. De fapt, această metodă de conversie a stat la baza structurii bistabilului T, Figura 3.49.

Utilizând un procedeu ca cel folosit mai sus, pentru obţinerea bistabilului T, se poate obţine oricare tip de bistabil prin modelarea funcţionării acestuia pe celelalte tipuri de bistabile.

3.3.3 Aplicaţii la automate

În prezentarea funcţionării şi structurii unui automat, Figura 3.7, s-a arătat că pe calea de reacţie starea următoare $Q(t+1)$ devine stare prezentă prin memorarea sa într-un element de memorare (registru). Acest element, de memorare, este un circuit basculant bistabil fie de tip latch fie de tip bistabil. Dacă este de tip latch atunci automatul, pe durata de transparenţă a latch-ului, are o comportare de tip asincron pentru că reacţia este continuă pe aceasţa durată. Iar dacă este de tip bistabil atunci automatul este sincron pentru că reacţia este întreruptă, aceasta se închide numai pe frontul de basculare a bistabilului când starea următoare se înscrie ca stare prezentă. Tipul de bistabil determină puternic doar structura semiautomatului deoarece partea combinaţională a acestuia, CLC1, Figura 3.8, trebuie să calculeze funcţiile de excitaţie specifice tipului de bistabil utilizat. În continuare se vor prezenta exemple de automate implementate cu diferite tipuri de bistabile.

Exemplul 3.16 Pentru automatul, a cărui sinteză s-a prezentat în Exemplul 3.12 să se realizeze implementarea semiautomatului cu cu bistabile D, JK şi SR

Soluţie. În implementarea din Figura 3.35 nu s-a specificat ce tip de element de memorie utilizează în bucla de reacţie, acum această reprezentare generică de memorie va fi substituită cu bistabile D, bistabile JK sau bistabile SR. Din tabelul din Figura 3.34 s-a păstrat numai partea corespunzătoare tranziţiei stărilor, fără ieşiri, şi s-a completat cu valorile funcţiilor de excitaţie pentru definirea semiautomatului implementabil cu bistabile D, JK şi SR în Figura 3.51-a. Valoarea funcţiei de excitaţie, pentru fiecare intrare a celor trei bistabile, se obţine prin citirea acesteia din tabelul de excitaţie al bistabilului respectiv, Tabelul 3.4, luând în considerare pe toate căile de tranziţie, comutaţiile între bitul z_i (starea prezentă) şi $wD_{i,i=0,1,2}$ (starea următoare). De exemplu, pentru calea de tranziţie L_5 se realizează tranziţia de la starea prezentă $z_2z_1z_0$=010 (q_1) la starea următoare $wD_1wD_2wD_3 = 011$ (q_4), adică printr-o comutaţie a biţilor de stare în felul următor $z_2 = 0 \rightarrow wD_2 = 0$, $z_1 = 1 \rightarrow wD_2 = 1$ şi $z_0 = 0 \rightarrow wD_3 = 1$, deci atunci la o implementare cu bistabile JK, se obţin, din tabelul de excitaţie al acestui bistabil, respectiv următoarele valori ale biţilor funcţiilor de excitaţie: $wJ_2\, wK_2 = 0-$; $wJ_1\, wK_1 = -0$; $wJ_0\, wK_0 = 1-$.

Pentru implementarea elementului de memorare cu bistabil de tip D, valoarea funcţiei de excitaţie este identică cu valoarea bitului stării următoare a bistabilului, conform relaţiei 3.22, de aceea în tabel biţii stării următoare a automatului sunt notaţi cu wD_2, wD_1, wD_0. Expresiile funcţiilor de excitaţie pentru bistabilul D, determinate în Exemplul 3.12, sunt:

Cale de tranz	Intrari x_2	x_1	x_0	Starea prezenta Q(t) Simb	z_0	z_1	z_2	Starea urmatoare (functie de excitatie pt bistabil D) Q(t+1) Simb	wD_1	wD_0	Functiile de excitatie pentru intrarile J K ale bistabilelor wJ_2	wK_2	wJ_1	wK_1	wJ_0	wK_0	Functiile de excitatie pentru intrarile S R ale bistabilelor wS_2	wR_2	wS_1	wR_1	wS_0	wR_0
L_1	0	0	–	q_0	0	0	0	q_0	0	0	0	–	0	–	0	–	0	–	0	–	0	–
L_2	1	–	–	q_0	0	0	0	q_1	0	1	0	–	1	–	0	–	0	–	1	–	0	–
L_3	0	1	–	q_0	0	0	0	q_2	0	0	1	–	0	–	1	–	0	–	0	–	1	0
L_4	–	–	0	q_1	0	1	0	q_3	1	1	0	–	–	0	0	–	1	0	–	0	0	–
L_5	–	–	1	q_1	0	1	0	q_4	0	1	1	–	–	0	1	–	0	–	0	0	1	0
L_6	0	–	–	q_2	0	0	1	q_0	0	0	0	–	–	1	–	1	0	–	1	–	0	1
L_7	1	–	–	q_2	0	0	1	q_4	0	1	1	–	–	0	–	0	0	–	–	0	–	0
L_8	–	0	–	q_3	1	1	0	q_4	0	1	1	–	–	0	1	–	0	1	–	0	1	0
L_9	–	1	–	q_3	1	1	0	q_5	1	0	0	–	–	1	0	–	–	0	0	1	0	–
L_{10}	–	–	–	q_4	0	1	1	q_0	0	0	0	0	–	1	–	1	0	–	0	1	0	1
L_{11}	–	–	–	q_5	1	0	0	q_0	0	0	0	–	1	0	–	0	0	1	0	–	0	–

a)

z_2 \ z_1z_0	00	01	11	10
0	0	0	0	$\bar{x}_0$
1	–	–	–	–

$wJ_2 = \bar{x}_0 z_1 \bar{z}_0$

z_2 \ z_1z_0	00	01	11	10
0	x_2	x_2	–	–
1	0	–	–	–

$wJ_1 = x_2 \bar{z}_2$

z_2 \ z_1z_0	00	01	11	10
0	$\bar{x}_2 x_1$	–	–	x_0
1	0	–	–	$\bar{x}_1$

$wJ_0 = \bar{x}_2 x_1 \bar{z}_2 \bar{z}_1 + x_0 \bar{z}_2 z_1 + \bar{x}_1 z_2 z_1$

z_2 \ z_1z_0	00	01	11	10
0	–	–	–	–
1	1	–	–	$\bar{x}_1$

b) $wK_2 = \bar{z}_1 + \bar{x}_1 = \overline{z_1 x_1}$

z_2 \ z_1z_0	00	01	11	10
0	–	–	1	0
1	–	–	–	x_1

$wK_1 = z_0 + x_1 z_2$

z_2 \ z_1z_0	00	01	11	10
0	–	$\bar{x}_2$	1	–
1	0	–	–	–

$wK_0 = z_1 + \bar{x}_2 \bar{z}_2$

z_2 \ z_1z_0	00	01	11	10
0	0	0	0	$\bar{x}_0$
1	0	–	–	0

$wS_2 = \bar{x}_0 \bar{z}_2 z_1 \bar{z}_0$

z_2 \ z_1z_0	00	01	11	10
0	x_2	x_2	0	–
1	0	–	–	0

$wS_1 = x_2 \bar{z}_2 \bar{z}_1$

z_2 \ z_1z_0	00	01	11	10
0	$\bar{x}_2 x_1$	0	0	x_0
1	0	–	–	$\bar{x}_1$

$wS_0 = \bar{x}_2 x_1 \bar{z}_2 \bar{z}_1 \bar{z}_0 + x_0 \bar{z}_2 z_1 \bar{z}_0 + \bar{x}_1 z_2 z_1$

z_2 \ z_1z_0	00	01	11	10
0	–	–	–	0
1	1	–	–	$\bar{x}_1$

c) $wR_2 = \bar{z}_1 + \bar{x}_1 z_2$

z_2 \ z_1z_0	00	01	11	10
0	0	0	1	0
1	–	–	–	$\bar{x}_1$

$wR_1 = z_2 z_1 + \bar{x}_1 z_2$

z_2 \ z_1z_0	00	01	11	10
0	0	$\bar{x}_2$	1	0
1	–	–	–	0

$wR_0 = z_2 z_1 + \bar{x}_2 z_0$

Figura 3.51 Explicativă pentru Exemplul 3.13: a) tabelul cu valorile funcţiilor de excitaţie; b,c) diagramele V-K pentru sinteza funcţiilor de excitaţie la bistabilele JK şi SR.

$$\begin{aligned} wD_2 &= \overline{x}_0\overline{z}_2 z_1\overline{z}_0 + x_1 z_2 z_1 \\ wD_1 &= x_1\overline{z}_2\overline{z}_1 + \overline{z}_2 z_1\overline{z}_0 + \overline{x}_1 z_2 z_1 \\ wD_0 &= \overline{x}_2 x_1\overline{z}_2\overline{z}_1\overline{z}_0 + \overline{z}_2\overline{z}_1 z_0 + \overline{x}_1 z_2 z_1 \end{aligned} \tag{3.27-a}$$

Din tabelul completat cu valorile funcţiilor de excitaţie se pot deduce expresiile logice ale acestora dar pentru sinteză vor fi necesare diagrame V-K de şase variabile de intrare: x_2, x_1, x_0, z_2, z_1, z_0. Se poate reduce dimensiunea acestor diagrame la diagrame de numai trei variabile z_2, z_1, z_0 prin introducerea unei variabile de intrare ca variabilă reziduu (un exemplu cum se introduce doar o singură variabilă reziduu este prezentat în Figura 3.28-d). În starea q_0, testându-se două intrări, modalitatea cum se introduc simultan aceste două variabile reziduu în coeficientul funcţiei wJ_0 se prezintă în continuare. Prima linie şi a patra din tabelul alăturat există definite din tabelul din Figura 3.51-a adică $x_2 = 0$, $x_1 = 0$ iar $wJ_0 = 0$; $x_2 = 0$, $x_1 = 1$ iar $wJ_0 = 1$. Pentru liniile doi şi trei se extinde linia a doua din tabelul din Figura 3.51-a, $x_2 = 1$, $x_1 = -$ iar $wJ_0 = 0$, la următoarele două: $x_2 = 1$, $x_1 = 0$ iar $wJ_0 = 0$ şi $x_2 = 1$ $x_1 = 1$ iar $wJ_0 = 0$. Expresia funcţiei de excitaţie $wJ_0 = \overline{x}_2 x_1$ s-a dedus din analiza tabelului alăturat. Se obţin următoarele expresii pentru funcţiile de excitaţie, Figura 3.51-b şi 3.51-c:

x_2	x_1	wJ_0	
0	0	0	
1	0	0	$\Rightarrow \{\ wJ_0 = \overline{x}_2 x_1$
1	1	0	
0	1	1	

Bistabilul JK

$$\begin{array}{lll} wJ_2 = \overline{x}_0 z_1\overline{z}_0 & wJ_1 = x_2\overline{z}_2 & wJ_0 = \overline{x}_2 x_1\overline{z}_2\overline{z}_1 + x_0\overline{z}_0 z_1 + \overline{x}_1 z_2 z_1 \\ wK_2 = \overline{z_1 x_1} & wK_1 = z_0 + x_1 z_2 & wK_0 = z_1 + \overline{x}_2\overline{z}_2 \end{array} \tag{3.27-b}$$

Bistabilul SR

$$\begin{array}{lll} wS_2 = \overline{x}_0\overline{z}_2 z_1\overline{z}_0 & wS_1 = x_2\overline{z}_2\overline{z}_1 & wS_0 = \overline{x}_2 x_1\overline{z}_2\overline{z}_1\overline{z}_0 + x_0\overline{z}_2 z_1\overline{z}_0 + \overline{x}_1 z_2 z_1 \\ wR_2 = \overline{z}_1 + \overline{x}_1 z_2 & wR_1 = z_2 z_1 + \overline{x}_1 z_2 & wR_0 = z_2 z_1 + \overline{x}_2 z_0 \end{array} \tag{3.27-c}$$

Implementarea părţii combinaţionale, conform relaţiilor 3.27-b şi 3.27-c, se poate face prin oricare modalitate de implementare de circuit combinaţional (porţi logice, DMUX+porţi logice, MUX, ROM sau PLA).

În implementarea automatelor se pune întrebarea pentru realizarea elementelor de memorare, din bucla de reacţie, care din cele trei tipuri de bistabile (D, JK şi SR) este indicat a se utiliza (nu s-a considerat şi bistabilul T deoarece acesta se obţine simplu dintr-un JK). Răspunsul este mai nuanţat.

Bistabilul D necesită pentru fiecare bit de stare z_i doar o singură funcţie de excitaţie aplicată pe intrarea D. Bistabilele JK şi SR necesită pentru fiecare bit de stare z_i două funcţii de excitaţie pe intrare respectiv wJ_i,wK_i pentru bistabilul JK şi wS_i,wR_i pentru bistabilul SR. În schimb comenzile aplicate pe intrările bistabilelor JK şi SR sunt vag formulate, de forma $JK = A-$, $JK = -\overline{A}$, în raport cu o comandă exact prescrisă ce se aplică la bistabilul D, vezi Tabelul 3.4. Această ambiguitate în comandă determină valori don′t care în diagramele V-K de sinteză, deci funcţiile de excitaţie pentru bistabilul JK sunt cele mai simple apoi cele pentru bistabilul SR în

raport cu funcţia de excitaţie pentru bistabilul D. Se poate constata validitatea acestei afirmaţii prin analiza funcţiilor de excitaţie obţinute în exemplul anterior relaţiile 3.27-a,b şi c. Pe un circuit integrat discret, în general, sunt incluse mai multe bistabile de tip D decât bistabile JK, dar cu un bistabil JK se poate modela uşor un D sau un T, în general are şi intrări asincrone, deci este mai versatil în aplicaţii.

În concluzie, pentru implementările cu circuite integrate discrete pentru sisteme simple bistabilul de tip D este preferat bistabilului JK, pe când la sisteme mai complexe bistabilul JK prin versatilitatea sa este de preferat.În schimb, în structurile integrate sunt utilizate aproape în exclusivitate bistabilele de tip D (completate cu multplexoare pe intrare).

Proiectarea şi implementarea unui circuit automat pot fi mult simplificate, în unele cazuri, prin **utilizarea circuitelor multiplexoare**. În primul rând o astfel de simplificare apare în cazul în care fiecare bloc de stare al automatului se testează doar o singură variabilă de intrare, caz în care acea variabilă poate fi introdusă ca variabilă reziduu. Simplificarea poate apare chiar şi când se testează mai multe intrări în unele blocuri de stare, dar expresiile coeficienţilor funcţiei pot fi calculate cu operatori simpli (AND,OR...) de variabilele reziduu. Sau, organigrama ASM a funcţiei, în blocurile unde se testează mai mult de o variabilă, se poate aduce, prin introducerea de stări noi între testarea a două variabile, Figura 3.52-c, numai la blocuri în care se testează doar o singură variabilă. Această simplificare este susţinută mai departe, in implementarea semiautomatului, dacă se alege ca element de memorare în buclă bistabilul de tip D.

În al doilea rând, simplificarea poate apare prin micşorarea numărului de intrări aplicate părţii combinaţionale prin utilizarea de circuite multiplexoare. Dacă în fiecare bloc de stare din cele n intrări se testează doar un număr q intrări, atunci aceste intrări pot fi selectate prin q circuite multiplexoare. Conectând intrările în automat corespunzător pe intrările de date ale multiplexoarelor şi aplicând cuvântul codului de stare pe intrările de selectare ale multiplexoarelor, atunci se obţin pentru partea combinaţională doar q intrări în fiecare moment. Această modalitate este foarte indicată la realizarea părţii combinaţionale pe circuit ROM. Deoarece capacitatea circuitului ROM depinde exponenţial de numărul de intrări, chiar şi reducerea unei singure intrări la automat va determina o înjumătăţire a capacităţii circuitului ROM.

Exemplul 3.17 Pentru automatul din Exemplul 3.12 să se structureze partea de semiautomat numai cu multiplexoare şi bistabile de tip D.

Soluţie. Tabelul de tranziţie al stărilor din Figura 3.51-a este refăcut în Figura 3.52 dar de această dată valorile funcţiilor de excitaţie ale bistabilelor wD_2, wD_1, wD_0 conţin intrările x_2, x_1, x_0 ca variabile reziduu.

Aceste funcţii de excitaţie sunt implementate pe MUX8:1 care au ca variabile de selectare biţii cuvântului de stare $z_2z_1z_0$, Figura 3.52-b. Aceeaşi implementare pe multiplexoare se poate realiza pornind de la relaţiile 3.27-a care se extind la forme canonice în funcţie de cele trei variabile z_2, z_1, z_0. Implementarea cu multiplexoare şi bistabile D apare ca o variantă atractivă de automat deoarece se realizează fără efort de sinteză deosebit şi cu un cost relativ scăzut .

Exemplul 3.18 Automatul din Exemplul 3.12 să se implementeze cu bistabile D şi circuite ROM.

Soluţie. Pentru automatul propus, tabelul de tranziţie al stărilor şi al ieşirilor, din

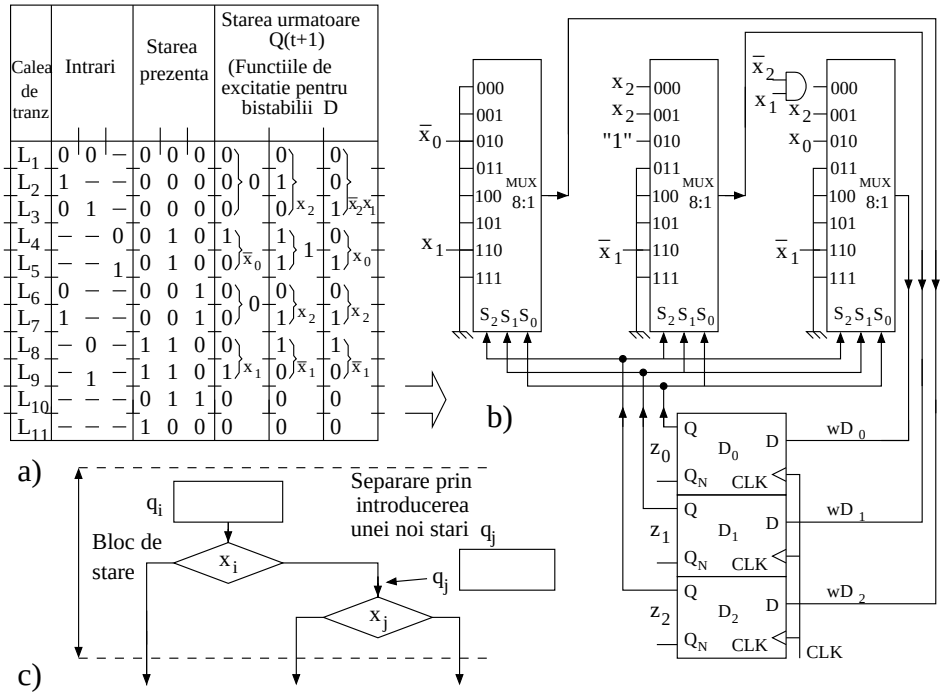


Figura 3.52 Explicativă pentru Exemplul 3.17: a) tabelul de tranziţie al stărilor cu intrările automatului introduse ca variabile reziduu; b) implementarea cu bistabile D şi mutiplexoare; c) eliminarea testării a două variabile într-un bloc de stare prin introducerea unei noi stări.

Figura 3.34, este o formă simplificată deoarece conţine numai tranziţiile pe cele 11 căi de tranziţie ($L_1, L_2, \ldots, L_{11}$) din tabelul de cu 64 tranziţii posibile. Automatul codificat cu trei biţi z_2, z_1, z_0 poate realiza în fiecare din cele opt stări câte opt tranziţii care corespund combinaţiilor de variabile de intrare x_2, x_1, x_0 testate. Informaţia privind efectul produs de o tranziţie necesită câte o locaţie, rezultă că este necesară o memorie cu $8 \times 8 = 64$ adrese, cuvântul de adresare fiind format din trei variabile de stare şi trei de intrare, $z_2 z_1 z_0 x_2 x_1 x_0$. Componenţa cuvântului de 1 byte dintr-o locaţie este: $D_7 = wD_2$; $D_6 = wD_1$; $D_5 = wD_0$; $D_4 = y_3$; $D_3 = y_5$; $D_2 = y_4$; $D_2 = y_2\_L$; $D_0 = y_1$. Pentru maparea pe memorie a tabelului de tranziţie al stărilor şi ieşirilor acesta trebuie completat la toate cele 64 de adrese ($000000 \div 111111$) prin extinderea atât pentru simbolurile indiferente ale subcuvântului $x_2x_1x_0$ cât şi pentru codurile neutilizate pentru ale stărilor 101 şi 111. Rezultatul extinderii este prezentat în Figura 3.53-a, unde conţinutul fiecărei locaţii este exprimat în hexazecimal şi pentru detaliere s-a specificat la fiecare locaţie corespondenţa la starea şi calea de tranziţie. Capacitatea circuitului ROM este de 64×1 byte iar structura automatului este prezentată în Figura 3.53-b.În general, pentru un automat cu k variabile de stare, n variabile de intrare şi m biţi de ieşire capacitatea necesară este $2^{k+n} \times (k+m)$ biţi.

Modalitatea de micşorare a capacităţii circuitului ROM utilizat în implementarea automatului este, ca şi în abordarea clasică aplicată la oricare memorie, fie de micşorare a numărului de intrări, fie de reducere a numărului de biţi de ieşire, evident că prima variantă este mult mai eficientă deoarece are o contribuţie exponenţială. Reducerea numărului de

Adresa z_2	z_1	z_0	x_2	x_1	x_0	Bloc de stare		Continut locatie (HEX)	
0	0	0	0	0	0	q_0	L_1	0	3
0	0	0	0	0	1	q_0	L_1	0	3
0	0	0	0	1	0	q_0	L_3	2	3
0	0	0	0	1	1	q_0	L_3	2	3
0	0	0	0	0	0	q_0	L_2	4	1
0	0	0	1	0	1	q_0	L_2	4	1
0	0	0	1	1	0	q_0	L_2	4	1
0	0	0	1	1	1	q_0	L_2	4	1
0	0	1	1	0	0	q_2	L_6	1	3
0	0	1	0	0	1	q_2	L_6	1	3
0	0	1	0	1	0	q_2	L_6	1	3
0	0	1	0	1	1	q_2	L_6	1	3
0	0	1	0	0	0	q_2	L_7	7	7
0	0	1	1	0	1	q_2	L_7	7	7
0	0	1	1	1	0	q_2	L_7	7	7
0	0	1	1	1	1	q_2	L_7	7	7
0	1	0	0	0	0	q_1	L_4	C	2
0	1	0	0	0	1	q_1	L_5	6	6
0	1	0	0	1	0	q_1	L_4	C	2
0	1	0	0	1	1	q_1	L_5	6	6
0	1	0	1	0	0	q_1	L_4	C	2

Adresa z_2	z_1	z_0	x_2	x_1	x_0	Bloc de stare		Continut locatie (HEX)	
0	1	0	1	0	1	q_1	L_5	6	6
0	1	0	1	1	0	q_1	L_4	C	2
0	1	0	1	1	1	q_1	L_5	6	6
0	1	1	0	0	0	q_4	L_{10}	0	C
0	1	1	0	0	1	q_4	L_{10}	0	C
0	1	1	0	1	0	q_4	L_{10}	0	C
0	1	1	0	1	1	q_4	L_{10}	0	C
0	1	1	1	0	0	q_4	L_{10}	0	C
0	1	1	1	0	1	q_4	L_{10}	0	C
0	1	1	1	1	0	q_4	L_{10}	0	C
0	1	1	1	1	1	q_4	L_{10}	0	C
1	0	0	0	0	0	q_5	L_{11}	1	2
1	0	0	0	0	1	q_5	L_{11}	1	2
1	0	0	0	1	0	q_5	L_{11}	1	2
1	0	0	0	1	1	q_5	L_{11}	1	2
1	0	0	1	0	0	q_5	L_{11}	1	2
1	0	0	1	0	1	q_5	L_{11}	1	2
1	0	0	1	1	0	q_5	L_{11}	1	2
1	0	0	1	1	1	q_5	L_{11}	1	2
1	0	1	0	0	0	–	–	–	–
1	0	1	0	0	1	–	–	–	–

Adresa z_2	z_1	z_0	x_2	x_1	x_0	Bloc de stare		Continut locatie (HEX)	
1	0	1	0	1	0	–	–	–	–
1	0	1	1	0	1	–	–	–	–
1	0	1	1	1	0	–	–	–	–
1	0	1	1	1	1	–	–	–	–
1	1	0	0	0	0	q_3	L_8	6	A
1	1	0	0	0	1	q_3	L_8	6	A
1	1	0	0	1	0	q_3	L_9	8	B
1	1	0	0	1	1	q_3	L_9	8	B
1	1	0	1	0	0	q_3	L_8	6	A
1	1	0	1	0	1	q_3	L_8	6	A
1	1	0	1	1	0	q_3	L_9	8	B
1	1	0	1	1	1	q_3	L_9	8	B
1	1	1	0	0	0	–	–	–	–
1	1	1	0	0	1	–	–	–	–
1	1	1	0	1	0	–	–	–	–
1	1	1	0	1	1	–	–	–	–
1	1	1	1	0	0	–	–	–	–
1	1	1	1	0	1	–	–	–	–
1	1	1	1	1	0	–	–	–	–
1	1	1	1	1	1	–	–	–	–

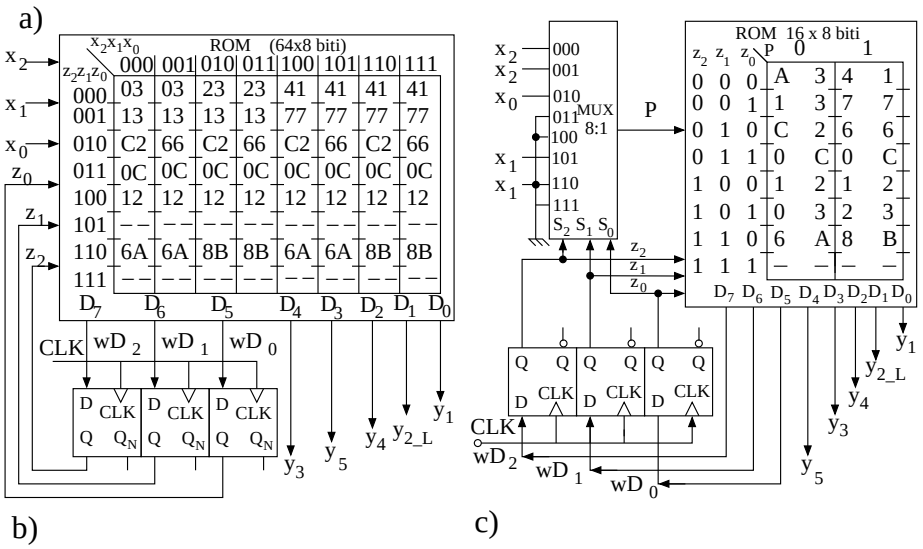


Figura 3.53 Explicativă pentru Exemplul 3.18: a) tabelul de tranziţie al stărilor şi al ieşirilor extins pentru maparea într-un circuit ROM de capacitate 64×8 biţi; b) structurarea automatului pe un circuit ROM şi bistabile D; c) structurarea aceluiaşi automat când se testează doar o singură intrare în fiecare bloc de stare.

intrări se poate aplica cu succes la automatele care testează în fiecare stare doar o singură intrare din cele n, deci cuvântul de adresare rezultă cu lungimea de $(k+1)$ biţi, capacitatea circuitului ROM rezultă de $2^{k+1} \times (k+m)$ biţi, se obţine o reducere de 2^{n-1} ori. Pentru automatul din acest exemplu nu putem (încă!) aplica această modalitate deoarece în starea q_0 (000) se testează simultan atât x_2 cât şi x_1.

Automatul poate fi redus la un automat care testează în fiecare stare doar o singură intrare dacă se aplică procedeul prezentat în Figura 3.52-c, adică introducerea unei stări noi q_6 între punctele de testare ale variabilelor de intrare x_2 şi x_1 din blocul de stare q_0. Rezultă astfel: 1-în blocul de stare q_0 (000) tranziţia $q_0 \rightarrow q_6$ pe calea $L_1 = \overline{x_2}$ şi tranziţia $q_0 \rightarrow q_1$ pe calea $L_2 = x_2$; 2-în blocul de stare q_6 (101) tranziţia $q_6 \rightarrow q_0$ pe cale $L_{12} = \overline{x_1}$ şi tranziţia $q_6 \rightarrow q_2$ pe calea $L_3 = x_1$. Ieşirile în starea q_6 sunt identice cu cele din starea q_0, ceea ce apare pentru funcţionarea automatului, când x_2=0, ca o staţionare de doua tacte în q_0. Tabelul automatului din Figura 3.34 se modifică prin introducerea stării q_6 şi a noilor căi de tranziţie L_1, L_{12} şi L_3 (ceea ce trebuie introdus şi în forma expandată din Figura 3.53-a).

Pentru automatul modificat structurarea este cea din Figura 3.53-c. În fiecare bloc de stare prin cuvântul de cod al stării $z_2z_1z_0$, aplicat pe intrările de selectare al MUX8:1, se selectează doar valoarea variabilei de intrare testate, notată cu P, deci cuvântul de adresare este $z_2z_1z_0P$. La adresa $z_2z_1z_00$ este stocată informaţia din blocul respectiv de stare când tranziţia are loc după calea $P = 0$, iar la adresa $z_2z_1z_01$ este stocată informaţia după calea de tranziţie $P = 1$. Această structurare se poate extinde şi pentru cazul când sunt testate simultan două intrări într-un bloc de stare, în cazul general capacitatea memoriei fiind de $2^{k+2} \times (k+m)$ biţi. Pentru această implementare pe intrare sunt necesare două multiplexoare care în fiecare bloc de stare, prin cuvântul de cod $z_2z_1z_0$, selectează cele două variabile testate notate cu P_1 şi P_0, iar cuvântul de adresare la memorie are forma $z_2z_1z_0P_2P_1$. Fiecare stare necesită patru locaţii(/adrese) pentru stocarea informaţiilor corespunzătoare cele patru căi de tranziţie $P_1P_0 = 00$, 01, 10 şi 11.

Bitul de adresare P, identic cu valoarea variabilei de intrare testate într-un bloc de stare şi utilizat pentru selectarea a uneia din cele două adrese corespunzătoare celor două căi de tranziţie, poate fi utilizat pentru selectare nu ca un bit de adresare ci ca un bit de validare asupra cuvântului citit din locaţia de adresă $z_2z_1z_0$. În această variantă câmpul cuvântului stocat la o adresă are lungimea dublă, deoarece cuprinde atât semicuvântul cu informaţie pentru $P = 0$ cât şi semicuvântul cu informaţie pentru $P = 1$, plus încă un subcuvânt care conţine valorile intrărilor $x_2x_1x_0$. Notând $[z_2z_1z_0]$ conţinutul unei locaţii de adresă $z_2z_1z_0$ atunci din tabelul memoriei de la Figura 3.53-c se poate obţine conţinuturile locaţiilor de la această variantă prin următoarea concatenare $x_2x_1x_0[z_2z_1z_00][z_2z_1z_01]$. Una din intrări din acest subcuvânt, pentru un bloc de stare $z_2z_1z_0$, va selecta prin valoarea sa, din cuvântul stocat la adresa $z_2z_1z_0$, fie semicuvântul pentru $P = 0$, fie semicuvântul pentru $P = 1$.

Implementarea automatului conform acestei variante de utilizare a bitului variabilei testate este prezentată în Figura 3.54. Pentru fiecare bloc de stare $z_2z_1z_0$, în subcuvântul $x_2x_1x_0$, intrarea care se testează are valoarea 1. De exemplu, dacă în blocul de stare $z_2z_1z_0 = 001$, când se tastează x_2 $(= 1)$ şi dacă într-adevăr intrarea x_2 are valoarea 1 atunci prin poarta P_2 se aplică la grupul 8×MUX 2:1 selectarea $P = 1$, iar pe ieşire rezultă semicuvântul 01110111 (77H), altfel rezultă semicuvântul pentru $P = 0$, adică 00010011 (13H). În cazul general, când se testează doar o singură intrare în fiecare bloc de stare, capacitatea memoriei este $2^k \times (n + 2k + 2m)$.

Această modalitate de implementare se poate extinde şi la cazul când se testează două intrări în fiecare bloc de stare. De această dată cuvântul din fiecare locaţie cuprinde patru subcuvinte, corespunzătoare celor patru tranziţii posibile, plus încă două subcuvinte de variabilele $x_2x_1x_0$ fiecare indicând una din cele două variabile care se testează. Evident, se generează doi biţi de selectare P_1 şi P_0 prin a căror combinaţii 00, 01, 10 şi 11 se selectează una din intrările grupului de 8×MUX 4:1. Capacitatea necesară a circuitului ROM este

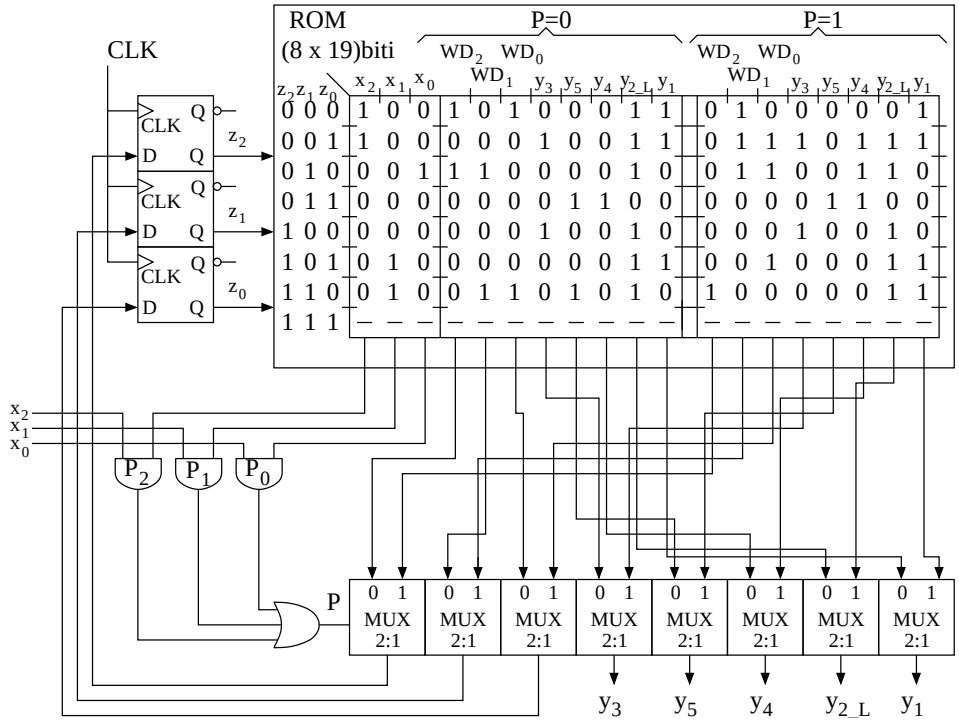


Figura 3.54 Explicativă pentru Exemplul 3.18. Utilizarea valorii variabilei testate într-un bloc de stare pentru selectarea subcâmpurilor cuvântului stocat la o adresă $z_2z_1z_0$.

$2^k \times (2n + 4k + 4m)$.

Pentru o funcţionare corectă a automatului prin relaţia 3.7 se poate calcula perioada minimă a semnalului de ceas. În cazul când automatul are cel puţin o variabilă de intrare sincronă trebuie luat în considerare şi timpul de transfer al acestei variabile din momentul aplicării impulsului de ceas la bistabilul de intrare (de sincronizare). Acest timp de transfer este compus din: τ_{pBI} – timpul de propagare prin bistabilul de sincronizare de intrare plus τ_{pmaxCLK}-timpul maxim de propagare prin reţeaua combinaţională, plus τ_{SUQ}-timpul de prestabilire la bistabilul de stare. Deoarece se comandă cu acelaşi semnal de ceas atât bistabilele din registrul de sincronizare de pe intrare cât şi cele din registrul de stare perioada minimă trebuie să respecte relaţia:

$$T_{CLKmin} \geq max\left\{(\tau_{\text{pmaxCLK}} + \tau_{\text{pBQ}} + \tau_{\text{SUQ}}), (\tau_{\text{pmaxCLK}} + \tau_{\text{pBI}} + \tau_{\text{SUQ}})\right\} \qquad (3.28)$$

Sinteza automatelor prezentate până acum a urmat, mai mult sau mai puţin, o succesiune de faze descrise în secţiunea 3.2.6, ultima fază fiind elaborarea documentaţiei. În practică, sunt frecvente cazurile când pentru un automat nu există documentaţie şi astfel, s-ar putea, să nu se înţeleagă funcţionarea sa. Funcţionarea automatului

este descrisă clar printr-o diagramă de tranziţie a stărilor sau un graf de tranziţie/organigramă ASM. Soluţia? pornind de la structura automatului(schema electrică) şi parcurgând în sens invers succesiunea procesului de sinteză, se ajunge la tabelul de tranziţie al stărilor şi graful de tranziţie al stărilor. Acest proces, invers al sintezei, este analiza automatului. Etapele care se parcurg în procesul de analiză, pornind de la schema electrică a automatului sunt:

1. Identificarea structurii automatului. Aceasta înseamnă segregarea părţii combinaţionale de cea de memorare (bistabilele din bucla de reacţie şi identificarea acestora – sunt cazuri când sunt utilizate mai multe tipuri de bistabile). Uneori, de mare folos este redresarea schemei sub forma clasică de automat ca în Figura 3.7 sau 3.8. Se identifică intrările principale, ieşirile, variabilele de stare (intrările secundare), variabilele funcţii de excitaţie şi, dacă este necesar, alegerea unor denumiri/simboluri pentru acestea.

2. Din analiza părţii combinaţionale se deduc ecuaţiile logice ale ieşirilor funcţiilor de excitaţie, în funcţie de cele n variabile de intrare şi k variabile de stare.

3. Construirea tabelului de tranziţie al stărilor şi al ieşirilor. Pentru toate combinatiile între cele 2^k stări prezente şi 2^n cuvinte de intrare se determină stările următoare prin utilizarea relaţiilor deduse pentru funcţiile de excitaţie şi tabelele caracteristice (vezi Tabelul 3.4) ale tipurilor de bistabile folosite. Stările următoare se pot calcula şi prin: introducerea expresiilor funcţiilor de excitaţie în ecuaţiile de funcţionare ale bistabilelor folosite; introducând în expresiile rezultate toate combinaţiile de 2^k stări prezente şi 2^n cuvinte de intrare, obţinându-se astfel valorile biţilor stărilor următoare.

4. Construirea grafului de tranziţie al stărilor/organigrama ASM. Asigurând simboluri literale pentru fiecare cod de stare, din tabelul de tranziţie al stărilor şi ieşirilor, se poate desena graful de tranziţie. Eventual, dacă nu sunt multe variabile, se poate trasa diagrama de variaţie în timp a semnalelor.

Exemplul 3.19 Pentru automatul cu schema electrică din figura 3.55-a să se deducă graful de tranziţie al stărilor.

Soluţie: Analiza acestui automat se va face marcând etapele specificate anterior.

1. Structura automatului poate fi redesenată ca în Figura 3.55-b. Se observă că ieşirile sunt de fapt tocmai biţii de stare z_1, z_0, deci este un automat Moore.

2. Funcţiile de excitaţie au expresiile:

$$wJ_1 = z_0, wK_1 = \overline{x}z_0$$
$$wJ_0 = \overline{x}, wK_0 = x\overline{z_1} + \overline{x}z_1 = x \oplus z_1$$

3. Tabelul de tranziţie al stărilor, Figura 3.55-c. Se determină starea următoare $z_1(t+1)z_0(t+1)$ pentru fiecare din cele opt combinaţii ale intrării x şi stării prezente $z_1 z_0$.

În tabel s-au introdus şi valorile intrărilor wJ_1, wK_1, wJ_0, wK_0 ale celor două bistabile JK (coloanele 4,5,6,7), valori calculate cu ajutorul relaţiilor funcţiilor de excitaţie pentru toate cele opt combinaţii de intrare. Valorile din coloanele 4,5,6 şi 7 nu fac parte din tabel dar s-au introdus pentru o mai uşoară determinare a stării următoare din starea prezentă. Utilizând tabelul caracteristic al bistabilului JK, vezi Tabelul 3.4, pentru fiecare configuraţie de intrare xz_1z_0 se deduc valorile biţilor stării următoare $z_1(t+1)z_0(t+1)$.

Expresiile biţilor stării următoare se pot obţine şi prin introducerea funcţiilor de excitaţie

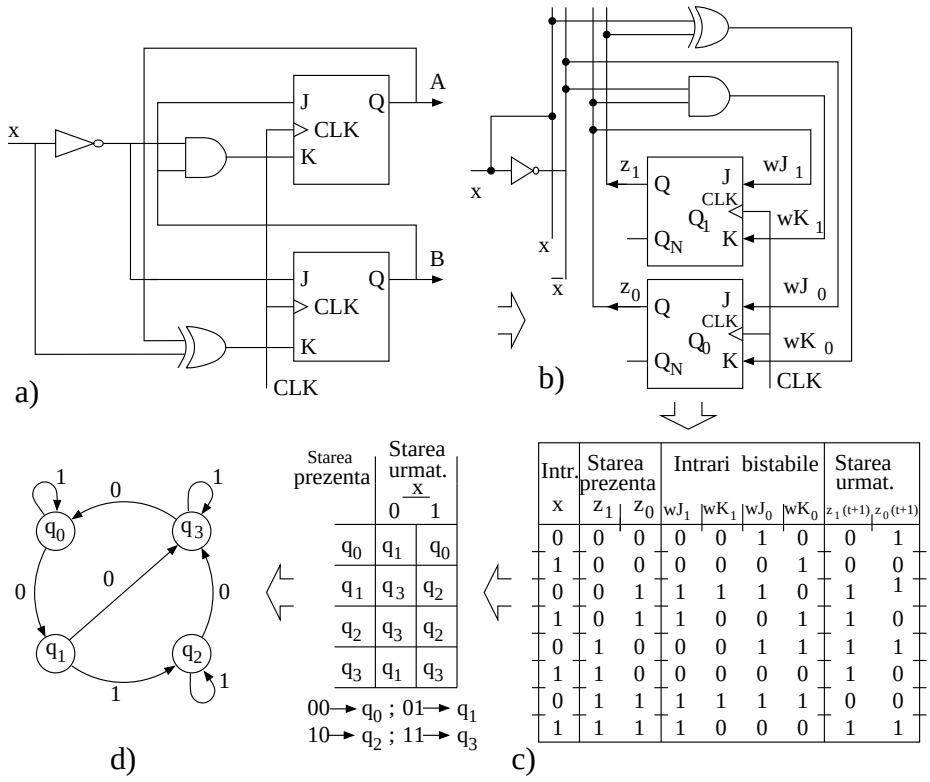


Figura 3.55 Explicativă la Exemplul 3.19: a) schema electrică a automatului; b) schema electrică redesenată în structura clasică de automat; c) tabelul de tranziţie al stărilor; d) graful de tranziţie al stărilor.

deduse anterior în relaţia 3.27 de funcţionare a bistabilului JK:

$$\begin{aligned} z_1(t+1) &= w_1J_1 \cdot \overline{z}_1(t) + \overline{wK_1} \cdot z_1(t) = z_0 \cdot \overline{z}_1(t) + (x + \overline{z}_0(t))z_1(t) = \\ &= \overline{z}_1 z_0 + xz_1 + z_1\overline{z}_0 \\ z_2(t+1) &= wJ_0 \cdot \overline{z}_0(t) + \overline{wK_0} \cdot z_0(t) = \overline{x} \cdot \overline{z}_0(t) + (\overline{x \oplus z_1(t)}) \cdot z_0(t) = \\ &= \overline{x}_0\overline{z}_0 + \overline{x}\,\overline{z}_1 z_0 + xz_1 z_0 \end{aligned}$$

În ultima formă s-a scris z în loc de $z(t)$. Valorile din coloanele 8 şi 9 din tabelul de tranziţie al stărilor se pot calcula şi cu aceste relaţii pentru toate configuraţiile xz_1z_0.

4. Asignând codurile de stare în felul următor: $00 \to q_0$, $01 \to q_1$, $10 \to q_2$, $11 \to q_3$ se obţine graful de tranziţie al stărilor, Figura 3.55-d.

3.3.4 Circuitul basculant bistabil asimetric (Triggerul Schmitt)

Circuitul asincron obţinut prin legături simetrice încrucişate între două inversoare,

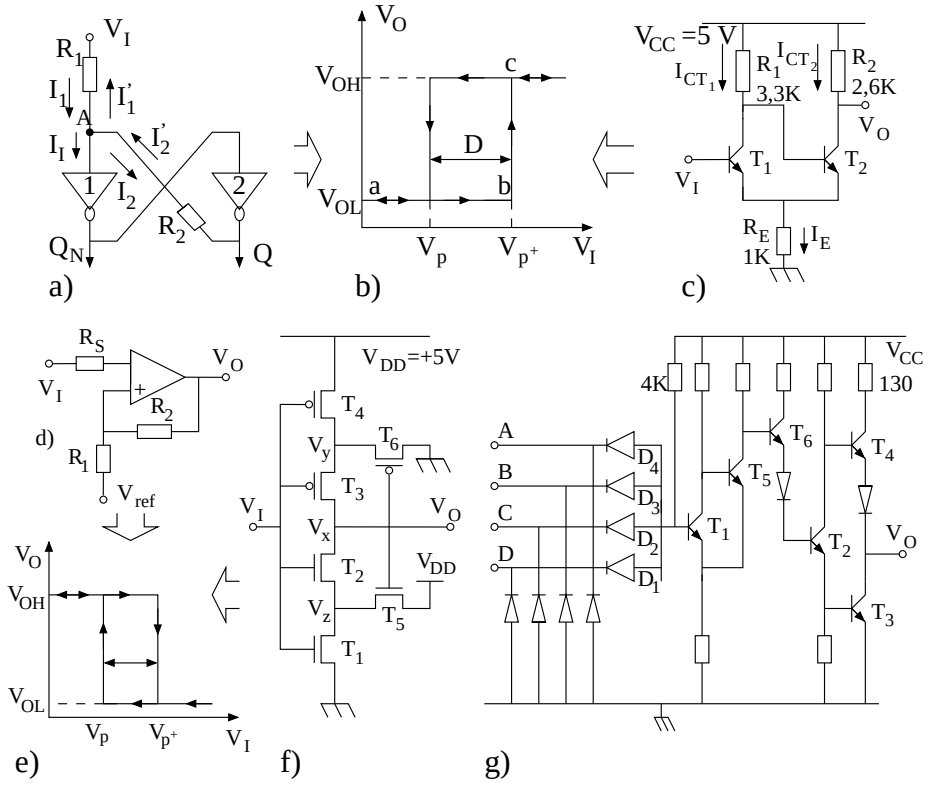


Figura 3.56 Triggerul Schmitt: a,b) structura de principiu şi caracteristica statică de transfer, neinversoare; c) structură realizată cu tranzistoare bipolare; d,e) structura şi caracteristica de transfer pentru un trigger Schmitt realizat cu amplificator operaţional; f) structură în tehnologie CMOS; g) structură de poarta NAND4 trigger Schmitt în tehnologie TTL.

Figura 3.36-a şi b, prezintă în caracteristica statică de transfer, Figura 3.36-c, două stări stabile; circuitul acesta fiind celula elementară pentru toate circuitele(basculante) bistabile. Dar dacă legăturile de conectare între cele două inversoare nu mai sunt simetrice din punct de vedere al conductanţei, una incluzând o rezistenţă R_2, se obţine un circuit, Figura 3.56-a, referit ca trigger Schmitt care prezintă o caracteristică statică de transfer similară cu cea a unui releu cu histerezis, Figura 3.56-b.

Considerând o variaţie a tensiunii de intrare la triggerul Schmitt, V_I, de la valoarea $0V$ până la tensiunea de alimentare şi apoi în sens invers la valoarea $0V$, tensiunea de ieşire V_O (Q) va bascula de la valoarea V_{OL} la V_{OH}, când $V_I = V_{p^+}$, şi de la V_{OH} la V_{OL}, când $V_I = V_{p^-}$; cele două valori(praguri) de basculare, V_{p^+} şi V_{p^-} nefiind egale, acestea fiind depărtate cu lăţimea de histerezis $\Delta = V_{p^+} - V_{p^-}$. Pentru determinarea valorilor caracteristicii de releu cu histerezis se va utiliza noţiunea de prag logic de comutaţie al unei porţi V_T (pragul logic de comutaţie al unei porţi defineşte acel punct pe caracteristica statică de transfer unde tensiunea de intrare este egală cu cea

de ieşire $V_T = V_I = V_O$; în acest punct, practic, curentul de intrare în poarta este $I_I = 0$). Ecuaţia de curenţi în punctul A se reduce la egalitatea $I_1 = I_2$ care este îndeplinită doar când tensiunea de intrare în inversor este V_T. La creşterea tensiunii de intrare V_I când aceasta devine egală cu V_{p^+}, tensiunea în punctul A are valoarea V_T, iar ieşirea basculează de la V_{OL} la V_{OH} se pot scrie relaţiile:

$$I_1 = \frac{V_I - V_A}{R_1} = \frac{V_{P^+} - V_T}{R_1} \qquad I_2 = \frac{V_A - V_O}{R_2} = \frac{V_T - V_{OL}}{R_2}$$

iar din egalitatea curenţilor se obţine expresia

$$V_{p^+} = V_T(1 + \frac{R_1}{R_2}) - V_{OL}\frac{R_1}{R_2} \tag{3.29-a}$$

Similar, la descreşterea tensiunii V_I, când aceasta devine egală cu V_{p^-}, tensiunea în punctul A are valoarea V_T, iar ieşirea basculează de la V_{OH} la V_{OL}, rezultă relaţiile:

$$I_1{}' = \frac{V_A - V_I}{R_1} = \frac{V_T - V_{p^-}}{R_1} \qquad I_2{}' = \frac{V_O - V_A}{R_2} = \frac{V_{OH} - V_T}{R_2}$$

iar din egalitatea curenţilor se obţine expresia

$$V_{p^-} = V_T(1 + \frac{R_1}{R_2}) - V_{OH}\frac{R_1}{R_2} \tag{3.29-b}$$

Expresia lăţimii histerezisului fiind

$$\Delta = V_{p^+} - V_{p^-} = \frac{R_1}{R_2}(V_{OH} - V_{OL}) \tag{3.30}$$

Pentru o poartă inversor 74LS04 cu $V_{OL} = 0,2V$; $V_{OH} = 3,6V$; $V_T = 1,7V$ şi alegând raportul R_1/R_2=0,37 se obţin: $V_{p^-} = 1V$, $V_{p^+} = 2V$ şi $\Delta = 1V$.

O structură de trigger Schmitt cu tranzistoare, care are o caracteristică statică de transfer de aceeaşi formă cu cea din Figura 3.56-b este prezentată în Figura 3.56-c [Toacşe ′96]. Cele două inversoare sunt realizate cu cele două tranzistoare T_1 şi T_2, existând figurată doar o singură legătură de reacţie din colectorul lui T_1 în baza lui T_2. Reacţia de la amplificatorul cu T_2 la amplificatorul cu T_1 apare ca un efect al căderii de tensiune produsă de I_{CT2} pe rezistenţa comună din emitor R_E şi care se aplică la intrarea tranzistorului T_1, $V_I = V_{BE} + I_E R_E$, $I_E = I_{CT1} + T_{CT1}$ (rezistenţa R_E simulează un generator de curent constant în emitoare, deci $I_E \approx$ const.). Când $V_I = 0$, T_1 blocat, T_2 în saturaţie $V_O = V_{OL}$. La creşterea lui $V_I \geq V_{BET_{1\text{on}}} + I_E R_E$, tranzistorul T_1 intră în conducţie şi prin micşorarea tensiunii pe colectorul său comandă, prin legătura de reacţie, înspre blocare tranzistorul T_2, adică micşorarea curentului I_{CT2}. Dar pentru că $I_E \approx$ const., curentul I_{CT1} va creşte şi va provoca o mai pronunţată scădere a tensiunii în colector deci o comandă în continuare de blocare pe baza lui T_2 (iată efectul de reacţie pozitivă prin R_E!). La o valoare a tensiunii de intrare V_I=V_{p^+}, T_2 se blochează, T_1 conduce în saturaţie, tensiunea de ieşire a basculat de la V_{OL} la V_{OH}. Invers, apare ca efect reacţia pozitivă la scăderea tensiunii de intrare, iar când ajunge la $V_I = V_{p^-}$ apare bascularea de la V_{OH} la V_{OL}. Cu valorile

de rezistenţe din figură şi $V_{BE(\text{on})} = 0,7V$, $V_{BE(\text{sat})} = 0,8V$, $V_{CE(\text{sat})} = 0,1V$ se obţin: $V_{p^+} = 2,5V$, $V_{p^-} = 1,6V$, $\Delta = 0,9V$.

O idee practică pentru realizarea unui trigger Schmitt pe baza unui amplificator operaţional este prezentată în Figura 3.56-d, cu o caracteristică inversoare, Figura 3.56-e. Reacţia pozitivă se obţine de pe divizorul R_1, R_2 între V_O şi V_{ref}, iar V_I se aplică pe intrarea inversoare (aplicând reacţia înspre intrarea inversoare iar V_I pe intrarea neinversoare se obţine o caracteristică statică de tip releu neinversoare). De pe divizorul rezistiv R_1R_2 se obţin relaţiile:

$$\begin{aligned} V_{p^+} &= V_{ref} + \frac{R_1}{R_1+R_2}(V_{OH} - V_{ref}) \\ V_{p^-} &= V_{ref} + \frac{R_1}{R_1+R_2}(V_{OL} - V_{ref}) \\ \Delta &= \frac{R_1}{R_1+R_2}(V_{OH} - V_{OL}) \end{aligned} \tag{3.31}$$

Cu aceste relaţii, în caracteristica inversoare să existe aceleaşi valori ca la caracteristica neinversoare prezentată anterior $V_{p^+} = 2V$, $V_{p^-} = 1V$, $\Delta = 1V$, se calculează $V_{ref} = 1,335V$ şi $R_1/R_2 = 0,416$.

O structură de trigger Schmitt în tehnologie CMOS este prezentată în Figura 3.56-f care realizează o caracteristică statică de releu cu histerezis, inversoare ca în Figura 3.56-e. Când $V_I = 0$ tranzistoarele T_3 şi T_4 sunt în conducţie dar conduc un curent neglijabil deoarece T_1 şi T_2 sunt blocate (Se consideră tensiunile de prag ale tranzistoarelor $V_{pn} = +1,0V$, $V_{pp} = -1,0V$), T_6 este blocat, T_5 este în saturaţie, $V_y = V_x \approx 5V$ iar $V_z = V_{DD} - V_{T5} = 3,5V$. Când V_I la $V_{pn} = 1,0V$, T_1 intră în conducţie şi împreună cu T_5 formează un amplificator, existând tendinţa de scădere a tensiunii V_z, dar T_2 este blocat. Crescând $V_I = V_{p^+}$, tensiunea $V_{GS,T_2} \geq V_{pn}$, şi în succesiune: T_2 intră în conducţie, $V_x = V_z = 0V$, T_5 intră în blocare, T_3 intră în blocare atrăgând după sine intrarea în conducţie a lui T_6, tensiunea de ieşire comută de la V_{DD} la 0V. Pornind, cu $V_I = 5V$, înapoi T_4, T_3 sunt blocate iar T_6 este în conducţie. Întâi intră în conducţie T_4 apoi T_3 iar T_6 se blochează, în reţeaua n, T_1, T_2 se blochează şi T_5 va conduce deci s-a realizat la V_{p^-} comutaţia tensiunii V_O de la $0V$ la V_{DD}.

Structura din Figura 3.56-g prezintă o poartă NAND4 trigger Schmitt. Triggerul Schmitt este format cu un amplificator pe tranzistoarele T_1, T_5 iar celălalt amplificator pe T_2, similar ca în Figura 3.56-c, şi este plasat în structura porţii între circuitul SI de intrare, compus din diodele $D_1 - D_4$, şi circuitul defazor T_2 care comandă circuitul de ieşire T_3, T_4.

Circuitul trigger Schmitt, prezentând o caracteristică statică de releu cu histerezis, care comută rapid datorită reacţiei pozitive (basculare) poate fi utilizat ca discriminator de nivel (prin cele două praguri V_{p^+}, V_{p^-}), este recomandat pentru următoarele tipuri de aplicaţii:

1. Obţinerea unei comutaţii rapide a ieşirii pentru variaţii lente pe intrare (formarea semnalelor cu front necorespunzător, discretizarea semnalelor analogice în semnale binare etc.).

2. Reducerea (eliminare) efectelor zgomotului asupra semnalului util (prin prezenţa histerezis-ului).

Exemplul 3.20 Cu o poartă 74HC132 (NAND2 cu trigger Schmitt pe intrare) să se realizeze un oscilator cu frevenţa de 10Hz utilizând un condensator $C = 0,47F$, Figura 3.57-a.

Soluţie. Poarta 74HC132 având o caracteristică de tip releu cu histerezis inversor printr-o conectare a ieşirii la intrare rezultă o structură de oscilator cu perioada de oscilaţie $T = 2\tau_p$, τ_p fiind timpul de propagare prin poartă. Pentru mărirea perioadei oscilaţiilor reacţia este realizată printr-o reţea RC integratoare. Într-o astfel de reţea, variaţia în timp a tensiunii pe condensator $v_C(t)$, pornind de la tensiunea iniţială $v(0)$ iar pe intrare se aplică o treaptă de tensiune $V = v(\infty)$, este exprimată prin relaţia :

$$v_C(t) = v(\infty) - [v(\infty) - v(0)]e^{\frac{-t}{RC}} \tag{3.32}$$

În momentul $t = 0$, când tensiunea pe condensator scade la valoarea de prag V_{p^-}, tensiunea de ieşire, v_O, va comuta de la $V_{OL} = 0V$ la $V_{OH} = V_{DD}$. Acest salt al tensiunii de ieşire aplicat pe intrarea reţelei integrative va determina creşterea tensiunii pe condensator, care în momentul $t = T_1$ atinge valoarea de prag basculare V_{p^+}

$$v_C(T_1) = V_{p^+} = V_{DD} - (V_{DD} - V_{p^-})e^{\frac{-T_1}{RC}} \rightarrow T_1 = RC\frac{V_{DD} - V_{p^-}}{V_{DD} - V_{p^+}}$$

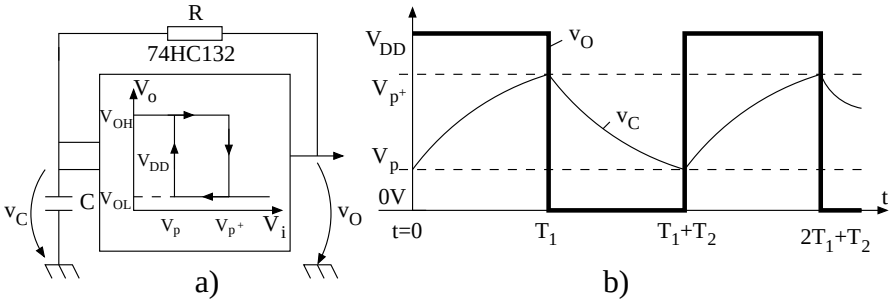


Figura 3.57 Circuit oscilator obţinut pe baza unei porţi (7HC132) inversor trigger Schmitt

Deoarece în momentul T_1 tensiunea de ieşire comută de la $V_{OH} = V_{DD}$ la $V_{OL} = 0V$ tensiunea pe condensator începe să scadă până în momentul $t = T_1 + T_2$ când se atinge valoarea de prag de basculare V_{p^-}

$$v_C(T_1 + T_2) = V_{p^-} = 0 - (0 - V_{p^+})e^{\frac{-T_2}{RC}} \rightarrow T_2 = -RC \cdot ln\frac{V_{p^-}}{V_{p^+}}$$

rezultă perioada oscilaţiei

$$T = T_1 + T_2 = RC \cdot ln\frac{V_{p^+}}{V_{p^-}} \cdot \frac{(V_{DD} - V_{p^-})}{(V_{DD} - V_{p^+})}$$

Cu valorile $V_{p^+}(\text{tipic}) = 2,38V$, $V_{p^-}(\text{tipic}) = 1,68V$, $C = 0,47F$, $T = 0,1s$ se obţine

$$R = \frac{1}{0,47 \cdot 10^{-6} \cdot ln\frac{2,38(4,5-1,67)}{1,67(4,5-2,38)}} = 330k\Omega$$

Pe intervalul de variaţie de temperatură $-40^{\circ}C \div +80^{\circ}C$, $V_{p+}(\max)$ poate creşte până la $3,15V$ iar $V_{p-}\min$ poate scădea la $0,9V$. Cu aceste valori extreme şi $R = 330K\Omega$ şi $C = 0,47\mu F$ introduse în relaţia anterioară se obţine pentru frecvenţa generată valoarea de $2,9Hz$!

3.3.5 Circuitul basculant monostabil

Pornind, de asemenea, de la celula de bază, Figura 3.36-b, se obţine structura de principiu a circuitului basculant monostabil – **monostabilul** – dacă una din legăturile de reacţie rămâne galvanică iar cealaltă se realizează capacitiv, Figura 3.58-a. Aceasta înseamnă că se transmite (prin condensatorul C_x) de la ieşirea porţii 1 la intrarea porţii 2 numai variaţii, în regim static transferul (pentru componenta continuă) este întrerupt. Funcţionarea de principiu a monostabilului este următoarea. Circuitul se păstrează într-o stare stabilă, $Q = 0$ $Q_N = 1$, un timp indefinit, dacă nu se aplică un impuls din exterior. La aplicarea unui impuls, din exterior, monostabilul basculează în starea opusă, $Q = 1$ $Q_N = 0$, care este menţinută de o reacţie pozitivă a celor două legături încrucişate. Dar această reacţie pozitivă există doar în intervalul de timp τ_w, care corespunde duratei regimului tranzitoriu prin condensatorul C_x, după care se întrerupe, iar monostabilul revine în starea $Q = 0$ $Q_N = 1$. Durata regimului tranzitoriu este funcţie de constanta de timp a legăturii de reacţie în care este introdus condensatorul. Modificând această constantă de timp apare, evident, posibilitatea ca acest circuit să poată genera/marca intervalele de timp controlabile (releu de timp).

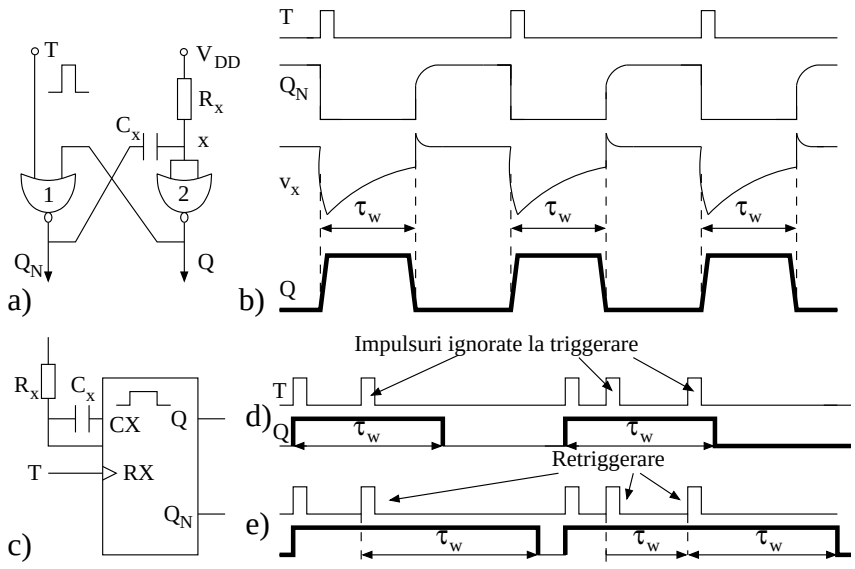


Figura 3.58 Circuitul basculant monostabil (Monostabilul): a,b,c) structură, diagrame de semnal şi simbol de reprezentare; d) diagrama de semnale pentru comandă normală; e) diagrama de semnale pentru comanda cu retriggerare.

Monostabilul din figură este realizat cu porţi NOR CMOS. Reamintim caracteristicile porţilor CMOS: salt de tensiune între 0 şi V_{DD}; curent de intrare neglijabil, prag de basculare $V_T = V_{DD}/2$ (dar dacă diodele de protecţie de pe intrare se polarizează în sens direct funcţionarea porţii se modifică). În starea stabilă a monostabilului ieşirea porţii 1 este în $V_{O1} = V_{DD}$, ieşirea porţii 2 este în $V_{O2} = 0V$, intrarea porţii 2 este $v_x = V_{DD}$, deci tensiunea pe condensator $v_{C_x}(t = 0) = 0V$. La aplicarea unui impuls pozitiv de comandă T pe intrarea porţii 1 ieşirea acesteia comută în $V_{O1} = 0V$,care transmisă prin condensatorul C_x determină tensiunea în punctul x $v_x = 0V$, ieşirea porţii 2 comută în V_{DD}, iar această tensiune de ieşire aplicată pe intrarea porţii 1 face ca să nu mai fie necesară prezenţa impulsului T (reacţia pozitivă s-a realizat). În continuare, printr-un curent de la V_{DD}, prin rezistenţa R_x şi ieşirea porţii 1, condensatorul C_x se încarcă (constanta de timp fiind R_xC_x) iar tensiunea v_{C_x} va creşte de la $v_{C_x}(t = 0) = 0V$ înspre $v_{C_x}(t = 0) = V_{DD}$. Dar, după un interval de timp τ_{w}, tensiunea pe condensator atinge valoarea de prag de comutaţie V_T a porţii 2, ieşirea acestei porţi comută în $V_{02} = 0V$, iar această tensiune aplicată la intrarea porţii 1 comandă ieşirea acesteia în starea $V_{O1} = V_{DD}$. Tensiunea pe condensator devine zero, $V_{O1} - v_x = V_{DD} - V_{DD} = 0$, transferul prin condensator se întrerupe, deci starea $Q_N = 0$, $Q = 1$ s-a terminat; variaţiile de tensiune sunt prezentate în Figura 3.58-b. Un monostabil comandat din exterior cu un impuls negativ se obţine printr-o structură cu porţi NAND. Calculul intervalului τ_{w}, de existenţă a stării instabile, se realizează pornind de la expresia variaţiei tensiunii pe condensator, relaţia(3.32), în care se introduce valoarea de prag de comutaţie V_T şi se obţine

$$V_T = V_{DD} - (V_{DD} - 0)e^{\frac{\tau_{\text{W}}}{R_xC_x}} \rightarrow \tau_{\text{w}} = R_xC_x \cdot ln\frac{V_{DD}}{V_{DD} - V_T}$$

iar pentru $V_T = \frac{V_{DD}}{2}$ rezultă

$$\tau_{\text{w}} = 0,69R_xC_x \tag{3.33}$$

Modul de comandă normală a unui monostabil este reprezentat prin diagrama de semnale din Figura 3.58-d (de exemplu circuitul 74LS121 care poate genera intervale de timp între $30ns$ şi $28s$). Dar, există şi monostabile cu facilitatea de retriggerare (74LS122). **Funcţionarea cu retriggerare**, Figura 3.58-e, realizează printr-un nou impuls T de comandă aplicat pe intrare o nouă declanşare a unei stări $Q_N = 0$ $Q = 1$, deci începerea unui nou interval de timp τ_{w}, chiar dacă anterior a fost comandat un interval τ_{w} şi acesta încă nu s-a consumat.

În general, la circuitele monostabile semnalul de comandă T se obţine din conjuncţia mai multor semnale de intrare, cum este la circuitul 74LS121, unde concură trei intrări $\overline{A_1}$, $\overline{A_2}$ şi B conform relaţiei $T = \overline{\overline{A_1} \cdot \overline{A_2}} \cdot B$. Valoarea $T = 1$ se realizează pentru combinaţiile $B\overline{A_1}\,\overline{A_2} = 100$, 101 şi 110, deci atât pentru fronturi pozitive ale lui B cât şi pentru fronturi negative ale lui $\overline{A_1}$ şi $\overline{A_2}$, rezultând o flexibilitate a comenzii monostabilului atât cu impulsuri pozitive cât şi negative. Se poate realiza, de exemplu, întârzierea cu decalaj τ_{W} variabil a unei succesiuni de impulsuri prin înserierea a două monostabile; primul, dimensionat prin R_xC_x, să producă întârzierea variabilă τ_{W}, iar al doilea să reproducă lăţimea impulsurilor aplicate la primul.

Marcarea intervalelor de timp, precum şi reproductibilitatea lor nu pot avea o precizie foarte ridicată cu ajutorul monostabilului. Aceasta se explică prin faptul că determinarea momentului în timp rezultă în punctul de intersecţie între o

dreaptă orizontală care reprezintă o tensiune de prag prescrisă V_T şi o curbă cu o variaţie exponenţială (variaţia tensiunii de pe un condensator, $v_C(t)$). Mai ales când $t \geq 3T$ ($T = RC$ constanta de timp), tensiunea pe condensator se orizontalizează iar intersecţia între $v_C(t)$ şi V_T devine foarte puţin precisă. Precizia intersecţiei poate fi îmbunătăţită dacă încărcarea/descărcarea condensatorului se face sub curent constant; în acest caz variaţia tensiunii pe condensator are o variaţie liniară $v_C(t) = kt$. Această modalitate de marcare a timpului prin încărcarea/descărcarea unui condensator este specifică în electronica analogică. În electronica digitală intervale de timp se obţin prin contorizarea unui număr prescris de impulsuri dintr-o succesiune de frecvenţă constantă.

3.3.6 Circuitul basculant astabil

La fel, ca la monostabil, şi circuitul basculant astabil (uneori referit ca multivibrator) se obţine din structura celulei de bază, Figura 3.36-b, dar prin realizarea capacitivă a ambelor legături de reacţie, deci se propagă între cele două inversoare numai regimurile tranzitorii, Figura 3.59-a. Transmiţându-se între porţile inversor numai regimurile tranzitorii circuitul este caracterizat numai de două stări instabile. După cum la monostabil durata stării instabile, pe un inversor, este determinată de valoarea constantei de timp a circuitului R_xC_x, la fel şi la astabil durata unei stări instabile este deteminată de constanta de timp a circuitului RC de la intrarea inversorului respectiv. Perioada semnalului dreptunghiular generat este suma duratei celor două stări instabile. În cazul când cele două constante de timp sunt egale, $R_1C_1 = R_2C_2$, factorul de umplere al semnalului generat este de 50%. Pentru analiza comutaţiei alternative între cele două porţi vezi problema P3.50.

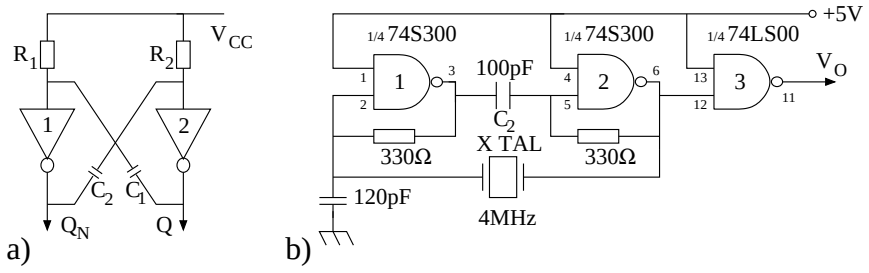


Figura 3.59 Circuitul basculant astabil (Astabilul): a) structură de principiu; b) structură de astabil având frecvenţa ($4MHz$) stabilizată cu cuarţ.

Avantajul simplităţii acestei structuri de astabil, realizat cu porţi logice, este diminuat de lipsa de stabilitate a frecvenţei semnalului generat(cauze: praguri diferite de deschidere pentru porţi, variaţia temperaturii, variaţia tensiunii, îmbătrânirea componentelor). Circuitele astabile care generează o frecvenţă precisă, şi o menţin în timp cu o abatere insignifiantă, se realizează cu cristale de cuarţ. Un cristal de cuarţ, introdus pe una din conexiunile de reacţie, substituind unul din condensatoare, are rolul unui circuit acordat şi care va "pilota" frecvenţa de oscilaţie pe frecvenţa sa proprie de rezonanţă. În Figura 3.59-a este prezentat un astfel de oscilator, utilizând trei porţi NAND ale unui circuit 74LS00, cu un cristal de cuarţ (X-TAL) având frecvenţa

proprie de rezonanţă de $4MHz$. Rezistenţele de 330Ω în paralel cu porţile 1 şi 2 contribuie la o liniarizare a caracteristicilor de transfer ale acestora.

Cristalul de cuarţ (SiO_2) este un piezoelectric, are proprietatea de a polariza sarcini electrice pe suprafeţele sale când este tensionat mecanic (comprimat, torsional, întins). Dar există şi efectul invers de piezoelectricitate – la aplicarea unei diferenţe de potenţial între suprafeţele sale (tăiat sub formă de dreptunghi, pătrat sau cerc cu grosimi în jur de 1mm) se produce o tensionare mecanică. Aplicarea unei diferenţe de potenţial alternative, cu o anumită fercvenţă, va determina vibraţia mecanică a cristalului; modul de vibraţie depinde de forma geometrică sub care este tăiat cristalul. Când este introdus într-un circuit electric cu o tensiune variabilă, în cazul nostru pe o conexiune de reacţie, va vibra cu frecvenţa tensiunii electrice aplicate pe feţele sale.

Din punct de vedere electric cristalul de cuarţ are o schemă echivalentă de circuit RLC în paralel cu o capacitate C_0 (această capacitate fiind cea introdusă de electrozii lipiţi pe suprafeţele sale). Când este comandat cu o frecvenţă mult diferită de frecvenţa de rezonanţă cristal de cuarţ poate fi echivalat cu o simplă capacitate C_0, dar când frecvenţa este cea de rezonanţă circuitul echivalent al cristalului este o capacitate în paralel cu o rezistenţă. Reactanţa cristalului se apropie de zero la punctul de rezonanţă serie, iar într-o conexiune de reacţie a circuitului astabil va pilota frecvenţa acestuia pe frecvenţa de rezonanţă (valori uzuale pentru rezonanţa $1KHz \div 40MHz$, determinată de modul de tăiere, formă şi grosime ale cristalului). Frecvenţele de pilotare a cristalului variază, relativ puţin, cu temperatura şi cu timpul (îmbătrânirea).

Oricare sistem sincron necesită un generator de semnal de ceas şi frecvent se impune ca acest semnal să aibă o abatere cât mai mică a frecvenţei ceea ce se poate realiza cu oscilatoare cu cuarţ. Abaterea de frecvenţă poate fi mai mică decât $2,5 \cdot 10^{-6}$, pentru o variaţie a temperaturii în intervalul $0 \div 50°C$, utilizând un cristal de cuarţ ermetic inclus într-o capsulă, referit oscilator RTXO (Room Temperature Crystal Oscillator). Cu oscilatoarele la care cuarţul este inclus împreună cu circuite de compensare pentru variaţia temperaturii, într-un pachet monolitic, referite ca oscilatoare TXCO (Temperature Compensated Oscillator) se obţin pentru variaţii ale temperaturii în intervalul $0 \div 50°$C abateri ale frecvenţei mai mici decât $5 \cdot 10^{-7}$.

Există variante constructive de astabile care pot funcţiona în regim de sincronizare declanşată sau în regim de comandă. În varianta de **astabil sincronizat**, la fiecare impuls de comandă aplicat din exterior se porneşte astabilul care generează semnale dreptunghiulare cu o fază fixată. Varianta de **astabil comandat** generează semnale dreptunghiulare atât timp cât pe intrarea astabilului se aplică un semnal de comandă exterior de nivel H sau L (în Figura 3.59-b pinul de intrare 13 al porţii 3 se conectează la $+5V$ sau la $0V$).

Exemplul 3.21 Să se modeleze funcţionarea de trigger Schmitt, monostabil şi astabil cu circuitul de temporizare 555.

Soluţie. Circuitul 555 (şi 556) este un circuit integrat în familiile TTL şi CMOS, care prin conectarea în exterior a două sau trei componente poate avea funcţionarea circuitelor trigger Schmitt, astabil şi monostabil. În structura sa, Figura 3.60, conţine: două comparatoare, COMP1 şi COMP2, a căror ieşiri comandă respectiv intrările SR ale unui latch, un tranzistor de descărcare TD, un etaj final de ieşire şi un divizor intern de la tensiunea V_{CC}, $V_{CC} = 4,5 \div 16V$, prin care sunt fixate valorile de referinţă V_{r_1} şi V_{r_2} ale comparatoarelor. Rezistenţele divizorului fiind egale rezultă $V_{r_2} = V_{CC}/3$ şi $V_{r_1} = 2V_{CC}/3$.

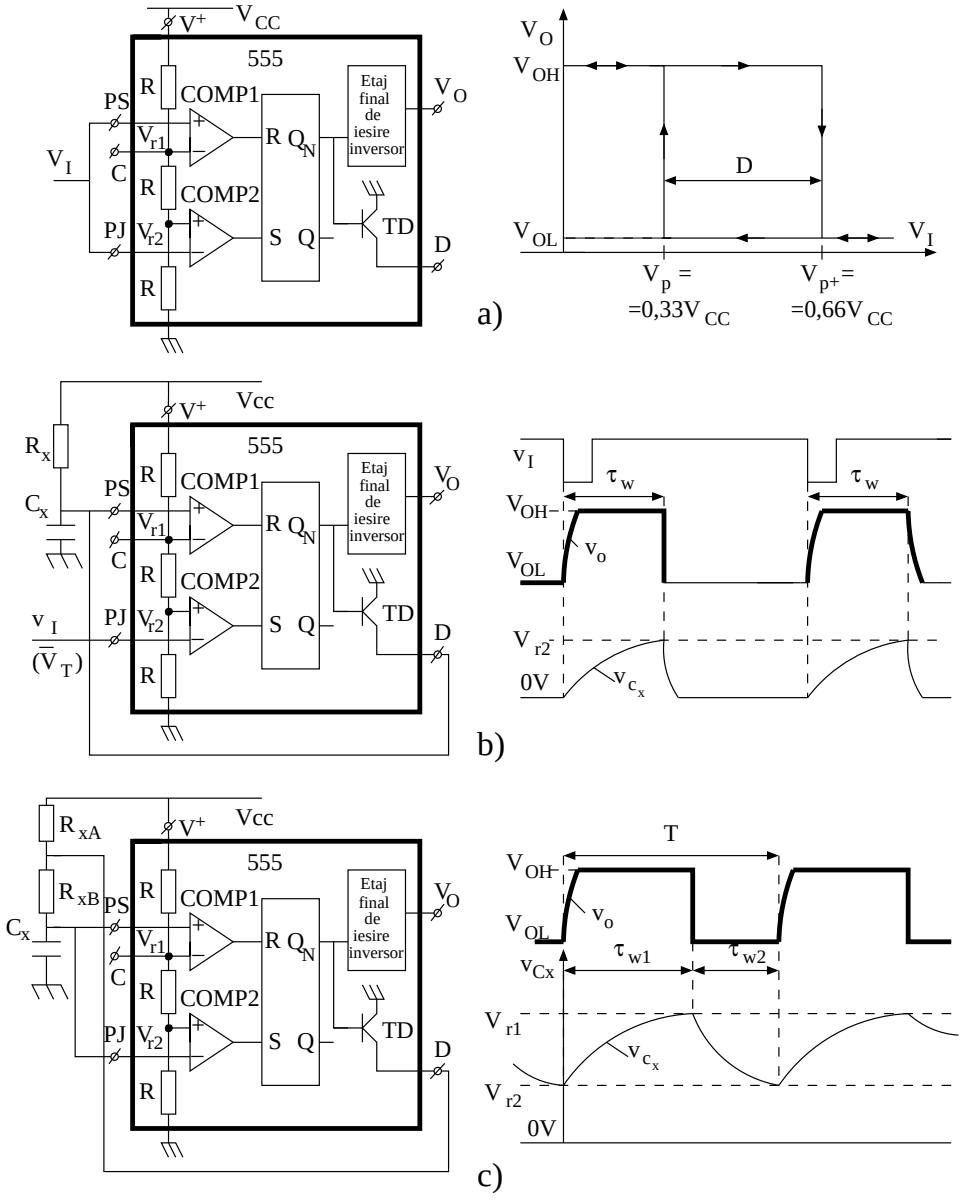


Figura 3.60 Circuitul 555: a) modelarea funcţionării de **trigger Schmitt**; b) modelarea funcţionării de **monostabil**; c) modelarea funcţionării de **astabil**.

1. Modelarea funcţionării de trigger Schmitt, Figura 3.60-a. Tensiunea de intrare V_I se aplică pe cele două intrări (conectate împreună), PS- pragul de sus şi PI- pragul de jos. Pentru $V_I = 0$, intrările $SR = 10$, $Q_N = 0$ iar ieşirea $V_O = V_{OH}$. La creştere, când tensiunea V_I ajunge la valoarea $V_{r_2} = 0,66V_{CC} = V_{p^+}$, $SR = 01$, $Q_N = 1$ iar V_O comută de la V_{OH} la V_{OL}. La scădere, când tensiunea V_I ajunge la $V_{r_2} = 0,33V_{CC} = V_{p^-}$, $SR = 10$, $Q_N = 0$ iar V_O comută de la V_{OL} la V_{OH}. Valoarea pragurilor V_{p^+} şi V_{p^-} pot fi modificate printr-o rezistenţă externă R_x conectată între V_{CC} şi terminalul C.

2. Modelarea funcţionării de monostabil, Figura 3.60-b. În starea stabilă $v_I = V_{OH}$, $SR = 00$, $Q_N = 1$, $v_O = V_{OL}$, tranzistorul TD conduce iar tensiunea v_{C_x}, pe condensatorul exterior, este zero. La aplicarea impulsului de comandă $\overline{V}_T = 0$, $SR = 10$, $Q_N = 0$, $V_O = V_{OH}$ iar TD se blochează permiţând ca v_{C_x} să crească înspre V_{CC}. Când v_{C_x}, după intervalul de timp $\tau_{\rm w}$, atinge valoarea $V_{r_2} = 0,66V_{CC}$, $SR = 01$, $Q_N = 1$, v_O comută de la V_{OH} la V_{OL}, TD intră în conducţie iar C_x este scurtcircuitat la masă.

Valoarea lui $\tau_{\rm w}$ se obţine prin aplicarea relaţiei 3.32 pentru regimul de încărcare al condensatorului C_x

$$V_{r2} = v_{C_x}(\tau_{\rm w}) = V_{CC} - [V_{CC} - 0]e^{\frac{-\tau_{\rm w}}{R_x C_x}}$$
$$\tau_{\rm w} = R_x C_x \cdot ln\frac{V_{CC}}{V_{CC} - \frac{2}{3}V_{CC}} = 1,1R_x C_x \tag{3.34}$$

3. Modelarea funcţionării de astabil, Figura 3.60-c. Se consideră $t = 0$ când $SR = 10$, $Q_N = 0$, $V_O = V_{OH}$, TD se blochează permiţând condensatorului C_x să se încarce prin rezistenţele $R_{XA} + R_{XB}$ înspre tensiunea V_{CC}. Când v_{C_x}, după intervalul de timp $\tau_{\rm w_1}$, atinge valoarea $V_{r_1} = 0,66V_{CC}$, $SR = 01$, $Q_N = 1$, v_O va comuta de la V_{OH} la V_{OL}, TD intră în conducţie iar C_x este descărcat la masă prin rezistenţa R_{XB}. Procesul de descărcare, cu constanta de timp $C_x R_{XB}$, continuă pe durata $\tau_{\rm w_2}$ iar v_{C_x} atinge valoarea $V_{r_1} = 0,33V_{CC}$. În acel moment $SR = 10$, $Q_N = 0$, v_O va comuta de la V_{OL} la V_{OH}, TD se blochează reconfigurând circuitul $R_{XA} + R_{XB}$ de reîncărcare al condensatorului C_x; în continuare cele două regimuri de încărcare/descărcare se succed generându-se un semnal dreptunghiular cu perioada $T = \tau_{\rm w_1} + \tau_{\rm w_2}$. Valorile pentru $\tau_{\rm w_1}$ şi $\tau_{\rm w_2}$ se obţin aplicând relaţia 3.32.

$$\tau_{w_1} = C_x(R_{XA} + R_{XB}) \cdot ln\tfrac{2}{3}(\tfrac{V_{CC}}{V_{CC} - \frac{2}{3}V_{CC}}) = 0,69C_x(R_{XA} + R_{XB})$$
$$\tau_{w_2} = C_x R_{XA} \cdot ln\tfrac{2}{3}\tfrac{V_{CC}}{V_{CC}/3} = 0,69C_x R_{XA}$$
$$T = \tau_{w_1} + \tau_{w_2} = 0,69(R_{XA} + 2R_{XB})C_x$$

iar coeficientul de umplere D are expresia

$$D = \frac{\tau_{w_1}}{\tau_{w_1} + \tau_{w_2}} = 1 - \frac{R_{XB}}{R_{XA} + 2R_{XB}}$$

3.4 CIRCUITE NUMĂRĂTOR

Împarţirea unui număr natural N la numărul natural C este definită de relaţia

$$N = p \cdot C + r, \qquad 0 \le r \le C - 1 \tag{3.35}$$

Deoarece în urma împărţirii pot rezulta doar un număr C de resturi distincte se poate afirma că oricare număr natural N poate fi inclus doar într-o clasă de resturi $\dot{r}$modulo$C$, adică

$$N \equiv \dot{r}\text{modulo } C, \qquad \text{cu } \dot{r} = \left\{\hat{0}, \hat{1}, \hat{1}, \ldots, \widehat{C-2}, \widehat{C-1}\right\} \tag{3.36}$$

unde clasele de resturi sunt:

$$\begin{aligned}
\hat{0} &= \{0, C, 2C, \ldots, jC, \ldots, \mid j \in N\} \\
\hat{1} &= \{1, C+1, 2C+1, \ldots, jC+1, \ldots, \mid j \in N\} \\
\hat{2} &= \{2, C+2, 2C+2, \ldots, jC+2, \ldots, \mid j \in N\} \\
&\vdots \\
\widehat{C-1} &= \{C-1, 2C-1, 3C-1, \ldots, jC-1, \ldots, \mid j \in N\}
\end{aligned}$$

Aflarea numărului de elemente ale unei mulţimi (cardinalul nulţimii) se poate realiza cu ajutorul circuitelor numărător modulo C (se obţine numărul exprimat în bază C).

Un circuit numărător modulo C este un automat a cărui funcţionare este descrisă de un graf de tranziţii, ciclice, care conţin un număr C de stări distincte; din fiecare stare există doar o singură tranziţie necondiţionată spre starea următoare, deci în starea q_i se va reveni după ce se vor parcurge toate celelalte $C-1$ stări, Figura 3.61. Tranziţia fiind necondiţionată se realizează numai la aplicarea impulsului de ceas. De fapt, circuitul numărător/(numărătorul) determină numărul de impulsuri de ceas care au fost aplicate (sau numărul de elemente ale unei mulţimi, fiecărui element îi corespunde un impuls de ceas).

Se poate face o mapare unu-la-unu între mulţimea claselor de resturi $\dot{r}$ şi mulţimea stărilor Q; uzual, clasele de resturi în ordine crescătoare se mapează pe stările automatului în ordinea de parcurgere a acestora (pornind de la una q_0, considerată ca iniţială, în felul următor: $\hat{0} \rightarrow q_0$, $\hat{1} \rightarrow q_1$, ... ,$\widehat{C-1} \rightarrow q_{C-1}$). Realizând această mapare numărătorul modulo C poate fi privit ca un identificator al claselor de resturi modulo C. De exemplu, pentru un numărător modulo 10 toate numerele următoare (de impulsuri de ceas aplicate): 3, 13, 23, 33, ... , 103, ... ,1003, ... , $j \cdot 10 + 3$, ... , care aparţin clasei de resturi 3modulo10 ($\hat{3}$), aduc automatul în starea q_3. Iar, de exemplu, pentru un numărător modulo 7 toate numerele următoare: 2, 9, 16, ... , 51, ... , $j \cdot 7 + 2$, ... care aparţin clasei de resturi 2 modulo 7, aduc automatul în starea q_2. Revenind tot dupa C impulsuri de ceas în aceeaşi stare circuitul numără în baza C.

Dar pentru această numărare în baza C se poate obţine o anumită codificare prin modul de asignare al bitilor z_{k-1}, ... , z_1, z_0 ai fiecărei stări q_j, din cele C. Numărul minim k al biţilor de cod, egal cu numărul celulelor de memorare (bistabile) ale automatului, conform relaţiei 3.17, este $\lceil log_2 C \rceil$. Pentru exemplul anterior, de numărător modulo 10, se poate adapta o numărare in bază 10 sub codul BCD (8-4-2-1) sau în cod Gray sau un cod Excess 3 etc. Deci circuitul următor poate asigura numărarea într-o anumită baza C si sub o anumită codificare.

Circuitul numărător poate fi implementat cu un automat de tip Moore imediat la care ieşirea este identică cu starea, relaţia 3.15, ($Q \equiv Y$); codul stării este numărul exprimat într-o anumită codificare. Foarte frecvent, automatul numărător mai este

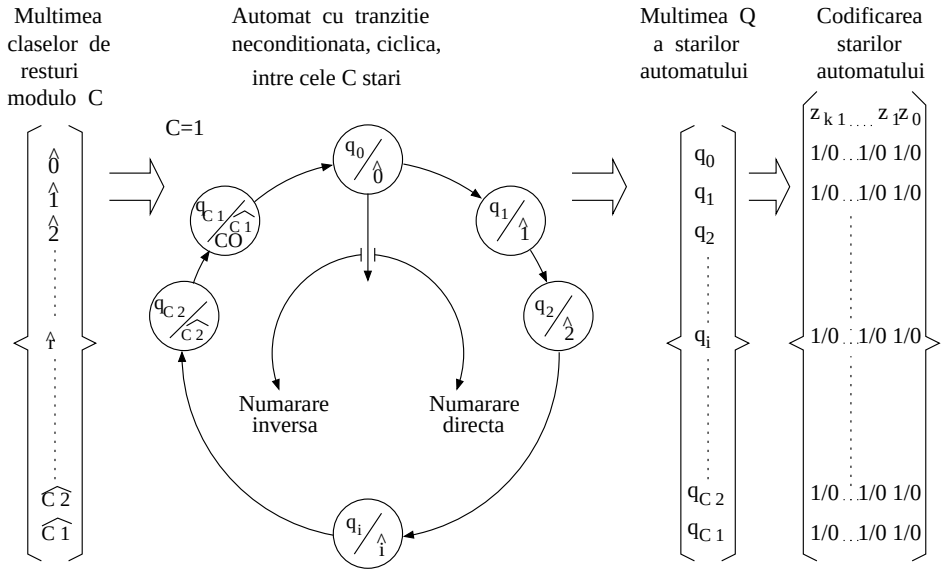


Figura 3.61 Circuitul numărător: Numărătorul modulo C poate fi privit ca un automat de identificare a claselor de resturi modulo C iar codificarea numărării rezultă prin modul de asigurare a celorlalte C stări.

înzestrat cu încă un bit de ieşire, $\mathbf{C}O = \{0, 1\}$, care este semnalul de depăşire a capacităţii de numărare (transportul următor) ce devine activ doar când se ajunge în ultima stare q_{C-1} după care, prin tranziţia în starea iniţiala q_0, se reîncepe ciclul tranziţiilor, deci relaţia de transfer a automatului este $f : Q \to Q \times CO$. De asemenea, se introduce şi un semnal de intrare $\mathbf{C}I = \{0, 1\}$ **- transportul anterior** deci relaţia de tranziţie a stărilor este $g : Q \times CI \to Q$. Semnalul CI poate fi interpretat şi ca un semnal de validare/enable (EN) care pentru valoarea 1 determină trecerea numărătorului în starea următoare, q_{i+1}, la aplicarea fiecărui impuls de ceas, iar pentru valoarea 0 rămânerea în aceeaşi stare, q_i.

Înzestrat cu cele două semnale, CI– transportul anterior şi CO– transportul următor, numărătorul ne aminteşte de circuitul sumator. Evident, se poate substitui circuitul numărător cu un sumator, la care pe o intrare se aplica permanent cuvântul 1 iar pe cealaltă intrare suma obţinută anterior, deci la aplicarea fiecărui impuls de ceas se adună unu la numărul anterior (**Orice număr se obtine din numărul anterior plus 1**). Cele două semnale introduc facilitatea de a obţine un circuit numărător modul C (care nu este număr prim) prin înserierea de circuite numărător a căror modulo sunt divizori ai lui C conform relaţiei:

$$C = C_1 C_2 \dots C_K \tag{3.37}$$

Graful ciclic de tranziţie al stărilor poate fi parcurs în unul din cele două sensuri rezultând, în consecinţă, pentru circuitul numărător o funcţionare de **numărare directă** sau **numărare inversă**; sau dacă se introduce pentru circuitul numărător o intrare I, pentru care valoarea $I = 0$ determină numărarea directă iar pentru $I = 1$

determină numărarea inversă, se obţine un **numărător reversibil**.

Existenţa la un circuit numărător modulo C, pe ieşire, pe lângă cuvântul de cod al stării prezente - pentru numărare - şi a semnalului CO - pentru depăşirea capacitaţii - determină două moduri de funcţionare/(utilizare) distincte. Prima funcţionare, cea de **numărare într-o bază modulo** C sub un anumit cod, când este considerată succesiunea cuvintelor de cod a stărilor parcurse. Cea de-a doua funcţionare, când se consideră doar semnalul de depăşire CO produs după C impulsuri de ceas aplicate, este cea de **divizare cu** C.

Din punct de vedere al implementării unui circuit numărător, în funcţie de aplicarea impulsului de ceas care determină trecerea din starea q_i in starea q_{i+1}, acesta poate fi de tip asincron sau de tip sincron.

3.4.1 Numărătoare asincrone

Pentru numărătorul care va fi analizat în această secţiune, denumirea de asincron reflectă faptul că în funcţionarea sa celulele de memorare (bistabilele) nu comută toate sincron cu semnalul de ceas, aceasta datorită structurării numărătorului ca o înseriere de bistabile de tip T. În secţiunea 3.3.2.4 s-a arătat ca bistabilul T în regim de basculă ($T = 1$) comută la fiecare impuls de ceas, rezultând un semnal de ieşire Q cu perioada dublă fată de cea a semnalului de ceas, Figura 3.49-e deci o divizare cu doi (2^1). Dacă semnalul de ieşire Q se aplică unui următor bistabil T (ca semnal de ceas), tot în regim de basculă, $T = 1$, după aceasta se obţine o divizare cu 4 (2^2), iar dacă se continuă cu un al treilea bistabil T înseriat se obţine un semnal divizat cu 8 (2^3) în raport cu semnalul înităial de ceas. Generalizând, prin înserierea a n bistabile T se obţine o divizare cu 2^n, deci un numărător modulo 2^n; o astfel de structurare de numărător modulo 2^3 este prezentată in Figura 3.62-b. Dar care din ieşirile Q sau Q_N ale celulei bistabil de rang i se aplică pe intrarea de ceas a celulei T de rang $i+1$? Răspunsul rezultă simplu din analiza succesiunii de numărare, în cod binar natural, din Figura 3.62-a.

Bitul de rang $2^0(z_0)$ din cuvântul de cod binar natural al numărului schimbă în valoarea opusă la fiecare impuls de ceas. Pentru numărarea în sens direct, bitul de rang 2^1 (z_1) schimbă în starea opusă numai când bitul z_0 schimbă din 1 în 0; de asemenea bitul de rang 2^2 (z_2) schimbă în starea opusă numai când bitul z_1 comută din 1 în 0, în tabelul succesiunii de numărare aceste comutaţii sunt indicate prin săgeţi. În concluzie, celula de bistabil T de rang i este comutată în starea opusă de către celula de rang $i-1$ numai când aceasta comută de 1 la 0. Deoarece în structura aleasă de numărător s-au considerat celulele bistabile T comandate pe frontul negativ al semnalului de ceas, rezultă că pentru intrarea CLK a celulei de rang i se aplica ieşirea Q a celulei de rang i-1 care produce front negativ la comutaţia lui z_{i-1} de la 1 la 0. Dacă celulele bistabile erau comandate pe frontul pozitiv de ceas atunci pe intrarea CLK a celulei de rang i s-ar fi aplicat semnalul Q_N de la celula de rang $i-1$, deoarece numai acest semnal produce un front pozitiv când această celulă comută de la 1 la 0.

Diagrama de semnale pentru acest numărător modulo 8, când parcurge toate cele 8 stări, este prezentată în Figura 3.62-c. Pornind din starea $z_2z_1z_0 = 000$ la aplicarea primului impuls de ceas, pe frontul negativ, în numărător se înscrie starea 001, după al doilea impuls se înscrie 010 şi prima celulă revine în $z_0 = 1$, dar frontul negativ

de la ieşirea acesteia comandă celula a două în $z_1 = 1$. La al patrulea front negativ există un transfer de la celula 2^0 la celula de rang 2^1, care comută pe z_1 în 0, şi la fel există un transfer de la celula 2^1 la celula 2^2 care comută pe z_2 în 1, deci numărul este 100. După al şaptelea impuls numărul înscris este 111, s-a atins capacitatea maximă, iar la al optulea impuls de ceas toate fronturile pe intrările de ceas sunt negative, la fiecare celulă se generează un transfer către celula următoare, cel de la celula de rang 2^2 nu este utilizat pentru o celulă următoare, numărătorul revine în starea iniţială $\hat{0} = 8$ modulo 8.

În diagrama de semnale prezentată s-a considerat că timpul de propagare, τ_{pCQ} ($\tau_{\text{pHL(CQ)}} = \tau_{\text{pLH(CQ)}} = \tau_{\text{pCQ}}$) pe un bistabil T, de la intrarea de ceas la ieşire, are valoarea de zero. În Figura 3.62-d s-a redesenat diagrama de semnale, cu comutaţiile întârziate ale celulelor datorate propagărilor τ_{pCQ} pentru primele patru semnale de ceas. Se observă că la al doilea impuls de ceas comutaţia ieşirii z_1 apare cu o întârziere de $2\tau_{\text{pCQ}}$, iar la al patrulea impuls de ceas comutaţia ieşirii z_2 apare o întârziere de $3\tau_{\text{pCQ}}$; deci celulele vor comuta asincron cu semnalul de ceas, de unde şi denumirea de numărător asincron. Evident, la un numărător asincron cu n celule întârziere maximă datorată propagării prin cele n celule este $n \times \tau_{\text{pCQ}}$. Rezultă că valoarea frecvenţei de ceas nu poate creşte peste $1/(n\tau_{\text{pCQ}})$. Avantajului simplităţii structurii numărătorului asincron i se opune limitarea frecvenţei de comandă a ceasului, $f_{CLK} \leq 1/(n\tau_{\text{pCQ}})$. Pentru numărătorul asincron modulo 2^n dimensiunea S(n) şi adâncimea D(n) sunt în O(n). Asincronismul în comutaţia celulelor generează stări false la tranzitia unora dintre două stări consecutive. De exmplu, la tranziţia din starea $1|_{10} = 001$ în starea $2|_{10} = 010$ se generează pe durata τ_{pCQ} starea falsă 000, iar la tranziţia din starea $3|_{10} = 011$ în starea $4|_{10} = 100$ se generează următoarele două stări false: 010 si 000. Utilizarea ieşirilor numărătorului z_2, z_1, z_0 ca intrări la un circuit combinaţional generează hazard, utilizarea lor este posibilă numai după consumarea regimului tranzitoriu($> n\tau_{\text{pCQ}}$). O soluţie de pentru evitarea hazardului constă în strobarea aplicării ieşirilor numărătorului, adică circuitul care are ca intrări cuvântul $z_2z_1z_0$ este validat cu un semnal (de strob) întârziat faţă de frontul negativ al semnalului de ceas de la numărător (cum ar fi, de exmplu, palierul pozitiv al semnalului ceas).

Structura de numărător asincron cu numărare directă poate fi transformată uşor într-o structură de numărător cu numărare inversă prin aplicarea pe intrarea de ceas a celulei de rang i a semnalului corespunzător (Q sau Q_N) de la ieşirea celulei de rang $i-1$. Din tabelul succesiunii de numărare inversă, din Figura 3.62-a, rezultă că celula de rang 2^0 comută la fiecare impuls de ceas, celula de rang 2^1 va comuta numai când celula 2^0 comută din 0 în 1, iar celula de rang 2^2 comuta numai când celula 2^1 comută din 0 în 1, adică tocmai invers ca la numărarea directă. Deci, pentru celulele cu comutare pe front pozitiv, se culege semnalul de pe iesirea Q a celulei anterioare, iar pentru comutaţia pe front negativ se culege de pe ieşirea Q_N a celulei anterioare, Figura 3.62-e. O structură de numărător comandat pentru numărarea directă sau numărare inversă se obţine prin introducerea unor selectoare între două celule de numărare succesive care va alege ieşirea potrivită de la celula anterioară şi o aplică pe intrarea de ceas a celulei următoare. Un astfel de selector este fie un MUX2:1, fie o poartă XOR.

Un numărător asincron modulo 2^n poate fi modificat pentru o funcţionare ca numărător modulo C, unde $C < 2^n$. O astfel de modificare se obţine prin eliminarea unui

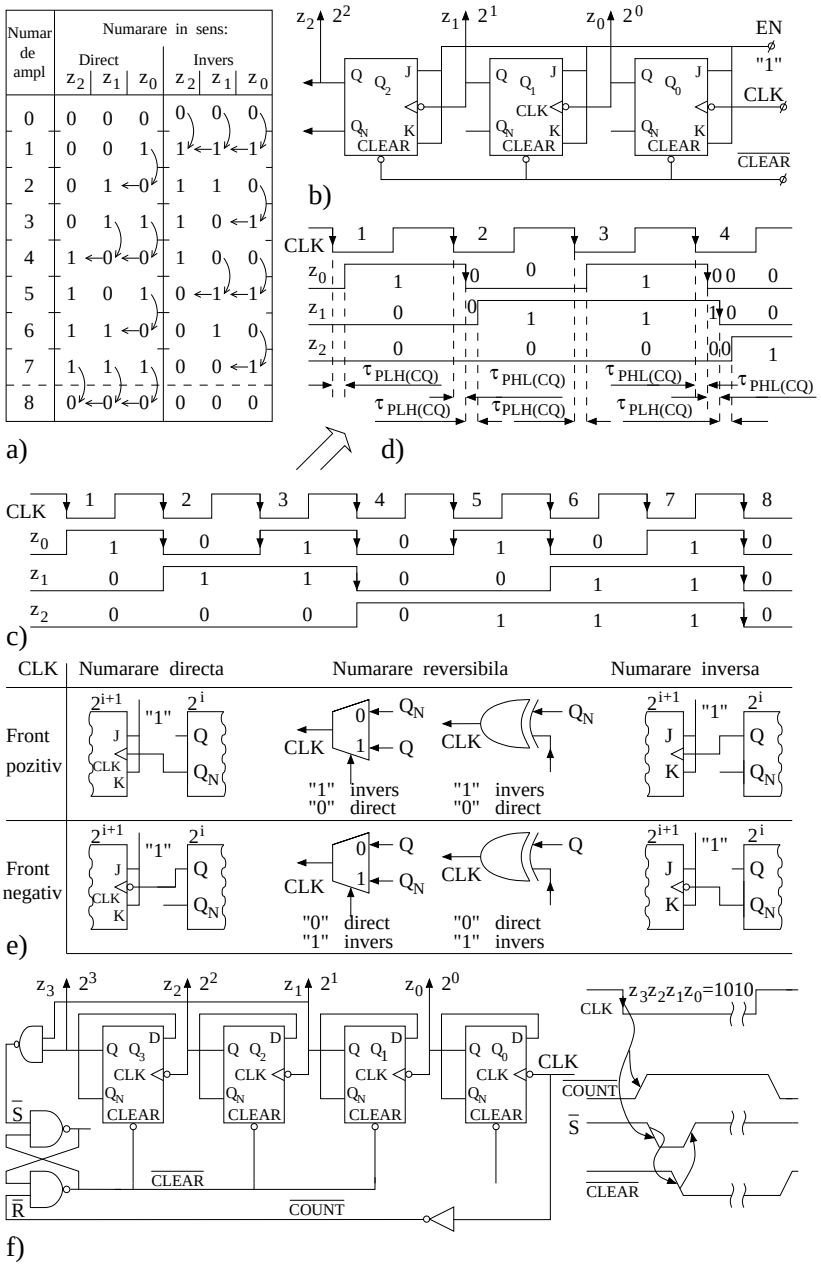


Figura 3.62 Numărătorul asincron: a) tabelul succesiunii de numărare directă şi inversă, modulo 8; b,c) structura şi diagrama de semnale pentru un numărător modulo 8; d) evidenţierea timpilor de propagare şi a stărilor false introduse; e) conexiunile pentru obţinerea semnalului de ceas la o celulă de rang $i+1$; f) numărător modulo 10 obţinut prin modificarea unui numărător modulo 2^4.

număr($2^n - C$) stări din graful de tranziţie al stărilor. Un decodificator identifică în cuvântul de cod al numărătorului când se realizează starea C şi atunci se forţează, prin intrările asincrone de stergere CLEAR a celulelor, înscrierea în starea iniţială 000...00. În Figura 3.62-f se exemplifică modificarea unui numărător modulo 16, realizat cu bistabilele D care modelează o funcţionare de bistabili T, într-un numărător modulo 10, $z_3z_2z_1z_0 = 1010$; circuitul decodificator este o poartă NAND2 care identifică starea 1010, $z_3 = z_1 = 1$, se generează un semnal de resetare, activ în zero, care forţează înscrierea în 0 a celulelor bistabile Q_3, Q_2, Q_1 (se şterge şi Q_2 deoarece stergerea lui Q_1 produce la ieşirea Q o comutaţie de la 1 la 0 care ar înscrie pe Q_2 în 1).

Bucla de forţare a stării 000 este transparentă, adică oricare înscriere în zero a unuia din bitii z_3 sau z_1, aplicaţi la intrarea porţii NAND2, ar produce la ieşirea acesteia o anulare a semnalului de resetare, $\overline{CLEAR}$ devine 1. Scurtarea duratei de activare a semnalului de resetare, $\overline{CLEAR} = 0$, duce la neînscrierea în starea 0 a unora din cele trei bistabile (Q_3, Q_2, Q_1). Presupunem: $\tau_{\text{pNAN2}} = 9nS, \tau_{\text{pRESET(tipic)}} = 25nS$ pentru bistabilul Q_3 si $\tau_{\text{pRESET(maxim)}} = 40ns$ pentru bistabilul Q_1. Cu aceste date din momentul activării semnalului $\overline{CLEAR} = 0$, după o întârziere $\tau = 9ns + 25ns = 34ns < 40ns$, datorită faptului că z_3 a fost înscris în zero, poarta NAND2 anulează semnalul de resetare ($\overline{CLEAR}$=1) înainte ca şi Q_1 să fi fost înscris în 0, deci cuvântul forţat în numărător nu este 0000 ci 0010. Transparenţa buclei este eliminată prin introducerea unui latch $\overline{S}$ $\overline{R}$ şi prin aceasta (chiar dacă unul din semnalele de ieşire z_1 sau z_3 devine mai devreme zero) se lungeşte durata de activare a semnalului de resetare ($\overline{CLEAR}$=0) până când la intrarea latch-ului apare $\overline{R}$=0 (adica în momentul apariţiei următorului front pozitiv de ceas CLK când semnalul $\overline{COUNT} = \overline{R}$ devine activ, $\overline{COUNT} = \overline{CLK}$). Dar şi cu eliminarea transparenţei buclei de forţare acesta structură de numărător modulo 10, obţinut din unul modulo 16, mai prezintă totuşi o funcţionare defectuoasă datorită forţării înscrierii unei stări (prin utilizarea intrărilor asincrone CLEAR).

La al zecelea impuls de ceas când numărătorul a ajuns în starea $z_3z_2z_1z_0 = 1010$, si prin detectarea acestei stări, se forţează înscrierea stării $z_3z_2z_1z_0 = 0000$. La următorul impuls de ceas are loc tranziţia în $z_3z_2z_1z_0 = 0001$ după care pe următoarele nouă impulsuri se ajunge nou în $z_3z_2z_1z_0 = 1010$. Rezultă că, pe durata celui de al zecelea impuls de tact, numărătorul a avut pentru scurt timp starea $z_3z_2z_1z_0 = 1010$ şi apoi, până la următorul impuls de ceas, starea $z_3z_2z_1z_0 = 0000$, deci în total 11 stări pentru 10 tacte. Aceasta înseamnă că numărătorul modulo 10 obţinut poate fi utilizat pentru funcţia de numărare numai dacă starea detectată de decodificator (1010), de scurtă durată, nu influenţează succesiunea codurilor de ieşire. În schimb acest numărător modificat poate fi utilizat ca divizor modulo 10 dacă în starea 1010 se generează un impuls de depăşire CO (obţinut printr-o poartă AND2 cu intrările z_3 si z_1).

Trebuie evidenţiat faptul că la un numărător asincron momentul de comutare precum şi modul cum comută sunt determinate de frontul impulsului de ceas deci nu este decuplată acţiunea "cum" de acţiunea "când", situaţie întâlnită la latch-uri. Pentru "cum" decizia este dată de comutaţia bistabilului anterior şi nu de starea bistabililor anteriori. Pentru aplicaţii de frecventă joasă, când stările false introduse nu sunt un dezavantaj (sau pot fi uşor eliminate prin strobare), datorită simplităţii sale circuitele numărător asincron pot fi utilizate.

Numărătoare asincrone, comercial obtenabile, sub forma de circuite integrate

(74X) sunt:
1 - binare pe patru celule : 74XX/69/93/177/197/293/393
2 - decadice :74XX/68/90/176/196/290/390/490

3.4.2 Numărătoare sincrone

La un numărător asincron toate celulele bistabil comută simultan cu frontul activ al semnalului de ceas. Pentru sinteza numărătorului modulo 2^n asincron, în cod binar natural (8-4-2-1), la care doar prima celulă comută sincron cu ceasul iar fiecare altă celulă primeşte semnalul pe intrarea de ceas tot la a doua comutaţie dar de la celula anterioară, s-a utilizat o înseriere de celule bistabile T; acest bistabil având particularitatea în funcţionare că este un numărător modulo 2^1 ($\hat{0}$=2 modulo 2). În schimb, sinteza unui numărător modulo C sincron, în orice cod şi nu numai în numai în cod binar, trebuie făcută ca a unui automat pornind de la graful/tabelul de tranziţie al stărilor. Sinteza ca automat a numărătorului sincron, pentru început, se va face considerând un numărător modulo 2^n în cod binar natural (pentru că din acesta se poate obţine usor un alt numărător modulo C, C$< 2^n$).

Funcţiile de excitaţie w_i ale automatului numărător, notate în acest caz cu T_i pentru că implementarea se va face cu bistabile T, se deduc din succesiunea stărilor, asignate în cod binar natural, prezentate în Figura 3.62-a (pentru modulo 8). Expresiile funcţiilor T_i se deduc foarte simplu pe baza observaţiei: fiecare bit $z_i, i \neq 0$, din cuvântul stării următoare, va comuta faţă de valoarea din starea prezentă numai când în starea prezentă toţi biţii anteriori lui z_i au valoarea 1; bitul z_0 comută la fiecare impuls de ceas. Deci celula de rang i va comuta la aplicarea impulsului de ceas dacă funcţia sa de excitaţie T_i are valoarea 1, valoare care se calculează în funcţie de starea tuturor celulelor anterioare. Introducând şi semnalul de transport anterior CI, considerat ca o intrare de validare EN, se obţin expresiile pentru funcţiile de excitaţie.

$$\begin{array}{ll|ll}
T_0 = EN & & T_0 = EN & \\
T_1 = EN \cdot z_0 & & T_1 = T_0 \cdot z_0 & \\
T_2 = EN \cdot z_0 \cdot z_1 & & T_2 = T_1 \cdot z_1 & \\
T_3 = EN \cdot z_0 \cdot z_1 \cdot z_2 & (a) & T_3 = T_2 \cdot z_2 & (b) \\
\vdots & & \vdots & \\
T_{n-2} = EN \cdot z_0 \cdot z_1 \cdot z_2 \ldots z_{n-4} \cdot z_{n-3} & & T_{n-2} = T_{n-3} \cdot z_{n-3} & \\
T_{n-1} = EN \cdot z_0 \cdot z_1 \cdot z_2 \ldots z_{n-4} \cdot z_{n-3} \cdot z_{n-2} & & T_{n-1} = T_{n-2} \cdot z_{n-2} &
\end{array} \tag{3.38}$$

Dacă se calculează şi funcţia de excitaţie T_n care este de fapt transferul CO (depăşirea de capacitate când numărătorul este plin cu 111....11) se adaugă şi expresia

$$\begin{array}{ll}
CO = T_n = EN \cdot z_0 \cdot z_1 \cdot z_2 \cdot \ldots \cdot z_{n-3} \cdot z_{n-2} \cdot z_{n-1} & (a) \\
CO = T_n = T_{n-1} \cdot z_{n-1} & (b)
\end{array} \tag{3.39}$$

Expresiile anterioare, obţinute recursiv, care calculează prefixe (expresiile cu indicele mai mic decât i pot fi considerate prefixe în procesul de calcul pentru expresia cu indicele i) pot fi implementate fie cu n porţi AND având de la 2 până la $n+1$ intrări (relaţia 3.38-a) fie cu n porţi AND2 (relaţia 3.38-b). Aceste două implementări (extreme) ale circuitului combinaţional pentru automatul numărător corespund celor

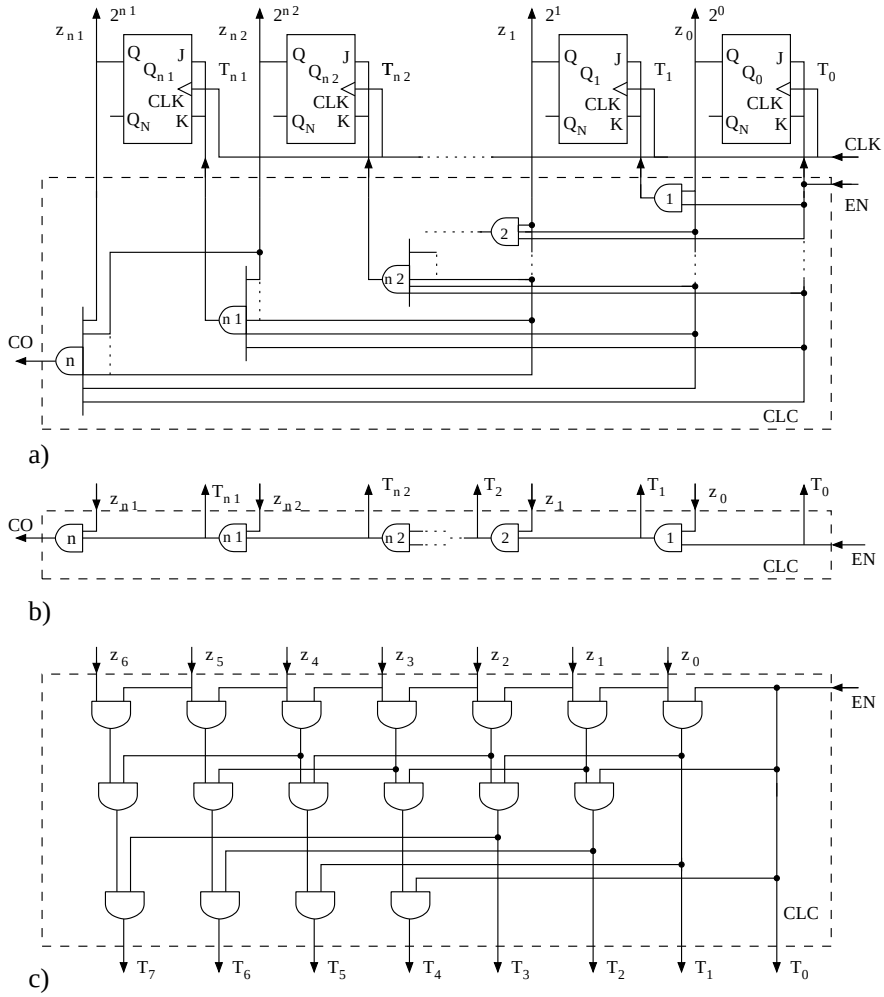


Figura 3.63 Automatul numărător sincron în codul binar natural: a) organizarea numărătorului sincron paralel; b) structura reţelei CLC pentru numărătorul sincron serie; c) structură arborescentă binară a reţelei combinaţionale pentru numărătorul sincron (n=8)

două tipuri de numărătoare: numărătorul sincron paralel, Figura 3.63-a, numărătorul sincron serie Figura 3.63-b. Aceste două implementări, deşi au produsul dimensiune-adâncime în acelaşi ordin, $O(n^2)$, reflectă regula că micşorarea dimensiunii la varianta serie duce la scăderea performanţei de viteză, iar la micşorarea adâncimii duce la creşterea dimensiunii (varianta paralelă) vezi relatia 2.9.

Pentru **numărătorul sincron paralel (NSP)**, din momentul aplicării impulsului de ceas până la următorul, timpul minim necesar, T_{CLK} minim, este egal cu τ_{pCQ} — timpul de propagare printr-o celulă T, plus timpul τ_{pAND}-timpul de propagare printr-o poartă AND, la care se adaugă timpul de stabilizare al bistabilului τ_{SU}, ceea ce se poate exprima prin relaţia generală

$$T_{CLKmin} = \tau_{NSP}(n) = \tau_{\text{pCQ}} + \tau_{\text{pAND}} + \tau_{SU} \in O(1) \tag{3.40}$$

Dimensiunea acestui tip de numărător, $S_{NSP}(n)$, se calculează din dimensiunea reţelei de memorare realizată din bistabilele T, S_T, plus dimensiunea reţelei combinaţionale (paralelă) S_{RCP}(n) compusă din n porţi cu două până la (n+1) intrări.

$$S_{NSP}(n) = n \cdot S_T + n \cdot S_{RCP}(n+1) \in O(n^2)$$

Aceste evaluări teoretice ale automatului numărător trebuie considerate doar ca limite, deoarece în practică implementarea acestui numărător, mai ales pentru n de valoare mare, este greu de realizat (ultima poartă AND prezintă n+1 intrări). În consecinţă, implementarea porţilor AND cu multe intrări, utilizând axioma de asociativitate, se vor compune din mai multe porţi cu un număr de intrări mai redus şi plasate pe mai multe niveluri, deci nu va fi un singur nivel de propagare τ_{pNAND} ci mai multe. De asemenea şi τ_{pCQ} se măreşte, mai ales la bistabilele din rangurile inferioare, a căror încărcare la ieşire creşte cu n. Totuşi, organizarea de numărător sincron paralel pentru valori mici ale lui n este recomandată, sunt uzuale numărătoarele cu n=4.

Prin utilizarea expresiilor 3.38-b se obţine **numărătorul sincron serie, NSS**, pentru care în Figura 3.63-b este prezentată numai organizarea reţelei combinaţionale. Pentru această organizare, în raport cu cea paralelă, perioada minimă a semnalului de ceas, T_{CLKmin}, se măreşte cu timpul de propagare prin încă (n-2) porţi AND2; într-adevăr funcţia de excitaţie T_i a bistabilului i este calculat numai după ce a fost calculată T_{i-1} ceea ce se exprimă prin relaţia

$$T_{CLKmin} = \tau_{NSS}(n) = \tau_{\text{pCQ}} + (n-1)\tau_{\text{pAND}} + \tau_{SU} \in O(n) \tag{3.41}$$

iar dimensiunea

$$S_{NSS}(n) = n \cdot S_T + n \cdot S_{AND2} \in Q(n)$$

Relaţia 3.41 arată că performanţele de viteză ale numărătorului sincron serie nu sunt mult îmbunătăţite în raport cu numărătorul asincron la care ($T_{CLKmin} \geq n\tau_{CQ}$)

Această structurare serie şi paralelă este similară cu cea a sumatorului serie cu transport progresiv (secţiunea 2.5.2.1) şi cea a sumatorului cu transport anticipat (secţiunea 2.5.2.2); la fel, ca la sumatoare, se pot face organizări ale CLC pentru calculul funcţiilor de excitaţie T_i prin mixarea parţială a celor două extreme. De exmplu se poate realiza calculul paralel pentru câte m celule bistabile, iar apoi calculul în serie între cele n/m grupe.

Se pot calcula funcţiile de excitaţie T_i, utilizând o funcţie logică AND2, $f(x_0, x_1) = x_0 \cdot x_1$, sub forma unui arbore binar luând numai perechi din variabilele EN, z_0, z_1, z_2,...,z_{n-2}, z_{n-1}. Aplicând această modalitate de calcul pentru relaţiile 3.38-a când $n = 8$ se obţine

$$\begin{aligned}
T_0 &= f(EN, EN) = EN \\
T_1 &= f(z_0, EN) \\
T_2 &= f(f(z_1, z_0), EN) \\
T_3 &= f(f(z_2, z_1), f(z_0, EN)) \\
T_4 &= f(f(f(z_3, z_2), f(z_1, z_0)), EN) \\
T_5 &= f(f(f(z_4, z_3), f(z_2, z_1)), f(z_0, EN)) \\
T_6 &= f(f(f(z_5, z_4), f(z_3, z_2)), f(f(z_1, z_0), EN)) \\
T_7 &= f(f(f(z_6, z_5), f(z_4, z_3)), f(f(z_2, z_1), f(z_0, EN)))
\end{aligned} \tag{3.42}$$

Organizarea sub formă de arbore (NSA), numai cu porţi AND2, conform relaţiilor 3.42, este prezentată în Figura 3.63-c. Perioada de ceas minimă pentru cazul general este

$$T_{CLKmin} = \tau_{NSA}(n) = \tau_{\text{pCQ}} + \tau_{\text{pAND2}} \cdot log_2\ n + \tau_{SU} \in O(log\ n) \tag{3.43}$$

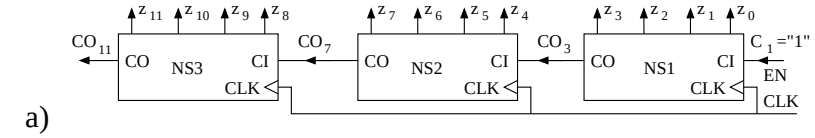
iar dimensiunea se calculează cu relaţia

$$S_{NSA}(n) = n \cdot S_T + n \cdot log_2\ n \in O(n \cdot log\ n)$$

Pentru un numărător sincron modulo 2^n, în cod binar natural, cu numărare inversă sinteza se realizează similar. Se porneşte pentru deducerea funcţiilor de excitaţie de la succesiunea stărilor asignate parcurse în sens invers din Figura 3.62-a. Se observă că fiecare bit z_i, $i \neq 0$, din cuvântul stării următoare, va comuta faţă de valoarea din starea prezentă numai când în starea prezentă toţi biţii anteriori au valoarea 0; bitul z_0 comută la fiecare impuls de ceas. Deci faţă de numărarea în sens direct, când se utilizează ieşirile Q de la celulele anterioare, la numărarea în sens invers se utilizează ieşirile Q_N ale celulelor anterioare. Pentru implementarea numărătorului invers se utilizează relaţiile 3.38 şi 3.39 în care se face substituţia $Q_i \rightarrow Q_{Ni}$. Semnalul de depăşire de capacitate/transport CO, care în acest caz are semnificaţia de împrumut, se generează când numărătorul ajunge la cuvântul compus din n zerouri şi nu când numărătorul ajunge la cuvântul de n unu-uri ca la numărarea directă.

Un numărător sincron reversibil trebuie să calculeze funcţiile de excitaţie cu semnalele Q_i, pentru numărare directă, şi cu semnalele Q_{Ni} pentru numărare inversă. Selectarea celor două semnale de la ieşirea unei celule bistabil se realizează, similar ca în Figura 3.62-e, fie un MUX2:1, fie cu o poartă XOR comandate cu un semnal de $S = \overline{Direct}/Invers$ ($\overline{D}/I$, up/down); pentru $S = 0$ numărător direct, pentru $S = 1$ numărător invers.

În general, numărătoarele existente sub formă de circuite integrate discrete sunt module numărător sincron compuse din patru celule bistabil (modulo 2^4); iar pentru capacităţi de numărare mai mari se înseriază astfel de module, Figura 3.64-a. Pentru înseriere, semnalul de depăşire de capacitate CO de la un modul se conectează ca semnal de transport anterior CI la modulul următor, iar semnalul de ceas se aplică simultan la toate modulele. În momentul când într-un modul se ajunge la capacitatea maximă 1111 se generează CO=1, deci următorul modul va fi incrementat la



a)

Nr de imp de ceas	CO_{11}	z_{11}	z_{10}	z_9	z_8	CO_7	z_7	z_6	z_5	z_4	CO_3	z_3	z_2	z_1	z_0	EN= C_1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
15	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
16	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
255	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
256	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
257	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
4095	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4096	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
4097	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

b)

Figura 3.64 Extensia capacităţii de numărare : a) organizare pentru modulo 2^{12} prin înserierea a trei circuite numărător modulo 2^4; b) secvenţă de cuvinte din procesul de numărare modulo 2^{12} în cod binar natural.

următorul impuls de ceas. Câteva cuvinte din conţinutul unui numărător modulo 2^{12}, organizat prin înserierea a trei modulele fiecare de câte patru biţi, din secvenţa de numărare sunt prezentate în tabelul din Figura 3.64-b. Dacă pentru această înseriere se folosesc module NSP perioada minimă a semnalului de ceas conform relaţiei (3.40), este

$$T_{NSP}(12) = \tau_{\text{pCQ}} + 3 \cdot \tau_{\text{pAND}} + \tau_{SU}$$

iar dacă se folosesc module NSS, conform relaţiei 3.41 perioada minmă este

$$T_{NSS}(12) = \tau_{\text{pCQ}} + (4 + 4 + 3)\tau_{\text{pAND}} + \tau_{SU}$$

3.4.2.1 Numărătoare presetabile

La numărătoarele sincrone presetabile se poate realiza ca, la un anumit impuls de ceas, tranziţia să se efectueze într-o stare următoare, q_p prescrisă din exterior. Codul stării următoare, starea de început a ciclului q_p, validat prin acţiunea semnalului de control de încărcare LOAD (în general activ în L) este înscris în celulele bistabil de stare ale automatului numărător. Dar, înscrierea (forţarea) unei stări următoare q_p din exterior înseamnă că în graful de tranziţii ciclic, Figura 3.61, nu se mai parcurg toate cele C stări, deci numărătorul devine un numărător modulo C_1; C_1 este egal cu numărul de stări parcurse, pe graful de tranziţie al stărilor, între starea finală q_f (din care se realizează forţarea tranziţiei în starea prescrisă q_p) şi starea de început a

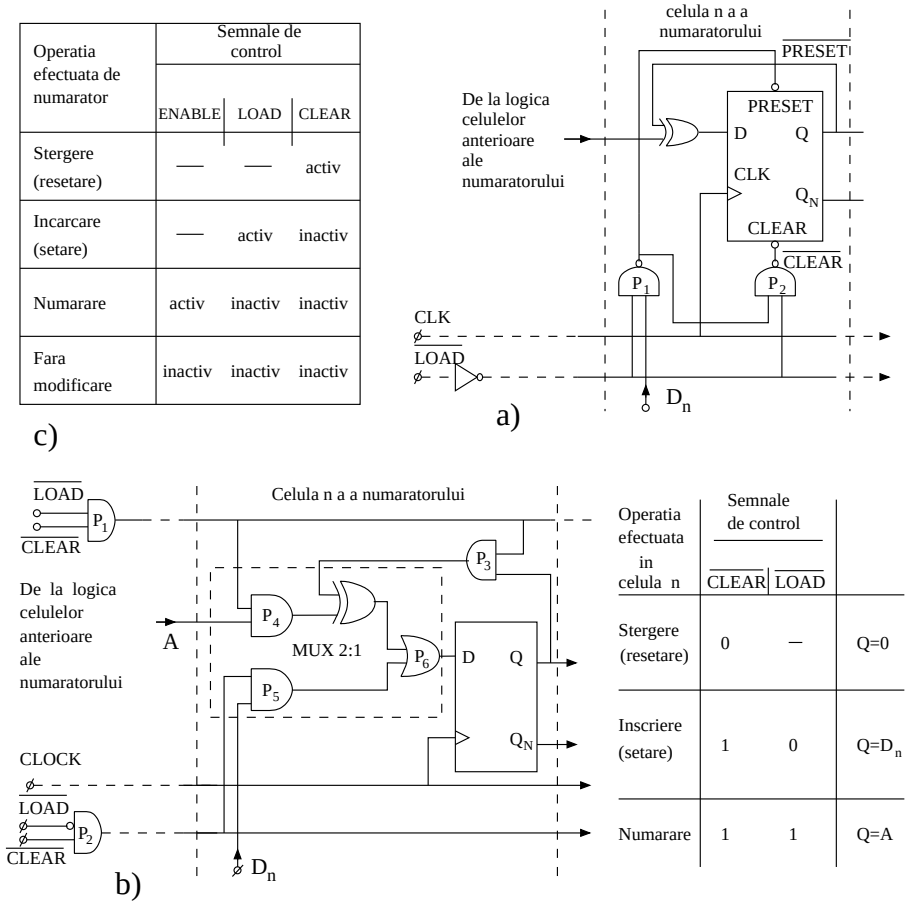


Figura 3.65 Modalitaţi de înscriere a numărătoarelor sincrone: a) înscriere asincronă; b) circuistica şi tabelul semnalelor de control pentru înscrierea sincronă; c) semnalele pentru controlul operaţiilor pe un numărător.

ciclului q_p. Rezultă că un numărător modulo 2^n, la care se implementează circuitul de presetare, poate fi transformat într-un numărător modulo C_1, $2 \leq C_1 \leq 2^n$-1.

Circuitul de presetare trebuie să conţină: un identificator (decodificator) al codului stării finale q_f, intrări externe pe care se aplică biţii de cod ai stării q_p, ce urmează a fi înscrisă, şi semnalului de încărcare, LOAD. Un decodficator al codului stării q_f se poate realiza uşor cu porţi logice în exterior, în general sub forma unei conjuncţii compuse cu biţii care au valoarea 1 în codul stării q_f. Dar, un astfel de decodificator poate fi eliminat dacă se alege ca starea finală, q_{C-1}, în care se generează semnalul de depăşire a capacităţii, CO, iar ca semnalul LOAD se va utiliza tocmai acest semnal de depăşire de capacitate. Foarte frecvent, mai există încă un semnal de ştergere, CLEAR (în general activ în L), care realizează înscrierea în starea $q_i = 00\ldots00$. Forţarea codului stării următoare în bistabilele de stare ale automatului numărător se poate realiza, în raport cu semnalul de ceas, sincron sau asincron.

Modalitatea de înscriere asincronă, Figura 3.65-b, utilizează intrările asincrone PRESET şi CLEAR (Figura 3.43-a) ale latch-ului din structura bistabilului. Cele două porţi P_1, P_2 sunt validate numai la activarea semnalului de încărcare, $\overline{\text{LOAD}} = 0$. Dacă bitul D_n, aplicat din exterior, are valoarea 1 rezultă $\overline{\text{PRESET}}$, $\overline{\text{CLEAR}} = 0$, 1, deci $Q = 1$, $Q_N = 0$, iar dacă D_n are valoarea 0 rezultă $\overline{\text{PRESET}}$, $\overline{\text{CLEAR}} = 1$, 0, deci $Q = 0$, $Q_N = 1$. Poarta XOR din această structură are numai rolul de a transforma bistabilul D în bistabilul T, Figura 3.50-b. Înscrierea asincronă întru-un numărător prezintă două inconveniente:

1. Buclă transparentă pentru generarea semnalului de încărcare. Prin utilizarea semnalului obţinut de la decodificatorul stării q_f si aplicarea acestuia ca semnal de încărcare LOAD in jurul numărătorului se închide o buclă transparentă care poate determina, eventual, înscrierea unei stări eronate, eventualitate analizată în Figura 3.62-f. Se poate evita această eventualitate prin introducerea unui latch, deci eliminarea transparenţei buclei.

2. Starea prescrisă din exterior, q_p trebuie să fie totdeauna cea anterioară stării de început a ciclului. La aplicarea impulsului de ceas care determină tranziţia în starea finală q_f pe ieşirile numărătorului apare codul stării finale (pentru care decodificatorul generează pe o durată scurtă de timp semnalul de încărcare asincronă) şi apoi codul stării forţate $q_{\text{p-1}}$ la numărarea directă sau $q_{\text{p+1}}$, la numărarea inversă; deci pe perioada acestui impuls de ceas ieşirile indică două stări q_f, pentru o durată scurtă de timp, şi $q_{\text{p-1}}$ sau $q_{\text{p+1}}$. Numai la următorul impuls de ceas se trece în starea de început a ciclului q_p şi se continuă tranziţiile până în q_f.

Pentru înscrierea sincronă, deoarece celulele bistabil sunt înscrise de către datele ce se aplică pe intrările de date când se aplica frontul activ de ceas, nu apar inconvenientele de la înscrierea asincronă. În Figura 3.65-b este prezentată circuistica si tabelul cu semnalele de control pentru o înscriere sincronă. Poarta XOR are numai rolul de a realiza bucla ce transformă bistabilul D în T; bucla se închide de la ieşirea Q înspre intrare când poarta P_3 este validată cu semnalul $\overline{\text{LOAD}}$, $\overline{\text{CLEAR}} = 1,\ 1$. Celula are pe intrarea de date o structură de MUX2:1 (neconsiderându-se poarta XOR). Pe o intrare a multiplexorului, poarta P_4, se aplică semnalul (funcţia de excitaţie) A, obţinut de la circuitul combinaţional al numărătorului, şi care este data ce determină starea celulei, $Q = A$, pe frontul activ următor de ceas, dacă porţile P_3 şi P_4 sunt validate prin valoarea 1 de la ieşirea porţii P_1 (când $\overline{\text{LOAD}}$, $\overline{\text{CLEAR}} = 1,\ 1$). Pe cealaltă intrare, poarta P_5, se aplică D_n care este înscrisă în celulă, $Q = D_n$, la următorul impuls de ceas dacă valoarea generată de poarta P_2 este 1 ($\overline{\text{LOAD}}$, $\overline{\text{CLEAR}} = 0,\ 1$, adică operaţia de înscriere). Pentru operaţia de ştergere ($\overline{\text{LOAD}}$, $\overline{\text{CLEAR}} = -,\ 0$) ieşirea în 0 a porţii P_2 va forţa în celulă $Q = 0$, la apariţia următorului impuls de ceas, indiferent de valoarea de prescriere D_n.

În general, semnalele de control pentru un numărător presetabil sunt: de încărcare — LOAD, de ştergere — CLEAR şi de validare — ENABLE (uneori compus din conjuncţia a două semnale). Din tabelul din Figura 3.65-a rezultă că operaţia de ştergere, CLEAR-activ, se realizează indiferent de valorile celorlalte semnale de control, deci CLEAR are prioritate maximă. Activarea semnalului LOAD încarcă din exterior celulele numărătorului chiar dacă acesta nu este validat, pentru operaţia de numărare, prin activarea semnalului ENABLE. Dacă toate cele trei semnale de control nu sunt activate numărătorul este inhibat/"îngheţat" într-o anumită stare, nu se produce nici o modificare la aplicarea semnalelor de ceas. Toate semnalele de control

sincrone trebuie să respecte în rapot cu frontul activ al semnalului de ceas restricţiile impuse de timpul de stabilizare τ_{SU} şi timpul de menţinere, τ_H. Semnalele de control pentru circuitele numărătoare sincrone uzuale, existente sub formă de circuite integrate MSI, sunt prezentate în Figura 3.66-a.

Pentru numărătorul sincron 74xx163, 4biţi, cod binar natural, unul dintre cele mai uzuale numărătoare, prin diagramele de semnal din Figura 3.66-b, se prezintă modurile de operare prin activarea semnalelor de control. Pe frontul pozitiv al impulsului de ceas notat cu 1, deoarece comanda de ştergere este activă,$\overline{\text{CLEAR}} = 0$, conţinutul numărătorului devine $Q_3Q_2Q_1Q_0 = 0000$, iar pe frontul celui de al doilea impuls de ceas, deoarece comanda de înscriere este activă, $\overline{\text{LOAD}} = 0$, cuvântul aplicat din exterior pe intrări $D_3D_2D_1D_0 = 1100$ devine conţinutul numărătorului $Q_3Q_2Q_1Q_0 = 1100 = 12|_{10}$. Imediat după al doilea impuls de ceas, prin activarea semnalului ENABLE = ENP $\cdot$ ENT = 1, se validează regimul de numărare. Pentru validarea regimului de numărare se conjugă cele două semnale de validare **ENT** (**EN**able **T**rickle) şi **ENP** (**EN**able **P**arallel). Diferenţa între aceste două semnale constă prin efectul lor; ambele validează operaţia de numarare dar ENT mai validează şi transportul următor al numărătorului, notat aici cu prin **RCO** (**R**ipple **C**arry **O**ut), ca o funcţie AND cu semnalul CO, de numărător plin, adică RCO = CO $\cdot$ ENT. La al cincelea semnal de ceas conţinutul numărătorului se umple $Q_3Q_2Q_1Q_0 = 1111$ şi se generează RCO, semnal care utilizat ca o comandă de înscriere poate, pe următorul impuls de ceas, forţa conţinutul numărătorului în starea $q_p = Q_3Q_2Q_1Q_0 = D_3D_2D_1D_0$. Regimul de numărare continuă până după al optulea impuls de ceas când numărătorul este devalidat, ENP $\cdot$ ENT = 0, în continuare fiind în regim de inhibare (fără modificări).

Divizoare de frecvenţă. Deoarece un numărător modulo 2^n generează prin CO (RCO) un semnal cu frecvenţa impulsurilor de ceas divizată cu 2^n atunci oricare numărător modulo C_1, $2 \leq C_1 \leq 2^n - 1$, este un divizor cu C_1 a frecvenţei de ceas. Iar, după cum se ştie, un numărător modulo C_1 se obţine dintr-un numărător presetabil modulo 2^n din care se elimină $(2^n - C_1)$, stări, deci tot la C_1 impulsuri de ceas se generează un impuls în exterior, impuls ce este utilizat si pentru a forţa înscrierea stării de început q_p a ciclului în automatul numărător.

Inconvenientul unui semnal obţinut de la un numărător modulo C este o valoare mult sub 50% a coeficientului de umplere. Mai mult, dacă $C \neq 2^n$ atunci şi semnalele generate la ieşirile Q_{n-1},Q_{n-2},...,Q_1,Q_0 ale celulelor bistabil nu mai sunt simetrice pe durata ciclului de C impulsuri de ceas. Soluţii de circuite care generează semnale simetrice după fiecare celulă bistabil şi pentru un C$\neq 2^n$ se găsesc în [Oberman '78]. Pentru a obţine semnale cu frecvenţa f_{CLK}/C, dar cu coeficient de exemplu 50%, se poate utiliza următoare modalitate: se aplică o frecvenţă $2f_{CLK}$ la numărătorul modulo C iar ieşirea acestuia se divide printr-un bistabil T (T=1) care generează o frecvenţă ($2f_{CLK}$/C):2.

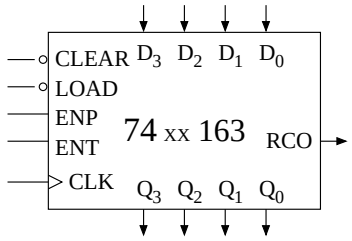
Exemplul 3.22 Utilizând circuitele numărător sincron presetabil 74xx163 şi 74xx161 să se realizeze numărătoare modulo 12. Caracteristicile de control pentru aceste numărătoare sunt date în Figura 3.66-a.

Soluţie. Implementarea cu 74xx163.

1. Figura 3.67-a, starea finală este $Q_3Q_2Q_1Q_0 = 1111$ detectată prin semnalul RCO, semnal care generează semnalul de încărcare sincron $\overline{\text{LOAD}} = \overline{\text{RCO}}$. Starea de început

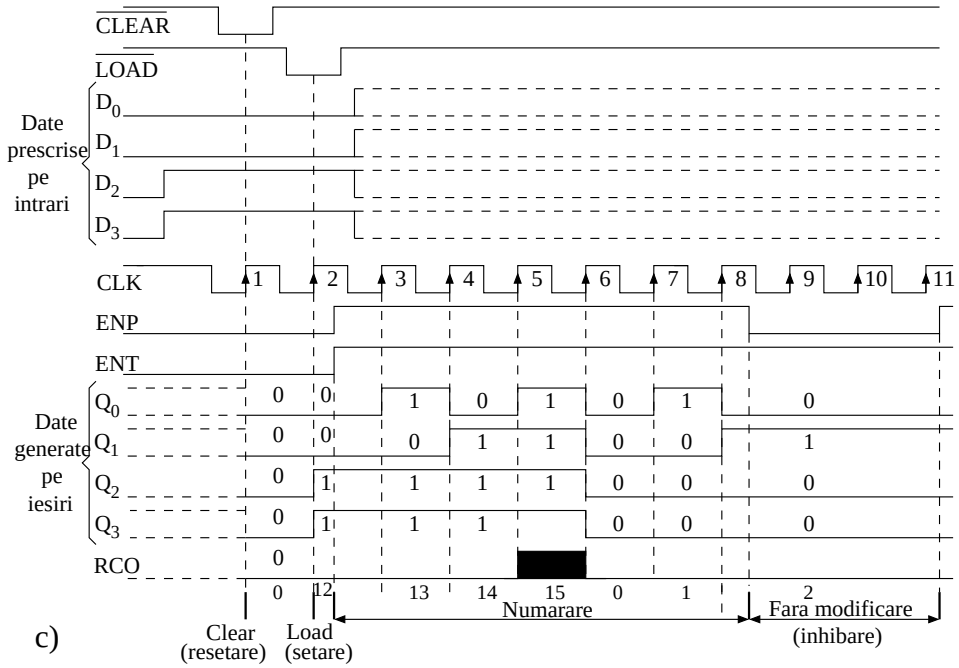
Codul circuitului	Tip de numarator (modulo, cod)	Comutare (front)	Resetare activa in (CLEAR)	Inscriere activa in (LOAD)	Alte caracteristici
74760/160	BCD/4 biti, binar	↑	L, asincron	L, sincron	
74162/163	BCD/4 biti, binar	↑	L, asincron	L, sincron	
4518/20	Dual BCD/4 biti, binar	↑ sau ↓	H, asincron	–	
74190/191	BCD/4 biti, binar	↑	–	L, asincron	Reversibil, o singura intrare ceas
74168/169	BCD/4 biti, binar	↑	–	L, sincron	Reversibil, o singura intrare ceas
74668/669	BCD/4 biti, binar	↑	–	L, sincron	Varianta de '168/169 imbunatatita
74568/569	BCD/4 biti, binar	↑	L,sinc/asinc	L, sincron	Ca si '668/669 dar cu iesiri TSL
4510/16	BCD/4 biti, binar	↑	H, asincron	H, asincron	Reversibil, o singura intrare ceas
74192/193	BCD/4 biti, binar	↑	L, asincron	H, asincron	Reversibil, doua intrari de ceas
74876	8biti, reversibil	↓ sau ↓	L, asincron	L, sincron	74869 are $\overline{\text{CLEAR}}$ sincron

a)



CLEAR stergere continut(sincron)
LOAD incarcare continut (sincron)
ENP, ENT validare functionare
Q_3, Q_2, Q_1, Q_0 iesire (de pondere), starea numaratorului
D_3, D_2, D_1, D_0 intrari pentru bitii cuvantului incarcat din exterior
RCO semnalizare stare $Q_3Q_2Q_1Q_0$=1111

b)



c)

Figura 3.66 Numărătoare sincronă sub formă de circuite MSI: a) caracteristicile numărătoarelor uzuale; b) diagrama de semnale pentru numărătorul 74xx163

a ciclului $0100 = 4|_{10}$ se înscrie la impulsul următor după cel care a comandat tranziţia în starea finală, $f_{RCO} = 1/12 \cdot f_{CLK}$. Numărarea nu este în cod binar natural, este, de fapt, un cod exces 4.

2. Figura 3.67-b, starea finală este $Q_3Q_2Q_1Q_0 = 1011 = 11|_{10}$, detectată de o poartă NAND3, care generează semnalul $\overline{\text{CLEAR}} = 0$. Ştergerea este sincronă cu semnalul de ceas, adică forţarea stării de început de ciclu $Q_3Q_2Q_1Q_0 = 0000$ se înscrie la impulsul următor după cel care a comandat tranziţia în starea finală. Activarea semnalului $\overline{CLEAR} = 0$ are frecvenţa $1/12 \cdot f_{CLK}$. Numărarea este în cod binar natural.

Implementarea cu 74xx161, care are încărcare sincronă dar ştergere asincronă. Un numărător divizor modulo 12 se obţine prin conexiuni similare ca în Figura 3.67-a. Pentru un numărător în cod binar natural, există următoarele 2 variante:

1. Figura 3.67-c, starea finală $1100 = 12|_{10}$ se detectează cu o poartă NAND2 care prin

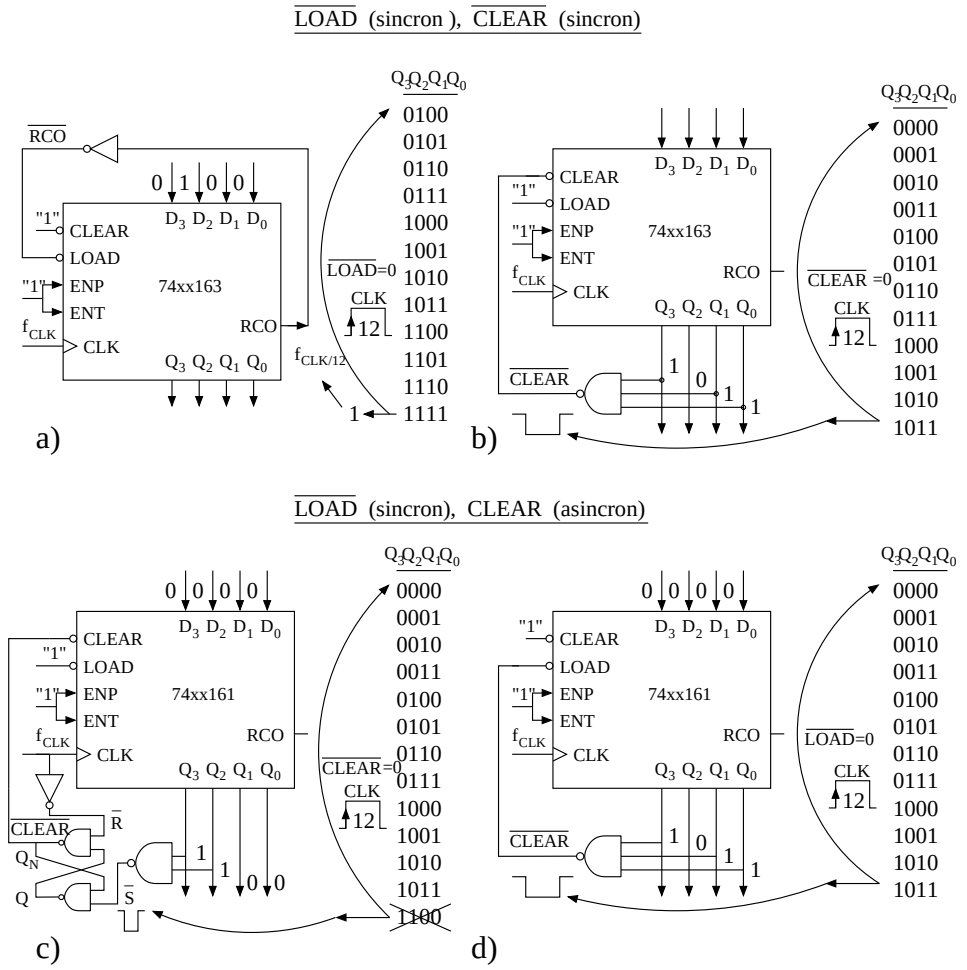


Figura 3.67 Numărătoare modulo 12: a) numărător divizor modulo 12; b,c,d) numărătoare modulo 12 în cod binar natural

semnalul $\overline{S}$, de scurtă durată, înscrie numărătorul în starea 0000. Deci pe perioada T_{CLK} a impulsului care a comandat tranziţia în starea finală există două stari, adica 1111 şi 0000; pentru ca în această perioadă starea a doua să fie 0000, şi nu o stare cu un alt cod, bucla ce se închide la intrarea asincronă CLEAR este transformată în buclă netransparentă prin introducerea unui latch $\overline{S}\,\overline{R}$. La următorul impuls de ceas se trece în starea iniţială 0001. Semnalul $\overline{\text{CLEAR}} = Q_N = 0$ are frecvenţa $1/12 f_{CLK}$.

2. Figura 3.67-b, structura şi funcţionarea se face prin utilizarea intrării sincrone LOAD, similar ca în Figura 3.67-b (unde se utilizează intrarea sincronă CLEAR).

Cu fiecare din aceste două circuite se mai pot realiza multe variante de numătoare modulo 12, dar nu cu numărare în cod binar natural, în funcţie de alegerea stării finale; oricare din cele 16 stări poate fi fixată starea finală.

Uneori este necesar ca pe baza unui numărător modulo C frecvenţa f_{CLK} să fie divizată cu un coeficient fracţionar C/k, $1 \leq k \leq C$. Se poate structura un divizor C/k ştiind că valoarea oricărui număr k exprimat în binar se obţine ca o sumă ponderată a puterilor lui doi cu biţii din cuvântul binar $...b_i..b_2b_1b_0$ ($k = \cdots + b_i 2^i + \cdots + b_2 2^2 + b_1 2^1 + b_0 2^0$). Dar, la un numărător modulo 2^n, după fiecare celulă de rangul i, $0 \leq i \leq n-1$, se pot obtine impulsuri de ceas divizate cu 2^{i+1}. Nu rămâne decât să se genereze aceste impulsuri şi, apoi, să se selecteze acestea de la fiecare celulă a numărătorului conform valorilor (ponderilor) bitilor din cuvântul de cod binar al numărului k. Aceste ponderi apar evidente dacă se consideră că frecvenţa rezultată $f_{CLK} : C/k$ se obţine ca un produs $k/C \cdot f_{CLK}$; astfel de circuite care realizează această relaţie sunt referite ca **multiplicatoare cu coeficient binar**.

Exemplul 3.23 Să se structureze un circuit pentru multiplicarea f_{CLK} cu coeficientul 5/16.

Soluţie. Numărul 5 exprimat în binar cu patru biti este $b_3b_2b_1b_0 = 0101$ iar exprimat ca parte a lui $16|_{10} = 10000$ devine fracţia $0, b_3b_2b_1b_0 = 0,0101(5|_{16} = 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4})$.

În circuitul multiplicator cu coeficientul binar 0,0101 frecvenţele divizate prin puterile lui 2, obţinute după celulele unui numărător modulo 2^4, se ponderează în felul următor:

$$b_3 \cdot 2^{-1} \cdot f_{CLK}; \qquad b_2 \cdot 2^{-2} \cdot f_{CLK}; \qquad b_1 \cdot 2^{-3} \cdot f_{CLK}; \qquad b_0 \cdot 2^{-4} \cdot f_{CLK}.$$

Utilizând un numărător sincron, paralel, modulo 16, Figura 3.68-a, se realizează cu porţi AND, în exteriorul acestuia, circuitele care generează frecvenţe de ceas divizate după puterile lui doi. Pe aceste porţi AND se introduc coeficienţii binari b_3, b_2, b_1, b_0 care ponderează subfrecvenţele respective. Funcţiile implementate pe fiecare poartă AND sunt

$$\begin{array}{ll} AND_4 : b_0 \cdot \overline{Q_3} Q_2 Q_1 Q_0 \cdot f_{CLK} & AND_2 : b_2 \cdot \overline{Q_1} Q_0 \cdot f_{CLK} \\ AND_3 : b_1 \cdot \overline{Q_2} Q_1 Q_0 \cdot f_{CLK} & AND_1 : b_3 \cdot \overline{Q_0} \cdot f_{CLK} \end{array}$$

Apoi trenurile de impulsuri obţinute la ieşirile acestor porţi AND sunt sumate printr-o poartă OR. Porţile AND_1 şi AND_3 care sunt ponderate cu biti cu valoarea zero ($b_3 = 0$, $b_1 = 0$) au aport nul în succesiunea impulsurilor de ieşire. Se obţine astfel, Figura 3.68-b, 5 impulsuri pe ieşire pentru 16 impulsuri de ceas aplicate la intrarea numărătorului deci o multiplicare cu 5/16. Succesiunea de 5 impulsuri nu este uniformă, distanţa între impulsuri este de patru iar uneori de două tacte de ceas. O distribuţie mult mai uniformă se obţine când se utilizează ieşirea de transport de la un circuit acumulator (vezi Figura 3.74-c). Pe lângă ieşirea O_n

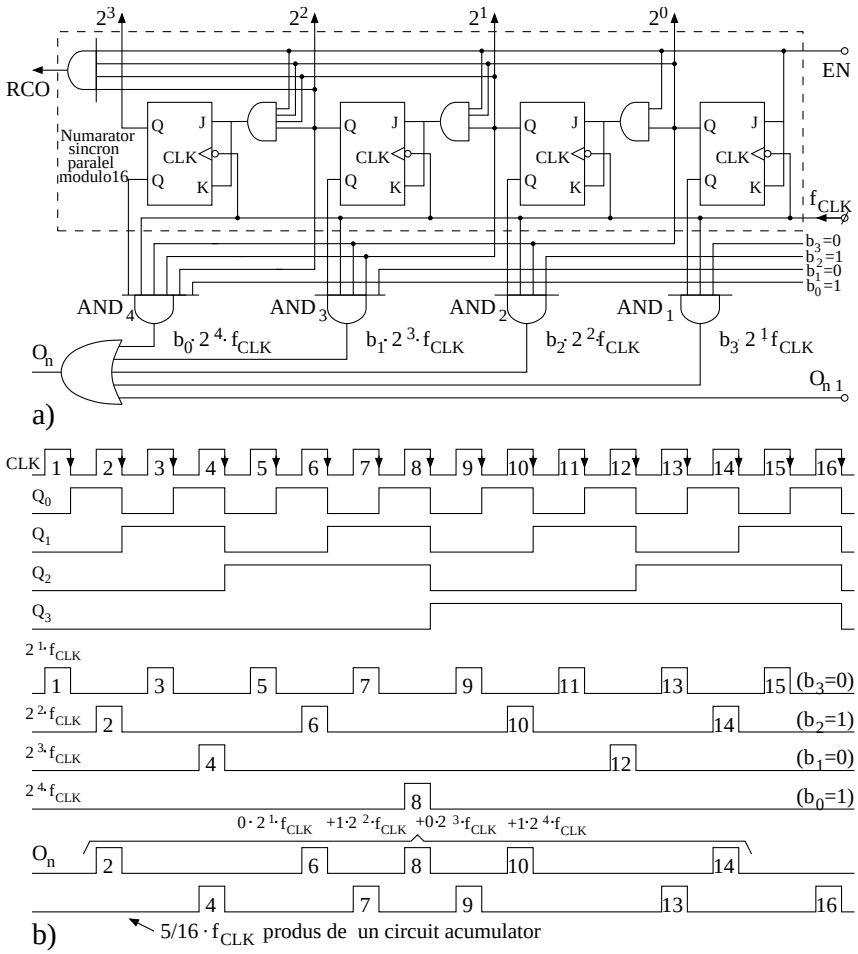


Figura 3.68 Multiplicatorul cu coeficient binar: a) organizarea circuitului; b) diagrama de semnale generate pentru multiplicarea $5/16 \cdot f_{CLK}$.

există şi o intrare O_{n-1} pentru ca multiplicatorul cu coeficient binar să poată fi înseriat cu alte circuite de acelaşi tip (pentru a creşte valoarea modulo).

Automate pe bază de numărător Numărătorul sincron însuşi este un automat de tip Moore, biţii de ieşire sunt identici cu biţii de stare, $Y \equiv Q$ (dacă nu se consideră şi ieşirea CO); deci automatul este identic cu semiautomatul său. Numărătoarele presetabile sunt automate particularizate prin modul de asignare a stărilor: codificarea este în cod binar natural până la codul numărului $2^n - 1$ sau până la 9 (la cele BCD). Ideea simplificatoare, în realizarea unui automat, sugerează ca asignarea stărilor cu tranziţii succesive ale automatului să se facă în codul binar natural, iar atunci pentru implementarea semiautomatului corespunzător automatului să fie utilizat un circuit integrat numărător presetabil. Pentru cazurile când în

graful de tranziţie a automatului există tranziţie într-o stare cu un cod care nu este în ordinea de numărare (după un bloc de decizie când sunt două căi de tranziţie) acel cod trebuie forţat prin numărătorul presetabil. Pentru o tranziţie în afara ordinii de numărare, numărătorului trebuie să i se "livreze" două informaţii: codul stării următoare ce trebuie înscris precum şi semnalul de comandă al încărcării. În consecinţă, pe baza produsului cartezian intrare-stare, $X(t) \times Q(t)$, trebuie implementate două circuite combinaţionale, din care unul calculează codul stării următoare, iar celălalt determină activarea semnalului de încărcare, care vor închide două bucle în jurul numărătorului presetabil, Figura 3.69.

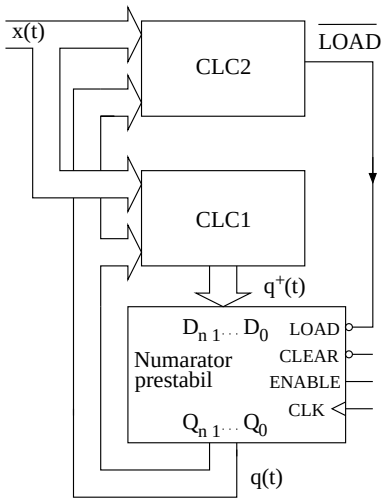


Figura 3.69 Organizarea de principiu a unui semiautomat pe bază de numărător presetabil.

Prima buclă, închisă prin CLC2, determină valoarea pe intrarea LOAD pentru comanda de înscriere. Recomandat pentru implementarea CLC2 este circuitul multiplexor pentru că trebuie să genereze o singură ieşire ($\overline{\text{LOAD}} = 0$ pentru înscriere, $\overline{\text{LOAD}} = 1$ pentru ordinea naturală de numărare) pe baza codului stării în care se află semiautomatul, iar biţii cuvântului de cod ai stării prezente q(t) se folosesc ca biti de selectie pentru multiplexor. În plus intrărilor multiplexorului, care prin selectare devin semnalul LOAD, pot să li se atribuie în afară de constantele 1 sau 0 şi valorile unei variabile de intrare. Determinarea valorii semnalului $\overline{\text{LOAD}}$, în funcţie rezultatul testării unei variabile într-un bloc de decizie, se realizează prin conectarea variabilei testate într-o anumită stare pe intrarea multiplexorului selectată de cuvântul de cod al acestei stare.

A doua buclă, inclusă prin CLC1, calculează biţii de cod $w_{n-1}, w_{n-2}, ..., w_1, w_0$ ai stării următoare, în cazul când nu este starea următoare $q^+(t)$ în ordinea normală de numărare. Pentru CLC1 recomandarea este o implementare cu porţi. Justificarea acestei recomandări se bazează pe faptul că prin asignarea aleasă, pentru stările semiautomatului, majoritatea tranziţiilor se efectuează între stări care au coduri succesive în ordinea normală de numărare, deci există puţine tranziţii între coduri care

strică această ordine normală. În consecinţă, pentru multe cuvinte de cod ale stării prezente $q(t)$ biţii stării următoare $q^+(t)$, calculaţi pe CLC1, sunt indiferenţi pentru prescrierea numărătorului. Existenţa multor combinaţii de intrare pentru care ieşirile de la CLC1 sunt indiferente duce la o minimizare puternică, deci la o implementare simplă cu porţi.

Exemplul 3.24 Pentru semiautomatul descris prin organigrama ASM şi asignarea fixată, din Figura 3.70, să se realizeze o implementare.

Soluţie. Se observă că pentru această variantă (Varianta1) de asignare a stărilor, când variabilele de intrare au valorile $x_1 = 1, x_2 = 0, x_3 = 1$, se parcurg ciclic stările $q_2 \to q_3 \to q_4 \to q_5 \to q_6 \to q_0 \to q_1 \to q_2 \to ...$, iar codurile parcurse sunt în ordinea de numărare în binar natural de la 000 până la 110; există un salt în cuvântul de cod de la 110 la 000 pentru calea de tranziţie $L_2(q_1 \to q_2)$. Mai există încă trei căi de tranziţie L_4, L_7 şi L_9 când iarăşi tranziţiile nu sunt în ordinea normală de numărare. Deoarece din cele 10 căi de tranziţie pentru 6 dintre acestea se parcurg coduri succesive de numărare în binar se va utiliza pentru implementare circuitul numărător 74xx163/161.

Tabelul de tranziţie al stărilor, Figura 3.70-b, conţine informaţia pentru sinteza celor două bucle ale semiautomatului. Deoarece sunt numai 7 stări intrarea D_3 a numărătorului este permanent în 0.

În coloana semnalului de încărcare $\overline{\text{LOAD}}$ se introduce valoarea 0(activ) numai atunci când codul stării următoare w_2, w_1, w_1 nu se obţine din codul stării prezente $z_2z_1z_0$ plus 1, în rest $\overline{LOAD}$=1(inactiv). Pentru sinteza semnalului $\overline{LOAD}$ intrările x_3, x_2, x_1 se introduc ca variabile reziduu. Bucla pentru încărcarea numărătorului este implementată pe un MUX8:1.

Pentru sinteza circuitului combinaţional din bucla ce calculează biţii D_2, D_1, D_0 valorile acestor biţi sunt identice cu cele ale biţilor stării următoare $D_2 = w_2$, $D_1 = w_1$, $D_0 = w_0$ când semnalul de încărcare este activ, $\overline{\text{LOAD}} = 0$, şi cu valori indiferent când $\overline{\text{LOAD}} = 1$. În expresiile biţilor D_2, D_1 şi D_0 se pot introduce intrările x_3, x_2, x_1 ca variabile reziduu rezultând o sinteză în funcţie numai de variabilele de stare z_2, z_1, z_0. Dar aceste variabile reziduu pot fi substituite cu una din valorile logice 0 sau 1 deoarece, la fiecare testare a unei variabile reziduu, numai una din cele două tranziţii posibile determină încărcarea numărătorului, pentru cealaltă valoare este indiferent ($\overline{\text{LOAD}} = 1$).

De exemplu, din starea prezentă $q_5(011)$ se efectuează tranziţia fie la q_5, când x_2=1, fie la $q_6(100)$, când x_2=0, dar ultima tranziţie este indiferentă pentru încărcarea numărătorului ($\overline{\text{LOAD}}$=1). Rezultă că în căsuţa de coordonate $z_2z_1z_0$=011(q_5) din diagramele V-K ale funcţiilor D_2, D_1, D_0 totdeauna se introduc respectiv valorile 0,1,1, Figura 3.70-d (nu există două valori diferite D_i care să fie determinate de cele două valori ale lui x_2). Din aceste diagrame V-K se deduc expresiile logice (pentru Varianta1):

$$D_2 = \overline{z_2}\ \overline{z_1} = \overline{z_1 + z_2}; \qquad D_1 = \overline{z_2}; \qquad D_0 = \overline{z_2}z_1 + z_2\overline{z_1} = z_2 \oplus z_1$$

iar structura rezultată de semiautomat este cea din Figura 3.70-c.

Se mai propun încă două variante de asignare a stărilor semiautomatului, ca în coloanele denumite Varianta2 şi Varianta3 din tabelul ASM, pentru a face o comparaţie cu implementarea după Varianta1. Pentru Varianta2 şi 3 nu se mai face sinteza buclei pentru determinarea comenzii $\overline{\text{LOAD}}$ deoarece o optimizare nu poate apare când se utilizează un circuit standard(MUX8:1). În schimb, pentru bucla ce calculează codul stării următoare, pentru a cărei sinteză contează numărul de stări indiferente din diagrama V-K, este normal a se determina care asignare este mai potrivită. Urmând aceeaşi procedură de sinteză, ca şi la Varianta1, se obţin

$$\begin{array}{lllll} Varianta2: & D_2 = \overline{z}_1\overline{z}_0 = \overline{(z_1 + z_0)}; & D_1 = z_0; & D_0 = 0 \\ Varianta3: & D_2 = z_1(\overline{z}_2 + \overline{x}_1); & D_1 = \overline{z_2 + z_0}; & D_0 = \overline{z}_1 + x_3\overline{z}_2 \end{array}$$

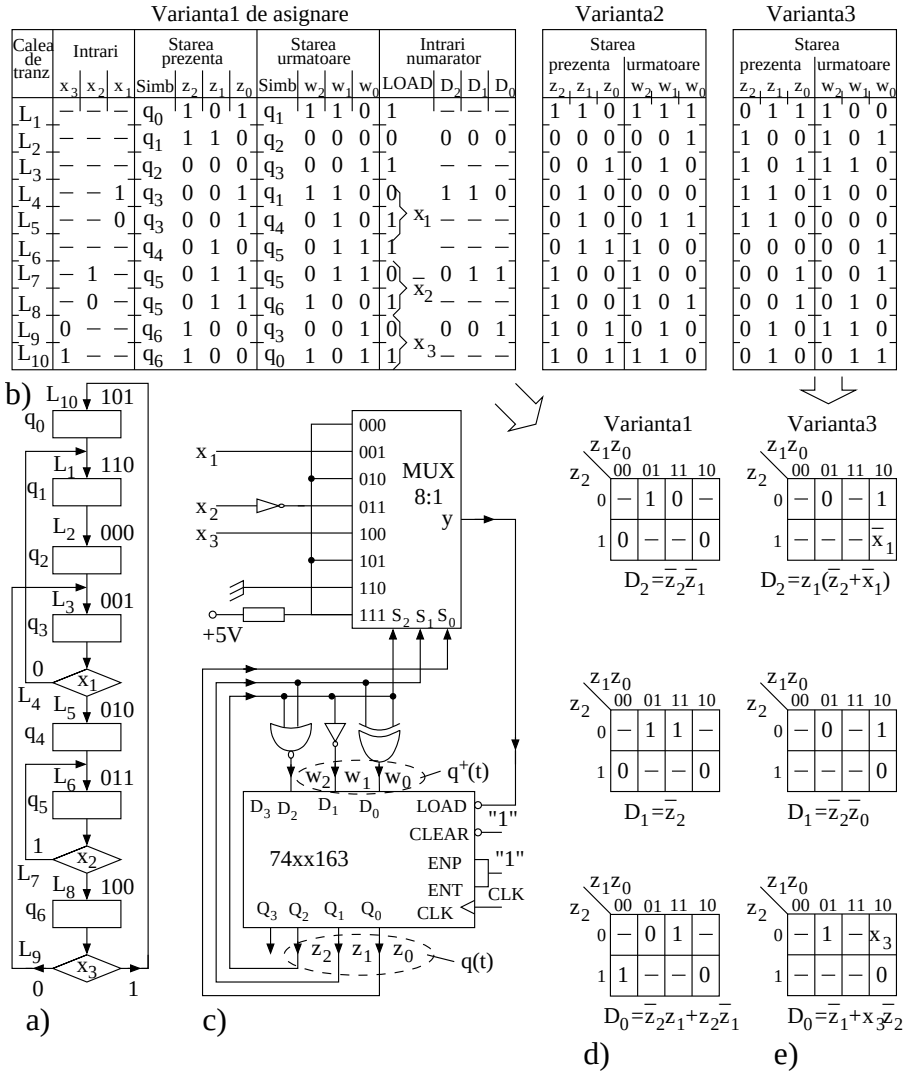


Figura 3.70 Exemplu de semiautomat implementat pe baza numărătorului 74xx163: a) organigrama ASM; b) tabelul combinat ASM; c) structura semiautomatului pentru Varianta1 de codificare; b,c) sinteza buclei pentru calculul stării următoare corespunzător Varianta1 şi Varianta3 de codificare.

Se poate observa că Varianta2, în raport cu celelalte două, realizează o implementare mai simplă pentru bucla ce calculează starea următoare (utilizează o singură poartă NOR2). Varianta3 este cea mai puţin simplă, deoarece în expresiile care calculează starea următoare intervin şi variabilele de intrare; nici una din tranziţiile controlate de x_1 şi x_3 nu sunt spre stări ale căror coduri să se obţină prin numărare.

3.4.2.2 Numărătoare în cod arbitrar

Un circuit numărător modulo C este un automat Moore cu un graf ciclic, deci poate fi privit ca un identificator de clase de resturi modulo C. Asignarea stărilor se face, în general, în cod binar natural — datorită faptului că în procesarea digitală în calculator numerele sunt reprezentate în sistemul de numeraţie binar — rezultând numărătorul binar sau în cod BCD-datorită faptului că în exteriorul calculatorului numerele sunt reprezentate în sistemul de numeraţie zecimal — rezultând numărătorul BCD. Dar, asignarea stărilor automatului numărător, Figura 3.61, se poate realiza în oricare cod, rezultând numărătorul în cod oarecare. Sinteza unui numărător în cod oarecare nu diferă cu nimic de sinteza unui automat. În anumite aplicaţii(evitarea apariţiei glitch-ului prin decodificare, utilizarea directă fără decodificator, o anumită secvenţă parcursă) se justifică alegerea unui anumit cod.

Exemplul 3.25 Să se realizeze, pe trei celule bistabil JK, un numărător în cod Gray.

Soluţie. Codurile progresive au proprietatea că la trecerea între două cuvinte de cod succesive se va schimba doar un singur bit. Pentru cele 16 cuvinte pe patru biţi se pot forma un număr de 55 de coduri progresive distincte, dintre acestea cel mai utilizat este codul binar reflectat(în raport cu o linie dusă după 2^i cuvinte de cod, cuvintele de cod de $(i+1)$ biţi de după linie sunt imaginea în oglindă a cuvintelor de cod de $(i+1)$ biţi dinainte de linie) este codul Gray.

Pornind de la tabelul de tranziţie al stărilor codificate în cod Gray, Figura 3.71-a, se deduc diagramele de V-K pentru biţii stării următoare w_2, w_1, w_0, apoi acestea pe baza tabelului de excitaţie al bistabilului JK, Tabelul 3.4, se convertesc în diagramele V-K, Figura 3.71-b,c şi d ale funcţiilor de excitaţie $JQ_2, KQ_2; JQ_1, KQ_1; JQ_0, KQ_0$ având următoarele expresii logice cu implementarea din Figura 3.71-e:

$$\begin{array}{lll} JQ_0 = z_1\ \overline{z_0} & JQ_1 = \overline{z_2}\ z_0 & JQ_0 = z_2 z_1 + \overline{z_2}\ \overline{z_1} \\ KQ_0 = \overline{z_1}\ z_0 & KQ_1 = z_2 z_0 & KQ_0 = z_2\ \overline{z_1} + \overline{z_2}\ z_1 \end{array}$$

Sinteza numărătoarelor în alte coduri de numărare se face în aceeaşi modalitate.

Dar există şi o altă abordare a unui numărător modulo C într-un cod oarecare, prin utilizarea unui numărător modulo C într-un cod uzual (realizat cu un circuit numărător presetabil) căruia i se ataşează un circuit de ieşire, CLC2 din Figura 3.8-c, care realizează funcţia de convertor din codul uzual folosit în codul oarecare. În Figura 3.71-f este structurată un numărător în cod Gray pe baza unui numărător modulo 256, în cod binar natural, plus o memorie ROM care conţine tabelul de conversie binar-Gray. Această abordare, în raport cu cea anterioară, are o viteză de numărare mai redusă, dar în schimb, prezintă o mai mare flexibilitate, poate fi implementat un numărător în orice cod numai prin schimbarea tabelului de conversie din ROM.

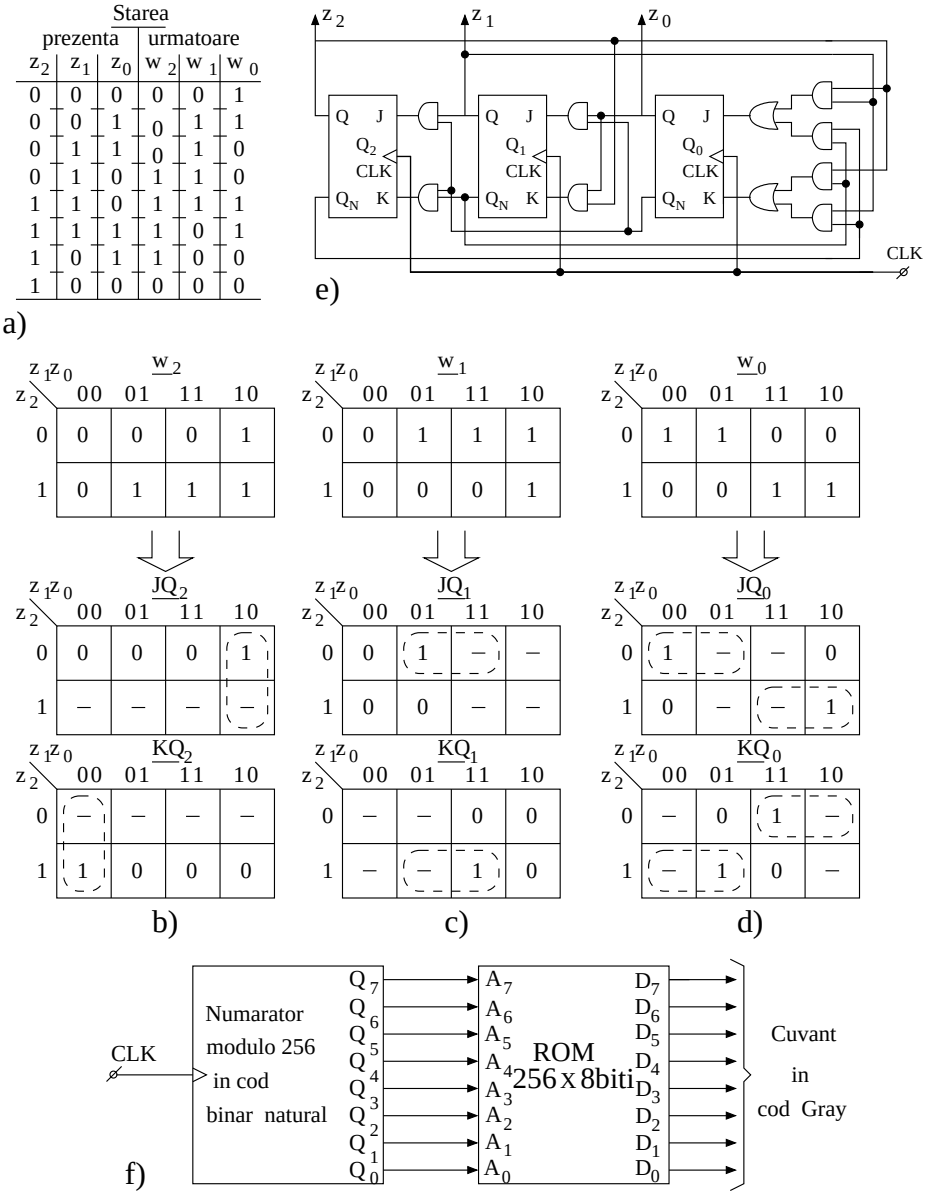


Figura 3.71 Numărător în cod Gray: a,b,c,d,e) fazele în procesul de sinteză, pornind de la tabelul de tranziţie al stărilor până la structura de circuit numărător; f) structurarea pe bază de circuit numărător binar modulo 256 plus convertor binar-Gray înscris în memorie ROM.

3.5 CIRCUITE REGISTRU

La nivel de bit suportul fizic pentru stocare/memorare, sincronizare, cuplare şi izolare este circuitul bistabil. La nivel de cuvânt, definit ca o succesiune de n biţi, suportul fizic pentru funcţiile de: **stocare/memorare, cuplare,sincronizare şi izolare/separare** este un circuit compus din n circuite bistabil referit ca circuit registru sau, uzual, registru. Odată înscrisă informaţia de un bit într-un bistabil aceasta este disponibilă a fi citită; la fel şi la un registru, cuvântul înscris poate fi citit în continuare. Într-o exprimare proprie sistemelor de procesare a informaţiei, unde informaţia este împachetată sub formă de cuvinte, un registru este referit prin termenul general de **port**. În această exprimare intrările de date, prin care se înscrie o informaţie/cuvânt într-un registru, constituie un **port de intrare**, iar ieşirile de date din registru constituie un **port de ieşire**.

În funcţie de modul cum se realizează conexiunile pentru extensia de la circuitul bistabil la organizarea de circuit registru există: **registru paralel**, **registru serie** şi combinaţii între acestea **serie-paralel** şi **paralel-serie**. Dar, pentru toate tipurile de registru, există un parametru comun **lungimea registrului**, adică numărul de celule bistabil care îl compun.

Circuitul registru, ca suport fizic pentru funcţiile enumerate mai sus, constituie o componentă fundamentală în arhitectura şi implementarea sistemelor digitale. Registrul este un circuit simplu, ca definiţie, şi cu o dimensiune de ordinul $O(n)$. Dar fiind, în fond, o structurare ordonată de circuite bistabil, în consecinţă, supus unei regularităţi de layout, rezultă că implementarea sa nu ridică dificultăţi. În secţiunile următoare se vor prezenta tipurile de circuite registru evidenţiind logica structurării acestora, precum şi unele aplicaţii care au un grad ridicat de generalitate.

3.5.1 Registru paralel

Registrul paralel are o structurare simplă obţinută doar prin considerarea în paralel a n celule bistabil, uzual de tip D; în afară de semnalul de ceas comun pentru toate celulele poate să nu existe nici o altă conexiune între celulele bistabil, Figura 3.72-a. Cuvântul pentru înscriere/(încărcare) se aplică pe intrările de date $D_{n-1}, D_{n-2}, ..., D_1, D_0$ — portul pe intrare — şi odată încărcat, prin aplicarea semnalului de ceas, cuvântul este accesibil permanent pe ieşiri $Q_{n-1}, Q_{n-2}, ..., Q_1, Q_0$ — portul de ieşire — dacă aceste ieşiri nu sunt de tip TSL. În general, registrul poate genera atât variabile negate, Q_N, cât şi cele nenegate, Q. Aceasta explică de ce la implementarea unei funcţii FNC sau FND, pe un circuit combinaţional, se consideră numai două niveluri logice şi nu trei deşi sunt utilizate atât variabile negate cât şi nenegate. Al treilea nivel pentru negarea unor variabile, situate la intrarea în circuit, nu este necesar deoarece atât variabilele negate cât şi cele nenegate sunt obtenabile de la registrul în care s-a stocat cuvâtul de intrare.

Pentru a spori flexibilitatea în utilizare, a structurării simple prezentate anterior, registrului i se adaugă anumite semnale de control, specificate pe schema bloc de registru reprezentată în Figura 3.72-b. În primul rând, un registru trebuie să prezinte obligatoriu un semnal de încărcare, LOAD (LD), care în general este semanlul de ceas, activ pe front (sau pe palier dacă registrul este realizat cu latch-uri). Uneori, pentru înscriere, în conjucţie cu semnalul LOAD, mai trebuie activat şi un semnal de

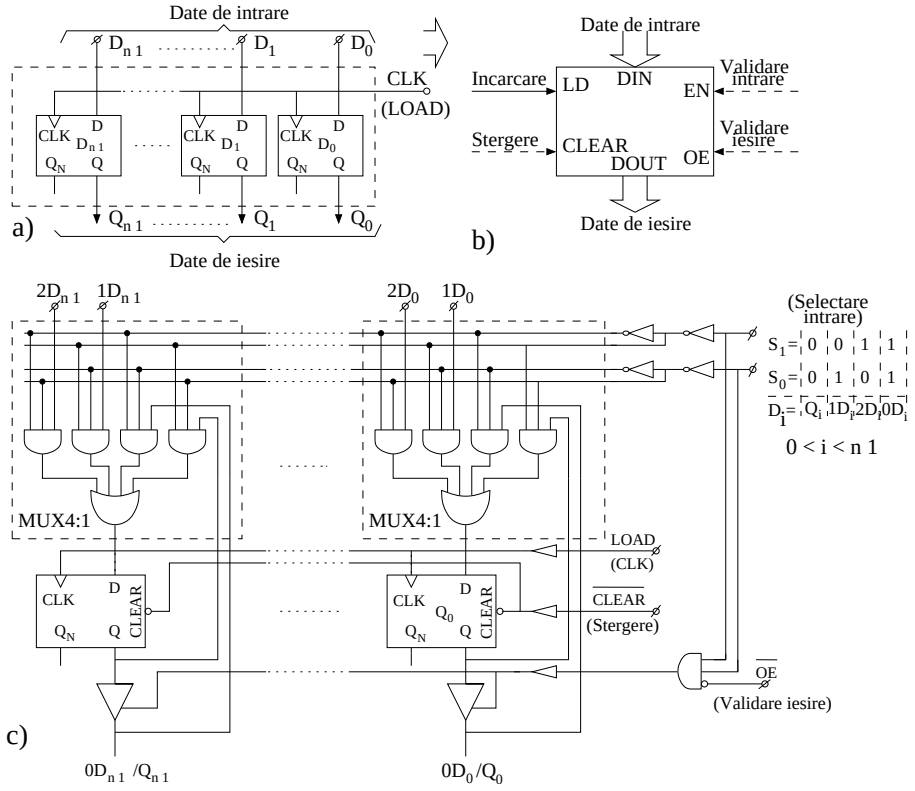


Figura 3.72 Registrul paralel: a) structură (cu n celule statice); b) reprezentare simbolică; structură de registru cu n celule în care se evidenţiază modul de aplicare a semnalelor de control.

validare, ENable.

Pentru facilitatea de ştergere a conţinutului registrului, în afară de înscrierea cu zero, poate exista un semnal de ştergere, CLEAR, care de fapt este o activare comună a tuturor intrărilor asincrone CLEAR ale celulelor bistabil. De asemenea, pentru a se putea realiza conectarea mai multor registre, la liniile aceleaşi magistrale, ieşirile acestora sunt de tip TSL, deci trebuie să existe o validare a ieşirii OE (Output Enable). Toate aceste semnale de control, deoarece se aplică la n celule, trebuie bufferate la intrarea în circuitul registru (pentru ca semnalele în exterior sunt fie comandate doar ca o singură unitate de sarcină de încărcare).

În Figura 3.71-c pe structura unui registru paralel de n biţi sunt adăugate semnalele de control. Semnalul de încărcare, LOAD, este semnalul de ceas, iar semnalul de ştergere este semnalul asincron $\overline{\text{CLEAR}}$. Activarea semnalului validare ieşire, $\overline{OE} = 0$, va comanda trecerea bufferului de ieşire TSL din starea de înaltă impedanţă în starea normală de funcţionare, deci ieşirea (la magistrală) este cuvântul înscris în registru $Q_{n-1}, Q_{n-2}, ..., Q_i, ..., Q_1, Q_0$.

Când $\overline{OE} = 1$, bufferul TSL nu este activat, la ieşirea registrului la magistrală

este cuvântul existent pe magistrală $OD_{n-1}OD_{n-2}\ldots OD_i\ldots OD_1OD_0$. Intrarea de date D a fiecărei celule bistabil D_i, $0 \leq i \leq n-1$, este conectată, printr-un MUX4:1, în funcţie de cuvântul de selectare EN=S_1S_2 la una din următoarele patru surse de date:

1. pentru EN = 00, $D_i = Q_i$, la fiecare impuls de ceas bistabilul se reîncarcă cu valoarea deja înscrisă;

2. pentru EN = 01, $D_i = 1D_i$, la fiecare impuls de ceas bistabilul se încarcă cu valoarea $1D_i$ de la sursa 1;

3. pentru EN = 10, $D_i = 2D_i$ la fiecare impuls de ceas bistabilul se încarcă cu valoarea $2D_i$ de la sursa 2;

4. pentru EN = 11 şi $\overline{OE} = 1$ $D_i = 0D_i$, la fiecare impuls de ceas bistabilul se încarcă cu valoarea OD_i existentă la ieşirea pe magistrală. Evident, că prin relaţia $S_1 \cdot S_2 \cdot \overline{OE} = 0$ bufferul TSL este în starea de înaltă impedanţă, deci ieşirea Q a bistabilului nu se aplică la ieşire pe magistrală;

În tehnologie MOS şi CMOS, uzual, registrele se realizează cu celule dinamice; astfel se obţine consum redus de putere şi densitate de integrare mărită. Structuri dinamice de latch D şi bistabil D sunt prezentate respectiv în Figurile 3.40-b şi 3.45-b.

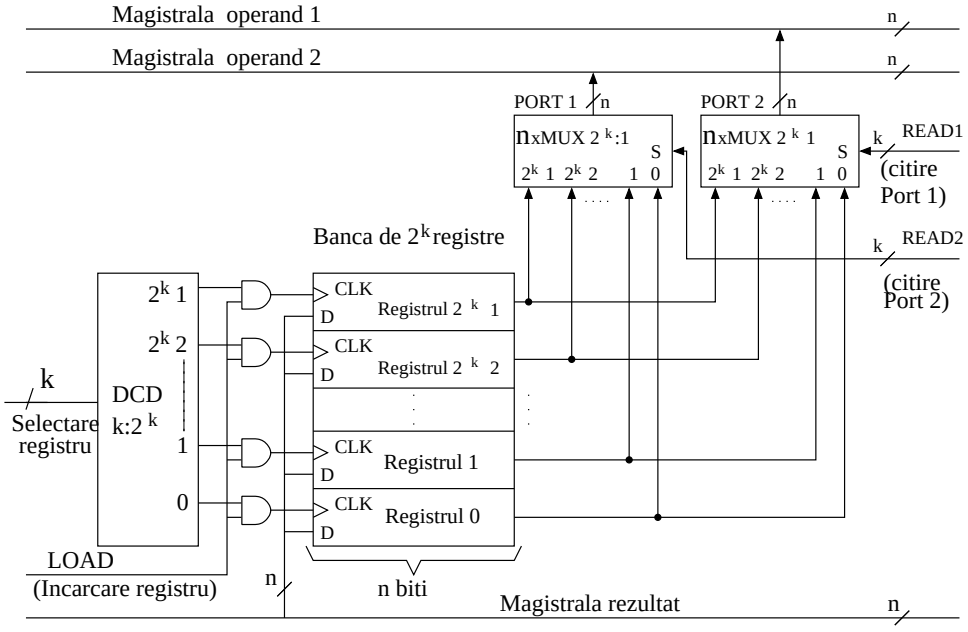


Figura 3.73 Organizarea unei bănci de 2^k registre cu lungime de cuvânt de n biţi, dublu port pe ieşire (cu evidenţierea decodificării pe intrare, pentru înscriere, şi selectării pe ieşire, pentru citire)

Pentru mărirea capacităţii de stocare, la mai mult de un cuvânt, un număr 2^k registre sunt grupate formând o **bancă de registre**; fiecare din cele 2^k registre putând fi selectat atât pentru înscriere cât şi pentru citire. În Figura 3.73 este structurată o bancă de 2^k registre, fiecare dintre registre având lungimea de n biţi, capacitatea totală totală este $(2^k \cdot n)$.

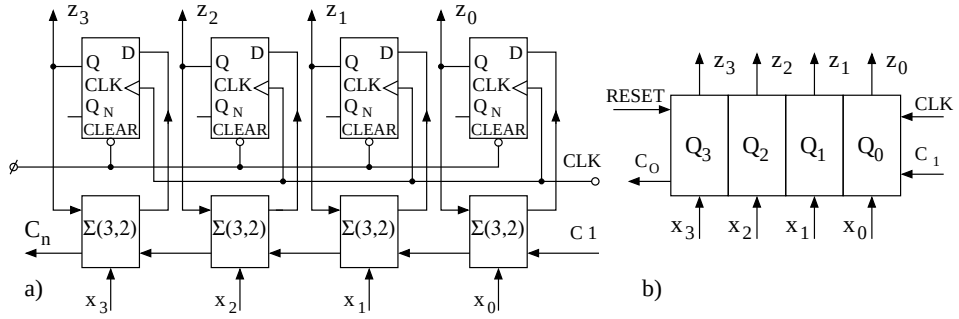
Cuvântul de n biţi de pe magistrala rezultat se va înscrie în acel registru pentru care semnalul de ceas devine activ, semnal format prin conjuncţia între semnalul de încărcare LOAD şi ieşirea de la DCDk:2^k, selectată prin cuvântul Selectare registru. Banca de registre este **dublu port** pentru a putea fi citite simultan două registre, adică pe fiecare din cele două magistrale 1 şi 2 se aplică conţinutul unui registru. Un grup de $n\times$MUX2^k:1, prin care este selectat unul din cele 2^k registre ca port de ieşire (conectare la magistrală), are organizarea similară ca în Figura 2.36 (acolo este o notaţie inversată, un grup de 2^n registre fiecare cu lungimea de k biţi).Această detaliere, a modului de operare asupra unei bănci de registre, completează descrierea căii de date prezentată în Figura 2.73.

3.5.2 Circuitul acumulator

Se obţine o structură hibridă printr-o joncţionare a celulelor bistabil ale unui registru paralel cu lungimea de n biţi cu celulele sumator complet $\sum(3,2)$ ale unui sumator modulo 2^n cu transport succesiv, Figura 3.74-a. Pe celula sumator complet cu ponderea 2^i, $0 \leq i \leq n-1$, a sumatorului se va suma: bitul x_i al cuvântului $X = x_{n-1}x_{n-2}\ldots x_i\ldots x_1x_0$, aplicat din exterior; bitul Q_i înscris în celula i a registrului şi bitul de transport C_{i-1}, de la celula sumator complet anterioară. Această celulă sumator complet, de pondere 2^i, va genera bitul sumă $s_i = C_{i-1} \oplus Q_i \oplus x_i$ şi bitul transport următor $C_i = Q_i \cdot x_i + Q_i \cdot C_{i-1} + x_i \cdot C_{i-1}$, care se aplică la celula sumator complet de rang $i+1$, ca bit de transport anterior; iar la aplicarea următorului impuls de ceas (t+1) bitul sumă calculat $s_i(t)$ se înscrie în celula i a registrului, deci $Q_i(t+1) = s_i(t)$. Extinzând la lungimea de n biţi, la impusul de ceas (t+1) în registru se înscrie suma aritmetică modulo 2 rezultată între: conţinutul anterior din registru, $Q(t) = Q_(n-1)(t)Q_{n-2}(t)...Q_i(t),...,Q_1(t)Q_0(t)$; cuvâtul aplicat din exterior, $X(t) = x_{n-1}(t)x_{n-2}(t),...,x_i(t),...,x_1(t)x_0(t)$ şi transporturile anterioare $C_{n-2}(t), C_{n-3}(t),...,C_{i-1}(t),...,C_1(t), C_0(t), C_{-1}(t)$.

În cazul în care valoarea iniţială a cuvântului în registru este zero $Q(0) = 0$, transportul anterior $C_{-1} = 0$, iar cuvântul extern X este constant în timp, în registru rezultă o sumare repetată a cuvântului extern, după k impulsuri de tact se obţine valoarea kX modulo 2^n. Apare evident faptul că circuitul acumulator este un automat, la care partea combinaţională este formată din celule sumator $\sum(3,2)$, de aceea, uneori, cuvântul de ieşire va fi notat ca un cuvân de stare cu $z_3z_2z_1z_0$.

Bazat pe această funcţionare, de sumare repetată modulo2^n, circuitul acumulator poate fi utilizat pentru modelarea funcţionării numărătorului sincron modulo2^n direct sau invers în oricare cod. De exemplu, pentru acumulatorul cu trei celule, n=3, pornind din starea iniţială $Q(0) = Q_2Q_1Q_0 = 000$ şi $C_{-1} = 0$, cu un cuvânt aplicat din exterior $X = 001$, se obţine o succesiune de numărare în sens direct, coloana a doua din Figura 3.74-c, corespunzătoare numărătorului sincron modulo 8 în cod binar natural. Modificarea funcţionării de numărător sincron modulo 8, în sens invers, în cod binar natural, se obţine pornind din starea iniţială $Q(0) = 0$ cu $X = x_2x_1x_0 = 111$



Nr de impuls de ceas	Starea initiala $z_2z_1z_0$=000 $x_2x_1x_0$=001 C_1=0	Starea initiala $z_2z_1z_0$=000 $x_2x_1x_0$=111 C_{+1}=0	Nr de impuls de ceas	Starea initiala $z_3z_2z_1z_0$=0000 $x_3x_2x_1x_0$=0101 c_1=0		
1	000+ 001 = 001	000+ 111 = 111	1	0000+ 0101 = 0101	9	1000+ 0101 = 1101
2	001+ 001 = 010	C_O=1 ← 111+ 111 = 110	2	0101+ 0101 = 1010	10	C_O=1 ← 1101+ 0101 = 0010 (3+5+5+5) modulo16=2
3	010+ 001 = 011	C_O=1 ← 110+ 111 = 101	3	1010+ 0101 = 1111	11	0010+ 0101 = 0111
4	011+ 001 = 100	C_O=1 ← 101+ 111 = 100	4	C_O=1 ← 1111+ 0101 = 0100 (5+5+5+5) modulo16=4	12	0111+ 0101 = 1100
5	100+ 001 = 101	C_O=1 ← 110+ 111 = 011	5	0100+ 0101 = 1001	13	C_O=1 ← 1100+ 0101 = 0001 (2+5+5+5) modulo16=1
6	101+ 001 = 110	C_O=1 ← 011+ 111 = 010	6	1001+ 0101 = 1110	14	0001+ 0101 = 0110
7	110+ 001 = 111	C_O=1 ← 010+ 111 = 001	7	C_O=1 ← 1110+ 0101 = 0011 (4+5+5+5) modulo16=3	15	0110+ 0101 = 1011
8	C_O=1 ← 111+ 001 = 000	C_O=1 ← 001+ 111 = 000	8	0011+ 0101 = 1000	16	C_O=1 ← 1011+ 0101 = 0000 (1+5+5+5) modulo16=0
c)	$f=f_{CLK}\cdot 1/8$ Numarator direct modulo8	$f=f_{CLK}\cdot 7/8$ Numarator invers modulo8	$f=f_{CIK}\cdot 5/16$ Numarator divizor cu coeficient fractionar			

Figura 3.74 Circuitul acumulator: a) structură de acumulator cu patru celule; b) reprezentare simbolică; c) analiza a trei exemple de numărător modelate pe bază de acumulator.

şi $C_{-1} = 0$, coloana a treia din figura 3.74-c.

Pentru modelarea unor numărătoare în alte coduri de numărare este necesar, uneori, să se ataşeze în exterior un decodificator, realizat cu porţi logice, pentru decodificarea unui anumit cuvânt de cod din registru şi, eventual, să se modifice valoarea cuvântului X aplicat din exterior(vezi problemele P3.73 şi P3.74). Pe un acumulator se poate modela orice proces care poate fi exprimat prin operaţii de: adunare, scădere, deplasare stânga/dreapta sau încărcare (forţare a unui cuvânt exterior).

De fapt, acumulatorul este un circuit de calcul pentru prefixe sumă modulo 2^n cu o funcţie generică $f = (X + X)modulo2^n$; de exemplu pentru $n = 3$ se pot scrie relaţiile

$$\begin{aligned}
&f_0 = (0 + 0) \text{ modulo } 8 = 0\\
&f_1 = (f_0 + X) \text{ modulo } 8 = X \text{ modulo } 8\\
&f_2 = (f_1 + X) \text{ modulo } 8 = 2X \text{ modulo } 8\\
&f_3 = (f_2 + X) \text{ modulo } 8 = 3X \text{ modulo } 8\\
&\quad\vdots\\
&f_7 = (f_6 + X) \text{ modulo } 8 = 7X \text{ modulo } 8\\
&f_8 = (f_7 + X) \text{ modulo } 8 = 8X \text{ modulo } 8 = f_0 = 0
\end{aligned}$$

sau relaţia generalizată pentru acumulator cu n celule se scrie

$$(\sum_{1}^{2^n} X) \text{ modulo } 2^n = 0 \tag{3.44}$$

Relaţia 3.44 arată că circuitul acumulator funcţionează ca un numărător de X-tuple pe un interval/(perioadă) de 2^n impulsuri de ceas. Altfel spus, pornind din starea iniţială $Q(0) = 0$ şi C_{-1}, pentru cuvântul X aplicat din exterior, acumulatorul va genera (se va umple) semnalul de transport următor C_O de un număr X ori pentru 2^n impulsuri de ceas aplicate. În Figura 3.74-c, coloanele 5 şi 7, este realizat cazul unui acumulator cu n=4, pentru $X = 5$; se observă că se generează de cinci ori semnalul $C_O = 1$ pe durata a 16 impulsuri de ceas, deci o multiplicare cu 5/16 a frecvenţei de ceas (Aceste impulsuri sunt reprezentate, comparativ cu cele generate de un multiplicator cu coeficient binar, în Figura 3.68-b; se observă că distribuţia lor în timp este mai uniformă decât cea obţinută de la un multiplicator binar). Cu circuitul acumulator se pot realiza multiplicatoare cu un coeficient fracţionar X/C unde numitorul poate fi diferit de puteri ale lui doi $C \neq 2^n$ [Oberman ′78]).

Pe baza acumulatorului se pot realiza circuite care calculează, în timp real, valorile unor funcţii (ridicarea la o putere, rădacină pătrată, log_2, funcţii trigonometrice de numărul de impulsuri N_{CLK}). Aceste valori sunt disponibile imediat ce se primeşte impulsul de intrare.

Exemplul 3.26 Să se realizeze o structură de circuit pe bază de acumulatoare care să calculeze pătratul numărului de impulsuri aplicate, N^2_{CLK}.

Soluţie. Notând cu S_n=N^2_{CLK} şi S_{n+1}=$(N_{CLK} + 1)^2$ se deduce relaţia recurentă

$$S_{n+1} = (N_{CLK} + 1)^2 = S_n + 2N_{CLK} + 1$$

Dacă se cunoaşte deja valoarea pătratul S_n, prin adunarea la această valoare a dublului număr de impulsuri aplicate, $2N_{CLK}$, plus 1, atunci la aplicarea următorului impuls de ceas

se generează valoarea S_{n+1}. Structurarea pe bază de acumulatoare care realizează acest calcul este prezentată în Figura 3.75-a. Acumulatorul $A_{cc}Q_2$ operează ca un sumator de acumulare cu $C_{-1} = 1$ deci un numărător în cod biar natural pentru numărul impulsurilor aplicate, N_{CLK}.

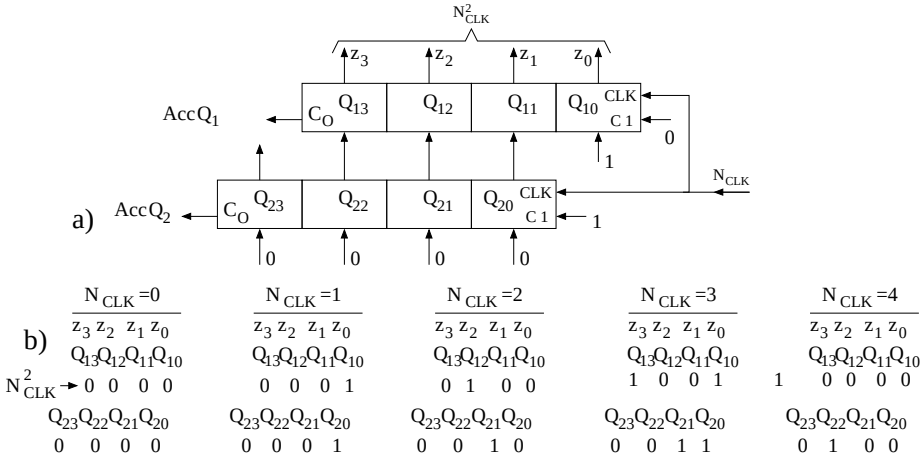


Figura 3.75 Calculul pătratului numărului de impulsuri recepţionate N^2_{CLK}: a) structură de circuit pe bază de acumulatoare; b) modificarea conţinuturilor acumulatoarelor pentru aplicarea succesivă a patru impulsuri de intrare, $N_{CLK} = 4$.

$A_{cc}Q_1$, pe ale cărui intrări se aplică dublul numărului de impulsuri deja recepţionate $2N_{CLK}$ (înmulţirea cu 2 se obţine prin deplasarea cu un rang spre stânga a celulelor lui $A_{cc}Q_2$ faţă de $A_{cc}Q_1$) plus 1 (este transportul anterior, $C_{-1} = 1$) la aplicarea următorului impuls prin sumare a $2N_{CLK} + 1$ la S_n (deja existentă în $A_{cc}Q_1$) va genera la ieşire codul binar $z_3\ z_2\ z_1\ z_0$ pentru valoarea lui $(N_{CLK} + 1)^2$. Analiza modificării conţinuturilor pentru 4 impulsuri aplicate succesiv în cele două acumulatoare este prezentată în Figura 3.75-b; pentru valori care necesită o exprimare pe mai mult de patru biţi se extinde numărul de celule pe acumulatoare.

3.5.3 Structura pipeline

Structura pipeline (conductă) este un suport hardware pentru a reduce timpul de procesare pe un CLC. Procesarea clasică a unui flux de date pe un CLC presupune o organizare ca cea prezentată în Figura 3.76. Un eşantion de date este înscris, prin activarea semnalului de ceas, în registrul de intrare R_1 şi aplicat pe intrările CLC, apoi la următorul semnal de ceas, datele procesate în CLC sunt stocate în registrul de ieşire R_2. Perioada minimă T_{CLK} trebuie să fie mai mare decât suma timpilor: τ_{pR}-timpul de propagare prin registrul de intrare (care este de fapt τ_{pCQ}), τ_{pCLC}-timpul de propagare prin CLC, τ_{SU}-timpul de stabilizare de la intrarea registrului de ieşire.

$$T_{CLK(min)} \geq \tau_{\mathrm{pR}} + \tau_{\mathrm{pCLK}} + \tau_{\mathrm{SU}} \tag{3.45}$$

Deoarece timpii de întârziere cauzaţi de registre (valori tipice $\tau_{\text{pR}} \approx 15 \div 30$ ns, $\tau_{\text{SU}} \approx 5 \div 15$ ns) mai pot fi reduşi doar de îmbunătăţiri tehnologice, apare evident faptul că mărirea ratei de procesare a fluxului de date se poate realiza numai printr-o micşorare a propagării τ_{pCLC} (care depinde de complexitatea reţelei CLC, pentru calcule de tip aritmetic τ_{CLC} poate avea valori în intervalul $150 \div 300$ ns). Uneori, funcţia de procesare pe CLC poate fi privită ca fiind compusă dintr-o succesiune de n subfuncţii, fiecare subfuncţie putând fi implementată respectiv pe câte un circuit independent $CLC_1, CLC_2, ..., CLC_n$ cu timpii de propagare $\tau_{\text{pCLC1}}, \tau_{\text{pCLC2}}, ..., \tau_{\text{pCLCn}}$. Conform acestei abordări, se poate structura o înseriere a acestor n circuite CLC_i, interfaţate prin câte un registru (registru pipe), pentru a aplica datele de intrare respectiv pentru a stoca datele de ieşire — aceasta fiind structurarea de tip pipeline având n etape de procesare, Figura 3.76-b.

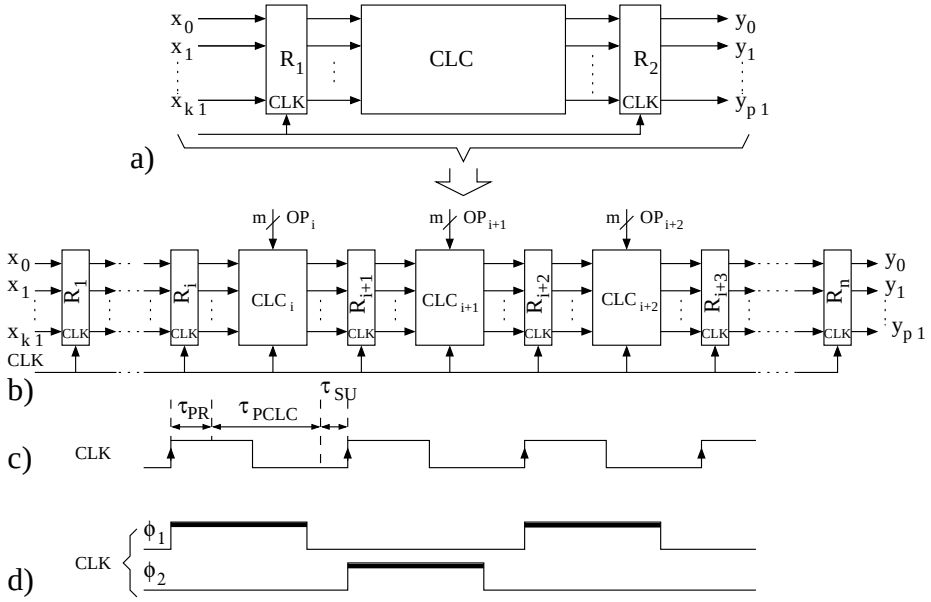


Figura 3.76 Organizarea de tip pipeline: a) structurarea pentru o procesare de tip non-pipe (clasică); b) structura unui pipe cu n etape de procesare; semnale de ceas pentru sincronizarea în pipe (c) cu o singură fază (pe front) şi (d) cu două faze Φ_1 şi Φ_2 (pe palier)

Înscriind un set de date, $x_{k-1}, x_{k-2}, ..., x_1, x_0$, în primul registru pipe, R_1, pe frontul activ al semnalului de ceas, acestea vor fi apoi procesate şi stocate succesiv în toate celelate $(n-1)$ etape din pipe, iar după $(n-1)$ tacte se obţin datele de ieşire, $y_{p-1}, y_{p-2}, ..., y_1, y_0$. Dacă fluxul de date aplicat pe intrare este continuu, după intervalul de timp n tacte de ceas **latentă în pipe (timpul de umplere al conductei)**, la fiecare următor tact se obţine un set de date prelucrate la ieşire. Perioada minimă a impulsului de tact se calculează tot cu relaţia 3.45 dar, de data aceasta, se consideră timpul de propagare maxim (care corespunde subfuncţiei cu timpul cel mai lung de procesare).

$$\begin{aligned}\tau_{\text{CLK(min)}} \geq \quad & \tau_{\text{pR}} + max\{\tau_{\text{pCLC1}}, \tau_{\text{pCLC2}}, ..., \tau_{\text{pCLCn}}\} + \tau_{\text{SU}} \\ & < \tau_{\text{pR}} + \tau_{\text{pCLC}} + \tau_{\text{SU}}\end{aligned} \tag{3.46}$$

şi în ipoteza că toate registrele pipe sunt identice, deci au aceleaşi valori pentru τ_{SU} şi τ_{pR}.

În raport cu procesarea clasică (non-pipe) la procesarea de tip pipeline se obţine o creştere de viteză egală cu raportul timpilor calculaţi după relaţiile 3.46 şi 3.47, dar această creştere de viteză se obţine numai după ce primul set de date a ieşit din ultima etapă iar fluxul de date la intrarea în primă etapă este continuu. Dacă fluxul de date nu este continuu, cu întreruperi frecvente, deci la fiecare reumplere a pipe-ului se consumă un timp egal cu latenţă în pipe, atunci se poate ajunge la un timp mediu de procesare mai mare decât la procesarea non-pipe!

Relaţia de timp 3.46, între perioada tactului de ceas şi timpii dintr-o etapă a pipe-ului, este explicitată grafic în Figura 3.76-c, când semnalul de ceas este activ pe frontul pozitiv. Când registrele pipe sunt realizate din latch-uri D comanda se efectuează pe palier cu două semnale Φ_1 şi Φ_2, defazate şi nesuprapuse, Figura 3.76-d; iar datele în pipe curg, alternativ, din etape cu număr impar în etape cu număr par, comandate alternativ de cele două faze Φ_1 şi Φ_2.

O uniformizare a implementării unei implementări de tip pipe se poate obţine dacă, în fiecare etapă, pentru CLC-ul este utilizată o aceeaşi structură programabilă care se programează pentru operaţia specifică OP_i cerută în etapa i de procesare printr-un cuvânt de control cu m biţi. Un astfel de CLC programabil poate fi un MUX2^m:1, programat să genereze una din cele 2^{2^m} funcţii de m variabile. Într-o etapă din pipe se poate scădea timpul de propagare pâna la patru niveluri logice, un AND-OR pe CLC şi încă două niveluri dacă registrul pipe este un registru latch. Totuşi, există o variantă prin care timpul se poate reduce chiar până la două niveluri de propagare, care acoperă atât propagarea cât şi memorarea, dacă se utilizează latch-uri Early, Figura 3.41.

Organizarea de tip pipeline este modul comun pentru structurarea căii de date a microprocesoarelor actuale. Execuţia unei instrucţiuni (ciclul de execuţie) se descompune în n subcicluri, deci calea de date se granulează în n etape succesive de procesare. Dacă se reuşeşte alimentarea continuă cu instrucţiuni în calea de date atunci viteza de procesare a microprocesorului creşte (teoretic!) de n ori faţă de un procesor o cu cale de date non-pipe.

3.5.4 Registrul de deplasare

Registrul de deplasare, la fel ca şi registrul paralel, este o structură simplă ce se obţine prin înserierea de celule bistabil, uzual de tip D, de unde şi denumirea de **registru de deplasare serie** sau, mai frecvent, **registru serial**.

În Figura 3.77-a este prezentată organizarea unui registru, prin înserierea a patru celule bistabil D, precum şi funcţionarea sa la aplicarea pe intrarea serie a cuvântului $X = 1011$. Considerând iniţial conţinutul registrului $Q_4Q_3Q_2Q_1 = 0000$ la primul impuls de ceas, pe front pozitiv, bitul cel mai semnificativ, MSB, este introdus în celula Q_1, deci conţinutul registrului serial este $Q_4Q_3Q_2Q_1 = 0001$. La al doilea impuls de ceas, conţinutul din prima celulă, $Q_1 = 1$, este transferat în celula a doua, deci $Q_2 = 1$, iar în prima celulă se înscrie valoarea bitului următor după MSB, deci

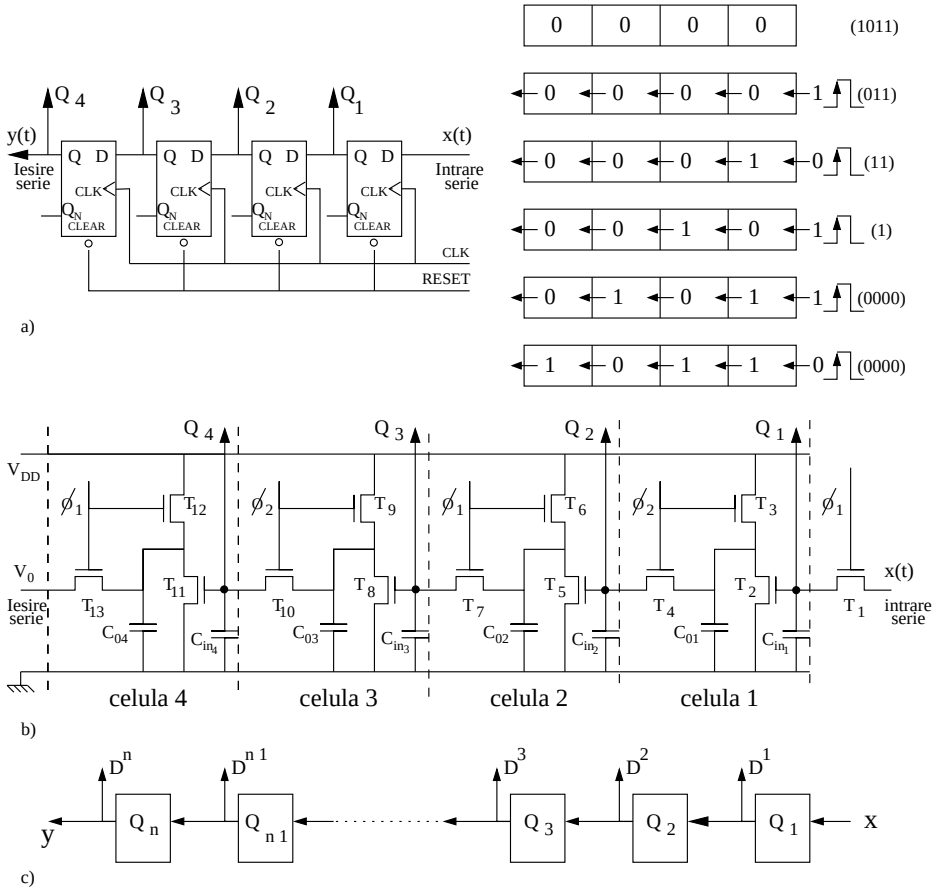


Figura 3.77 Registru de deplasare: a) structurarea şi funcţionarea unui registru de deplasare cu patru celule ($n = 4$); b) registru de deplasare, $n = 4$, pe bază de inversoare dinamice cu raport; c) simbol pentru registru de deplasare compus din n celule (ca reţea digitală de întârziere)

$Q_1 = 0$, rezultând conţinutul registrului $Q_4Q_3Q_2Q_1 = 0010$. La al patrulea impuls de ceas întreg cuvântul 1011 a fost deja introdus în registru iar conţinutul acestuia este $Q_4Q_3Q_2Q_1 = 1011$; valoarea MSB este disponibilă la ieşirea serie. Începând cu al cincilea impuls de ceas, transferul serial celulă după celulă se continuă, bitul din celula Q_4 se pierde, fiind înscris cu bitul care a fost în celula Q_3, la fel si în celula Q_1, se înscrie valoarea $Q_1 = 0$, deoarece cuvântul $X = 1011$ aplicat pe intrare a fost deja introdus. Pe tactele de ceas 4, 5, 6, 7 pe ieşirea serie se obţine cuvântul Y care este identic cu cuvântul X aplicat pe intrarea serie pe tactele 1, 2, 3, 4, deci după un interval de patru tacte, la al optulea tact conţinutul registrului va fi din nou $Q_4Q_3Q_2Q_1 = 0000$. Pe un registru serial de n celule transferul unui bit, aplicat pe intrare şi până la iesire, se face pe durata a n perioade de ceas, iar transferul până la o iesire intermediară i, $1 \leq i < n$, necesita i perioade de ceas. Formal, transferul prin registrul serial poate fi exprimat în felul următor:

$$\left.\begin{array}{l} Q_i(t+T) = Q_{i-1}(t) \times CLK \\ Q_1(t+T) = x(t) \cdot CLK \end{array}\right\} \Rightarrow Q_i(t+iT) = x(t) \tag{3.47}$$

unde $x(t)$ este valoarea bitului, din cuvântul X, aplicat pe intrarea serie la momentul t.

Fizic, transferul corect prin registrul serial, pe durata unui tact, bitul de la celula Q_i la celula Q_{i+1}, poate fi realizat numai dacă timpul de menţinere de la celula următoare, Q_{i+1}, este mai mic decât timpul de propagare pe celula anterioară, Q_i, $\tau_{\mathrm{H(Q_{i+1})}} < \tau_{\mathrm{pCQ(Q_i)}}$. Dacă această condiţie nu este îndeplinită înseamnă că semnalul pe intrarea de date al celulei Q_{i+1}, care este semnalul de ieşire de la celula anterioară, se modifică în fereasta de decizie Δ (relaţia 3.23, Figura 3.46 a şi b). La bistabile este îndeplinită relaţia $\tau_{\mathrm{H}} < \tau_{\mathrm{pCQ}}$ deci, dacă registrul serial se realizează cu acelaşi tip de registre, atunci este realizată condiţia anterioară de transfer corect. Dacă celulele registrului serial sunt latch-uri, care sunt transparente pe durata palierului activ de ceas, atunci un bit este transferat continuu înspre ieşire din celulă în celulă cât timp se menţine activ palierul semnalului de ceas.

Pentru implementările integrate, mai ales pentru număr de celule de valoare ridicată, sunt recomandate structurile de registre serie pe bază de celule dinamice.Toate astfel de registre dinamice se comandă prin două semanle de ceas Φ_1, Φ_2, Figura 3.76-d, nesuprapuse şi aplicate alternativ. În Figura 3.77-b este prezentat un registru serie cu patru celule, fiecare celulă fiind un invers dinamic cu tranzistoare nMOS. Pentru micşorarea puterii disipate tranzistoarele de sarcină, T_3, T_6, T_9 si T_{12}, sunt în conducţie numai pe durata palierului Φ_2. Pe durata palierului activ Φ_1, valoarea tensiunii de intrare, corespunzătoare bitului $x(t)$ aplicat pe intrarea serie, este transferată, prin tranzistorul de trecere T_1, condensatorului C_{in1}, de pe intrarea primei celule 1. Valoarea tensiunii pe C_{in1}, după anularea semnalului Φ_1, nu trebuie să iasă din intervalul ΔV_H (ori ΔV_L) până la activarea semnalului Φ_2. La activarea lui Φ_2, tranzistorul T_3 intră în conducţie, condensatorul C_{01} de pe iesirea primului inversor se încarcă la tensiunea corespunzătoare care, prin tranzistorul de trecere T_4, comandat în conducţie tot de Φ_2, este transferată ca tensiune de încărcare şi pe condensatorul C_{in2} de pe intrarea celulei a doua. Când semnalul Φ_1 devine din nou activ tensiunea de pe C_{02}, corespunzătoare tensinii de pe C_{in2}, este transferată pe condensatorul C_{in3} de la intrarea celulei a treia şi de asemenea, tensiunea corespunzătoare următorului bit de pe intrarea $x(t+T)$ este transferată condensatorului C_{in1}, de la intrarea primei

celule. Sunt comandate succesiv: întâi toate celulele cu număr impar cu faza Φ_1, apoi toate celulele cu număr par cu faza Φ_2.

Registrul serie dinamic prezentat mai este referit şi ca **registru serie dinamic cu raport**. Termenul de raport specifică faptul că în layout-ul inversorului trebuie realizată o anumită valoare a raportului între coeficientul W/L al tranzistorului inversor şi coeficientul W/L al tranzistorului de sarcină. Numai realizând un anumit raport tensiunea de ieşire V_{0L} a etajului inversor, de pe condensatorul C_0, care se transferă prin tranzistorul de trecere ca tensiune de intrare la inversorul următor, poate încărca C_{in} la o valoare în intervalul ΔV_L; tensiunea corespunzătoare pe intrarea inversorului este în intervalul ΔV_H. Se pot realiza celule de inversor dinamic şi fară raport. La un **inversor fară raport** un semnal de ceas comandă tranzistorul de sarcină şi simultan tranzistorul de trecere care încarcă capacitatea de pe propria intrare, şi nu tranzistorul de trecere care încarcă intrarea inversorului următor cum este la structura cu raport; încercaţi să desenaţi această structurare. O structură de registru serial CMOS dinamic , se poate obţine simplu prin înserierea de celule ca cea prezentată în Figura 1.54-b; trebuie ca cele două porţi de transmisie corespunzătoare la două inversoare succesive să fie comandate una cu Φ_1/Φ_2 iar cealaltă cu Φ_2/Φ_1.

Registrul serie, în funcţionarea corectă, poate realiza întârzieri controlate, de multiplii ai perioadei semnalului de ceas, între semnalul aplicat pe intrarea serie şi momentul obţinerii acestuia la ieşirile diferitelor celule componente, deci poate fi considerat echivalentul, în digital, al liniei de întârziere analogică cu întârzieri multiple. Daca se exprimă operaţia de întârziere cu o perioadă de ceas, a unui bit b printr-o celulă a registrului, cu simbolul (operatorul) Db, iar operaţia de deplasare a bitului b prin i celule ale registrului cu D^ib (o aplicare de i ori a operatorului $D(D(D...D(b))...)) = D^ib$) se poate considera o reprezentare generică pentru un registru serie cea din Figura 3.77-c. Bitul b aplicat pe intrare devine ieşirea Q_i după i operaţii de întârziere:

$$Q_i = D^ib, 1 \leq i \leq n,\ D^0b = Ib = b \tag{3.48}$$

unde I este operatorul unitar (identic).

Pentru un cuvânt X cu lungimea de n biţi $X = b_nb_{n-1}...b_2b_1$ introdus bit după bit, începând cu cel mai semnificativ, b_n, prin aplicarea succesivă a unui număr de n tacte, într-un registru de deplasare cu n celule, conform relaţiei 3.47 se poate adopta o exprimare polinomială (după puterile lui D)

$$X = b_nD^n + b_{n-1}D^{n-1} + \ldots + b_2D^2 + b_1D^1 \tag{3.49}$$

conţinutul celulelor bistabil fiind: $Q_n = b_n$, $Q_{n-1} = b_{n-1}$,..., $Q_2 = b_2$ si $Q_1 = b_1$. Deoarece tactul de ceas se aplică sincron la toate celulele, analiza conţinutului registrului, valorile Q_i, se consideră numai pe durata dintre fronturile active de ceas (interval numit uneori **"dit"**).

Registru, intuitiv, poate fi considerat ca o "lupă" sub care intră şirul X în devenire spre şirul Y. În aceată abordare, registrul de deplasare serie poate fi utilizat pentru o implementare alternativă a automatelor de identificare a anumitor secvenţe.

Exemplul 3.27 Automatul cu graful de tranziţie din Exemplul 3.3 să se implementeze pe bază de registre de deplasare.

Soluţie. Când în şirurile de biţi de intrare x_1, x_0, intrate sub "lupa" a două registre de deplasare, se identifică secvenţa de perechi $00, 00, 11, 10$ şi se generează ieşirea $y = 1$.

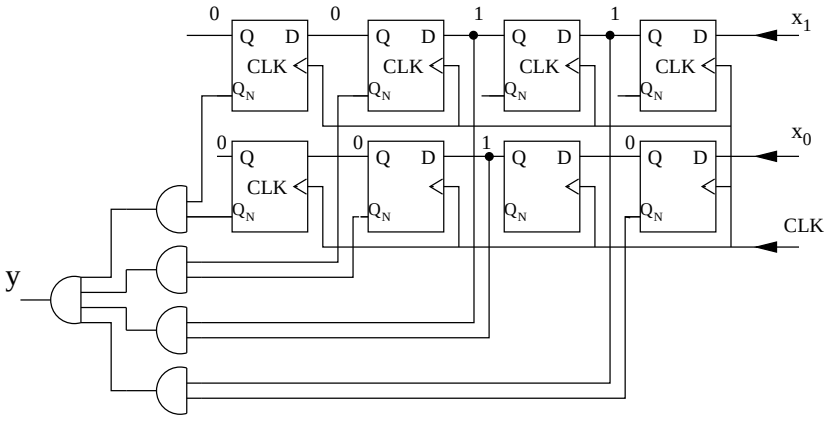


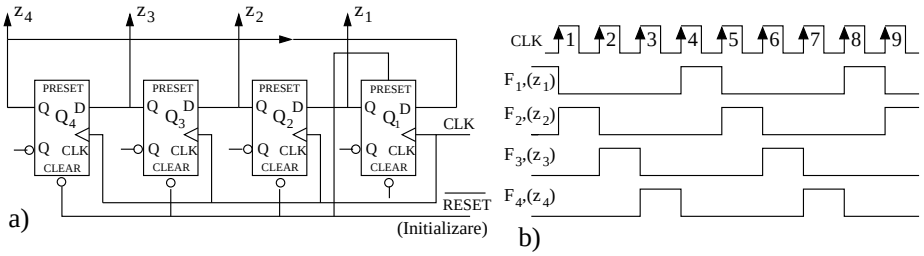
Figura 3.78 Modalitate alternativă, pe bază de registru de deplasare, pentru un automat de identificare de secveţe.

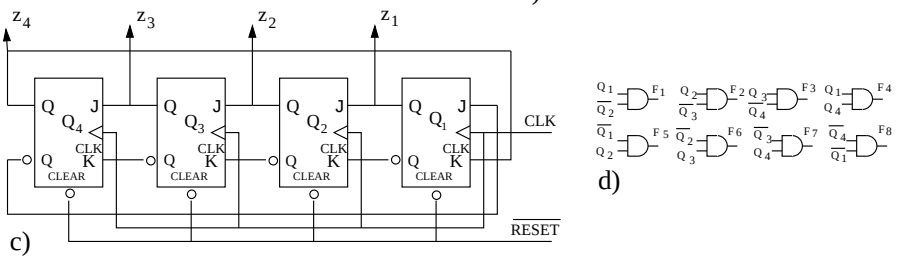
Implementarea corespunzătoare este cea din Figura 3.78; două registre serie de patru celule, unul pentru şirul intrări x_1, altul pentru şirul intrări x_2, iar ieşirile lor se compară permanent pe o reţea de porţi AND. În raport cu implementarea "clasică" de automat, unde este necesar un registru de stare cu trei celule plus partea de CLC, această implementare necesită mai multă circuistică dar, în schimb, se elimină efortul de sinteză pentru automat, structurarea fiind intuitivă şi imediată.

Registru inel. Structura de registru inel se obţine dintr-un registru de deplasare la care ieşirea serie se conectează la intrarea serie, deci se închide o buclă de reacţie în jurul registrului de deplasare. La un registru inel cuvântul iniţial, nu este pierdut prin deplasarea sa ci, este recirculat în interiorul inelului. Exprimarea formală a transferului ciclic se obţine, prin adaptarea relaţiilor (3.47) la existenţa reacţiei, în felul următor:

$$Q_i(t+T) = Q_{i-1}(t) \cdot CLK, \quad Q_1(t+T) = Q_n(t) \cdot CLK. \tag{3.50}$$

O structură de registru inel, compus din patru celule bistabil D, este prezentată în Figura 3.79-a. În general, într-un registru inel se forţează iniţial, prin intrările asincrone CLEAR şi PRESET ale celulelor, un cuvânt care are numai bitul cel mai puţin semnificativ egal cu 1, restul biţilor sunt zerouri. Prin deplasarea acestui cuvânt bitul cu valoarea 1 va ocupa toate cele n poziţii din registru, iar după n impulsuri de tact bitul cu valoarea 1 ajunge în poziţia iniţială. Această deplasare liniară poate fi asociată cu funcţionarea unui numărător modulo n, după n impulsuri se ajunge la acelaşi conţinut în inel, de unde şi denumirea, uneori, de **numărător în inel**. Pentru registrul inel cu patru celule succesiunea de cuvinte din inel este: 0001, 0010, 0100, 1000, 0001,.... Registrul inel este de fapt un automat Moore, starea următoare se calculează din starea prezentă cu ajutorul relaţiilor (3.50) care de fapt sunt funcţiile de excitaţie (pentru uniformitate, în tratare ca automat, cuvântul de stare este notat şi prin $z_4z_3z_2z_1$.





Impuls de ceas	z_4	z_3	z_2	z_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1
4	1	1	1	1
5	1	1	1	0
6	1	1	0	0
7	1	0	0	0
8	0	0	0	0
9	0	0	0	1
10	0	0	1	1
11	0	1	1	1

e)

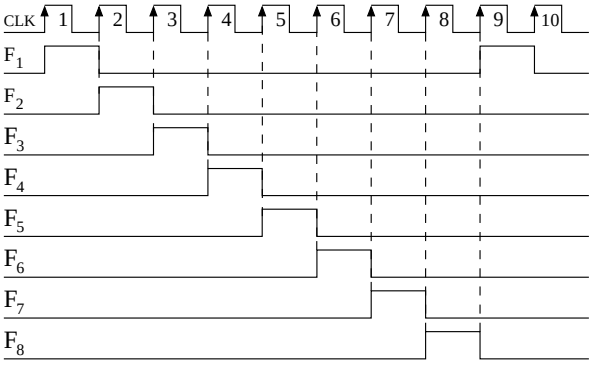


Figura 3.79 Registrul inel: a) structură de registru inel cu patru celule ($n = 4$); b) diagrama semnalelor de faze generate de un registru inel cu $n = 4$; c) structură de numărător Johnson cu $n = 4$ şi succesiunea normală a cuvintelor în procesul de numărare (e); d) decodificatorul de faze pentru un numărător Johnson cu $n = 4$ şi diagrama corespunzătoare pentru semnalele de fază (f).

Nu se recomandă utilizarea unui numărător în inel în raport cu un numărător clasic în cod binar, deoarece primul necesită n celule bistabil pe când al doilea numai $log_2 n$ celule bistabil. Totuşi, numărătorul inel poate fi utilizat pentru producerea dintr-o secvenţă de n impulsuri de ceas a unui număr de n semnale defazate şi nesuprapuse fiecare cu frecvenţa $1/n$, adică pentru realizarea unui **generator de faze**. Registrul inel prin modul său de funcţionare este un generator de faze, fiecare bit z_i, $1 \leq i \leq n$, al cuvântului de stare este o fază F_i; în Figura 3.79-b sunt prezentate fazele generate de un registru inel cu $n = 4$. Uneori, în registrul inel este recirculat un compact de m 1-uri în interiorul cuvântului de n biţi. Pentru această funcţionare se pot genera cele n faze printr-un sistem de porţi AND cu două intrări; poarta care generează o fază la un moment dat are o intrare conectată la celula care indică primul bit 1 din compact, iar cealaltă intrare la celula care indică ultimul bit din compact.

Uzual, un generator de n faze se structurează dintr-un numărător modulo n care comandă un decodificator cu $\lceil log_2 n \rceil$ intrări şi n ieşiri; ieşirea i a decodificatorului fiind faza F_i. Dar, deoarece ieşirile numărătorului nu comută simultan, chiar şi la un numărător sincron!, iar decodificatorul poate introduce hazard, în ieşirile decodificatorului/(faze) se produc glitch-uri. Pentru eliminarea glitch-urilor ieşirile decodificatorului sunt introduse intr-un registru, comandat pe încărcare cu un semnal complementat faţă de semnalul de ceas care este aplicat numărătorului, deci semnalele de fază de la ieşirea registrului sunt "curate" dar şi generate cu întârziere faţă de numărul corespunzător din numărător. Evident că, în raport cu această structurare de generator de faze, simplitatea generatorului de faze cu registru inel este preferată.

Numărătorul în inel, analizat ca automat, utilizează din cele 2^n stări posibile doar n stări, restul de $(2^n - n)$ stări sunt ilegale pentru funcţionarea sa. Dacă automatul ajunge accidental, de exemplu, la aplicarea tensiunii sau datorită zgomotului, intr-o stare ilegală poate rămâne un timp indefinit în starea respectivă, sau se realizează un ciclu între stări ilegale pe care le parcurge, la fel, indefinit. De exemplu, pentru un numărător în inel cu patru celule ($n = 4$), modulo 4, care are numai 4 stări normale, la funcţionarea normală se parcurge ciclul compus din următoarea succesiune de cuvinte de stare: 0001, 0010, 0100, 1000, 0001, ...Între restul de $(2^4 - 4 = 12)$ stări ilegale, când una din acestea apare accidental în numărătorul în inel, se poate stabili unul din următoarele 5 cicluri (care reprezintă o funcţionare anormală):

1. 0000, 0000, 0000, 0000, ...

2. 0101, 1010, 0101, 1010, ...

3. 0011, 0110, 1100, 1001, 0011, 0110, ...

4. 0111, 1110, 1101, 1011, 0111, 1110, ...

5. 1111, 1111, 1111, 1111, ...

Pentru eliminarea funcţionării anormale se utilizează, în sinteză, abordarea de risc minim, Figura 3.30-b, adică din oricare stare ilegală se dirijează tranziţii, într-un număr finit de tacte, spre o stare legală. Practic, numărătorul în inel cu $n = 4$ se realizează cu un circuit de autocorecţie care este o poartă NOR cu trei intrări; la intrările porţii NOR se aplică biţii z_3, z_2, z_1 ai cuvântului de stare iar ieşirea porţii

se introduce, ca semnal de reacţie, la prima celulă. Deci reacţia nu se mai realizează ca în ecuaţia (3.50) ci prin ecuaţia

$$Q_1(t+T) = \overline{(Q_3(t) + Q_2(t) + Q_1(t))} \times CLK \tag{3.51}$$

De exemplu, dacă în numărător apare accidental cuvântul de stare 1011, pe următoarele două tacte, deoarece Q_1 este produs permanent de poarta NOR, stările vor fi 0110, 1100, iar la al treilea tact se înscrie starea normală 1000,deci intră în ciclul de funcţionare normală (vezi problema P3.78). În general, un numărător în inel modulo n cu autocorecţie utilizează o poartă NOR cu $(n-1)$ intrări, iar corectarea se realizează în maximum $(n-1)$ tacte.

Numărătorul Johnson (sau Moebius). Acest tip de numărător este tot un registru inel cu particularitatea că reacţia de la celula Q_n la celula Q_1 se complementează, adică se culege de pe Q_N. O structură de numărător Johnson cu patru celule, pe bază de bistabile JK, este prezentată în Figura 3.79-c. Funcţiile de excitaţe pentru toate celulele rămân acelaşi ca şi la registrul inel, $wJ_i = Q_{i-1}$, $wK_i = Q_{N(i-1)}$, în afară de prima celulă unde prin reacţia inversată se introduce un defazaj de 180°, $wJ_1 = Q_{Nn}$, $wK_1 = Q_n$; de fapt, şi la implementarea cu bistabile JK celulele au o funcţionare de bistabil D, deoarece intrările fiecărei celule sunt comandate în opoziţie (de la Q şi Q_N ale celulei anterioare). Spre deosebire de un registru inel cu n celule, care generează în ciclul său n cuvinte de stare, numărărorul Johnson cu n celule poate avea $2n$ stări. Denumirea şi de numărător Moebius este datorită asemanării care se face cu bucla Moebius. Această buclă se poate confecţiona dintr-o bandă de hârtie care, înainte de a se uni (şi lipi) cele două capete, se răsuceşte o dată (încercaţi). Pornind cu un creion, dintr-un punct al buclei Moebius, drumul parcurs până se ajunge în acelaşi punct este dublu faţă de lungimea benzii deoarece se parcurg ambele feţe intr-un traseu continuu.

Pentru numărătorul Johnson, cu $n = 4$, din figură prin activarea semnalului $\overline{RESET} = 0$ cuvântul înscris va fi $z_4z_3z_2z_1 = 0000$. La primul impuls de ceas în celula Q_1 se înscrie 1 deoarece, prin reacţia inversată, intrările de date au valorile $wJ_1 = Q_{N4} = 1$ şi $wK_1 = Q_4 = 0$. Se continuă înscrierea în Q_1, a valorii 1 şi pe următoarele trei impulsuri de ceas, când se ajunge în numărător la starea $z_4z_3z_2z_1 = 1111$, deci Q_4 este înscris în starea 1. La al cincilea impuls de ceas în Q_1 se înscrie valoarea 0, deoarece $wJ_1 = Q_{N4} = 0$ şi $wK_1 = Q_4 = 1$, şi se continuă înscrierea în 0 în această celulă şi pentru impulsurile 6, 7, 8 când se ajunge ca celula Q_4 să comute din nou în 0, conţinutul numărătorului este $z_4z_3z_2z_1 = 0000$, deci ciclul în numărător se reia, următoarele patru impulsuri vor înscrie iarăşi Q_1 în starea 1; această funcţionare este sintetizată în tabelul din Figura 3.79-e. Ciclul de numărare este $2 \times 4 = 8$.

Cel mai simplu numărător Johnson este cel cu $n = 1$, care se obţine printr-o reacţie inversată în jurul unui element de întârziere, bistabil D, adică bistabilul de tip T, Figura 3.50-b. Impulsurile de ceas aplicate sunt divizate cu doi, deci un ciclu de numărare egal cu 2. Dar se pune întrebarea: nu se pot realiza numărătoare Johnson modulo un număr impar? Răspunsul este afirmativ. Se pot realiza şi pentru modulo $(2n-1)$ dacă prin sinteza, ca automat, se elimină una din stări, în general starea al cărui cod este format numai din 1 (vezi problema 3.75).

Aplicatia cea mai eficientă pentru numărătorul Johnson este, ca şi a numătorului în inel, cea de generator de faze. Dar spre deosebire de numărătorul în inel, unde

fazele sunt direct cele n ieşiri din celulele bistabil, la numărătorul Johnson cele $2n$ faze se generează prin n porţi AND cu două intrări; în Figura 3.79-d sunt indicate conexiunile la intrarea porţilor AND2 pentru $n = 4$, iar în figura 3.79-f diagrama de semnale pentru fazele generate.

Ca automat, numărătorul Johnson utilizează în ciclul său normal de funcţionare numai $2n$ stări din totalul de 2^n; deci un număr de $(2^n - 2n)$ stări ilegale/anormale. Abordând din punct de vedere al riscului minim, Figura 3.30-b, trebuie realizat un circuit de autocorectare care va forţa,din oricare stare ilegală după un număr de maximum $(n-1)$ tacte, înscrierea stării legale $00\ldots01$. Stările normale de funcţionare sunt numai cele cuprinse in următorul ciclu: $00\ldots00$, $00\ldots01$, $00\ldots11$, $\ldots$, $01\ldots11$, $11\ldots11$, $11\ldots10$, $11\ldots00$, $\ldots$, $10\ldots00$, $00\ldots00$, $00\ldots01$, $\ldots$. Oricare cuvânt de stare anormală, în care poate ajunge accidental numărătorul, poate fi scris numai sub forma $\times\ldots\times10\times\ldots\times$, care după maximum$(n-2)$ tacte de ceas va fi deplasat în numărător în poziţia $10\times\ldots\times$, iar după încă n tacte (prin reacţia inversată) în poziţia $0\times\ldots\times0$. În această poziţie cuvântul de stare, prin "amprenta" lui, cu două zerouri extreme, pe z_n şi z_1, poate fi detectat printr-o poartă NOR, $(\overline{Q_n + Q_1} = 1)$, iar la următorul tact ieşirea porţii NOR, care este 1, forţează în numărătorul Johnson înscrierea cuvântului (normal) $00\ldots01$ (vezi problema 3.79).

3.5.5 Registrul serie-paralel

Cu n bistabile D se poate realiza un registru serie sau se poate realiza un registru paralel, dar se poate realiza un registru care să aibă funcţiunile amândurora? Răspunsul este afirmativ, chiar mai mult, ca registru serie , deplasarea poate fi bidirecţională, stânga/ dreapta, şi nu numai unidirecţională. Pentu a implementa toate aceste funcţiuni într-un singur circuit, referit ca registru serie-paralel, celula bistabil trebuie inzestrată cu un comutator selectabil (multiplexor) pe intrarea de date pentru a se putea programa conexiunile necesare în funcţie de programarea funcţiunii dorite. O astfel de celulă din poziţia i-a a registrului, cu un MUX4:1 pe intrarea de date, a fost prezentată în Figura 3.72-c, cele patru surse de date fiind: două surse (independente) $2D_i$, $1D_i$, de încărcare paralelă; o sursă, $0D_i$, de încărcare de pe magistrala de ieşire şi o sursă, Q_i, pentru încărcarea cu propria-i dată a celulei.

În organizarea de registru serie-paralel, cu partu celule, din Figura 3.80-a fiecare celulă este înzestrată cu un MUX4:1. (Notaţiile sunt cele corespunzătoare circuitului registru univerasl 74xx194 care va fi utilizat în problemele ataşate acestui capitol). Cele patru intrări, selectate cu semnalele S_1S_0, sunt:

– $(S_1S_0 = 00)$ pentru menţinerea nemodificată a conţinutului registrului, fiecare celulă se reîncarcă la aplicatea impulsului de ceas cu propria dată:

$$w_Aw_Bw_Cw_D = Q_AQ_BQ_CQ_D$$

($w_Aw_Bw_Cw_D$ este notaţia de la automate corespunzătoare stării următoare);

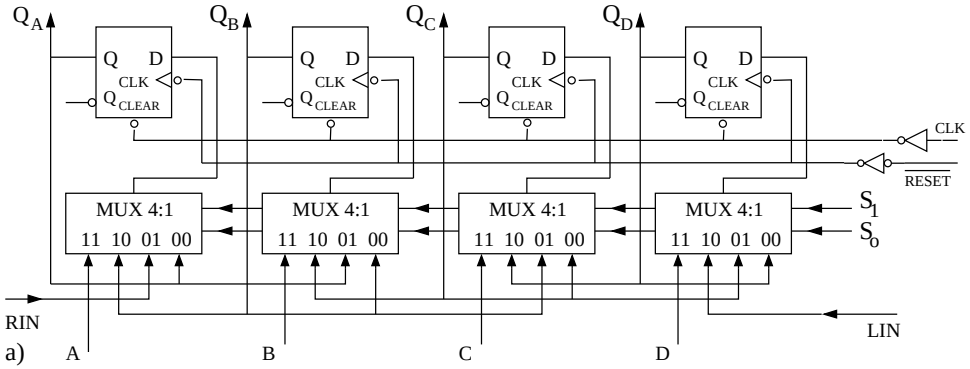
– $(S_1S_0 = 01)$ pentru deplasare conţinutului registrului cu o poziţie spre dreapta, fiecare celulă se încarcă, la aplicarea impulsului de ceas, cu conţinutul celulei vecine din stânga în felul următor: $w_B = Q_A$, $w_C = Q_B$, $w_D = Q_C$, iar $w_A = RIN$. Intrarea RIN este intrarea serie pentru deplasare spre dreapta, la aceasta se aplică câte un bit la fiecare impuls de ceas; această intrare poate fi

conectată la ieşirea serie a unui circuit registru similar, situat în stânga, atunci când se lucrează cu cuvinte multiplu de patru. Ieşirea serie, pentru deplasarea spre dreapta, este Q_D;

– $(S_1S_0 = 10)$ pentru deplasarea conţinutului registrului cu o poziţie spre stânga, fiecare celulă se încarcă, la aplicarea impulsului de ceas, cu conţinutul celulei vecine din dreapta în felul următor: $w_A = Q_B$, $w_B = Q_C$, $w_C = Q_D$, $w_D = LIN$. Intrarea serie pentru deplasarea stânga este LIN iar ieşirea serie este Q_A;

– $(S_1S_0 = 11)$ pentru încărcarea fiecărei celule cu o dată independentă în felul următor: $w_A = A$, $w_B = B$, $w_C = C$, $w_D = D$, registrul se încarcă (paralel) la aplicarea impulsului de ceas, cu cuvântul de patru biţi $ABCD$ de la o sursă independentă.

În tabelul din Figura 3.80-b este sintetizată funcţionarea registrului serie-paralel care corespunde circuitului 74xx194 iar simbolul de reprezentare pentru acest circuit este dat în Figura 3.80-c.



Functia	Selectare multiplexor		Starea urmatoare			
	S_1	S_0	w_A	w_B	w_C	w_D
Fara modificare	0	0	Q_A	Q_B	Q_C	Q_D
Depl. dreapta	0	1	RIN	Q_A	Q_B	Q_C
Depl. stanga	1	0	Q_B	Q_C	Q_D	LIN
Incarcare	1	1	A	B	C	D

b)

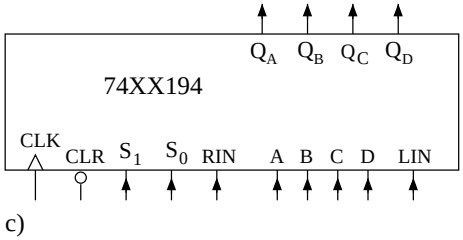


Figura 3.80 Regitrul serie-paralel: a) structurarea pentru un circuit cu patru celule; b),c) modul de funcţionare şi simbolul de reprezentare pentru circuitul 74xx194, registru universal cu patru celule.

Pentru un registru operaţia de deplasare (shiftare), stânga/dreapta cu o poziţie, a unui cuvânt pe un registru de deplasare necesită un tact, deci pentru n poziţii n tacte; dimensiunea circuitului fiind în $O(n)$, iar adâncimea în $O(1)$, cu produsul dimensiune adâncime în $O(n)$. Aceeaşi operaţie de deplasare, cu oricare număr i poziţii, $1 \leq i \leq n$, se poate realiza pe un circuit combinaţional de deplasare (vezi

2.5.4) într-un singur tact; acest circuit combinaţional având adâncimea în $O(log\, n)$ iar dimensiunea în $O(n\, log\, n)$ rezutând un produs dimensiune-adâncime în $O(n\, log^2 n)$. Încă o dată apare, evident, compromisul ce trebuie făcut între adâncime şi dimensiune în alegerea unei soluţii (relaţia 2.9). În calea de date a microprocesoarelor, Figura 2.73, unde shiftarea trebuie realizată pe un singur tact, indiferent de numărul de poziţii ($\leq n$), se include pe lângă unitatea aritmetico-logică şi un circuit de deplasare combinaţional, referit prin termenul de barrel shifter.

La un registru serie-paralel cuvântul de intrare, cu lungimea de n biţi, poate fi prezentat pentru încărcare fie în format serie, fie în format paralel. Aplicat cuvântul de intare, câte un bit la fiecare celulă bistabil, pentru încărcarea paralelă este necesar un singur tact de ceas, pe când la încărcarea serie se aplică fie la intarea RIN (deplasare dreapta) fie la LIN (deplasare stânga) câte un bit al cuvântului pentru fiecare tact de ceas, deci un număr de n tacte. De asemenea, pentru ieşirea paralelă cuvântul, cu lungimea de n biţi, conţinut în registru poate fi accesat permanent (dacă ieşirea nu este de tip TSL), dar la ieşirea serie se obţine câte un bit la fiecare tact de ceas, deci cuvântul se obţine într-un număr de n tacte. Combinând cele patru operaţii (intrare serială, intrare paralelă, ieşire paralelă şi ieşire serială) disponibile pe un circuit registru serie-paralel universal apare posibilitatea utilizării registrului ca suport pentru următoarele patru **operaţii de conversie/transfer: paralel-paralel, serie-serie, paralel-serie, serie-paralel**, Figura 3.81. Prin schimbarea modurilor de selectare, pe durata de funcţionare, se pot succeda atât pe intrare cât şi pe ieşire prezentarea serie cu cea paralelă şi invers. Conversiile serie-paralel şi paralel-serie sunt operaţii fundamentale în implementarea transmisiilor la distanţă pe un singur fir; la emisie cuvintele, printr-o conversie paralel-serie, sunt serializate, transmise pe un singur fir, iar la recepţie, printr-o conversie serie-paralel, sunt deserilizate.

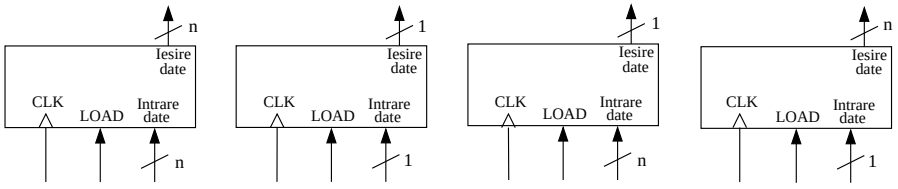


Figura 3.81 Registrul serie-paralel utilizat pentru funcţiunile de: a) transfer paralel-paralel; b) transfer serie-serie; c) conversie paralel-serie; d-conversie serie-paralel.

Se poate realiza un circuit din k registre serie, fiecare cu lungimea de n biţi, cu deplasare stânga-dreapta (reversibil), iar deplasările (nr. de poziţii, sens) se fac sincron în toate aceste registre. Într-un astfel de circuit pot fi stocate n cuvinte fiecare cu lungimea de k biţi, deci poate fi privit ca o memorie de capacitate (n × k) biţi. Se pot realiza, pe această structurare, două tipuri (speciale) de memorie:

1. **Memoria FIFO** (**F**irst-**I**n-**F**irst-**O**ut). Pe fiecare impuls de tact se înscrie în celulele bistabile de intrare (2^0) un cuvânt cu lungimea de k biţi şi este deplasat stânga cu o poziţie. După n tacte s-au înscris în registre n cuvinte şi pe tactele n+1, n+2, n+3, ... se obţin la ieşire (celulele bistabile de pondere 2^{n-1}) cuvintele în ordinea în care au fost înscrise (primul înscris primul citit!)

2. **Memoria LIFO** (**L**ast-**I**n-**F**irst-**O**ut). Pentru înscriere se procedează ca şi la memoria FIFO (deplasare stânga) dar de data aceasta citirea nu se mai face pe celulele de ieşire ci pe celulele bistabile de intrare (cu ponderea 2^0), la care de fapt s-a introdus cuvântul. Pentru citirea cuvintelor înscrise în memorie LIFO comanda de citire produce o deplasare dreapta, deci la celulele bistabil de intrare se obţine ultimul cuvânt introdus (ultimul înscris primul citit!).

Exemplul 3.28 În sistemele de calcul, în general, procesarea şi transferul informaţiei, reprezentată sub formă de cuvânt, se realizează simultan asupra tuturor biţilor cuvântului respectiv - aceasta este procesarea paralelă. În unele situaţii, procesarea sau transferul paralel este înlocuit cu variantele seriale, adică succesiv bit după bit pentru cuvântul respectiv. Un caz tipic, în acest sens, este transmisia la distanţa pentru care se procedează în modul următor: la partea de emisie, unde informaţia ce trebuie transmisă este sub forma unui cuvânt, acesta este serializat, prin intermediul unui registru paralel-serie şi transmis bit cu bit pe un singur fir, iar la recepţie cuvântul este convertit din nou la forma de cuvânt, prin intermediul unui registru serie-paralel. În continuare se va prezenta, ca structurare de principiu, conversia serie-paralel din punctul de recepţie, cea paralel-serie dintr-un punct de emisie poate fi privită şi realizată ca o "imagine în oglindă".

Pentru **transferul serial asincron pe o linie** se utilizează următorul protocol, care este standardizat. Cuvintele, sub formă de byte, se transmit pe linie, bit după bit începând cu D_0, LSB, şi terminând cu D_7, MSB; când nu este transmisie pe linie nivelul de tensiune prezent este H. Transmisia unui cuvânt de un byte începe printr-o tranziţie $H \rightarrow L$ a tensiunii de pe linie, Figura 3.82-a. Pentru receptor tranziţia $H \rightarrow L$ sesizată constituie informaţia că începe transmisia unui byte (poate fi şi un zgomot), iar dacă nivelul L al liniei se păstrează un timp T (durată de timp în secunde alocată aplicării unui bit al cuvântului) se consideră că acesta a fost bitul de "atenţionare" (şi nu a fost un zgomot care, în general, are o durată mai scurtă) referit ca **bitul de start**, aplicat obligatoriu la începutul transmisiei fiecărui byte. Apoi, se transmit cei opt biţi ai cuvântului, fiecare fiind aplicat pe linie o durată T, începând cu D_0 şi terminând cu D_7. Valoarea fiecărui bit recepţionat, pentru a micşora cât mai mult erorile de transmisie, este testată la mijlocul intervalului de timp T, alocat bitului respectiv; în consecinţă bitul D_0 este testat după un interval $3/2T$ de la sezizarea unei tranziţii $H \rightarrow L$ (şi dacă nivelul L s-a păstrat un interval T, bitul de start). După cele $(8+1)$ intervale T se mai introduc unul sau două intervale T, în care linia este comandată în nivelul H, ce se constituie în unul sau doi biţi se stop. **Bitul sau biţii de stop** sunt introduşi pentru a pregăti o transmisie corectă a următorului cuvânt, care poate porni imediat sau oricând. Deci, pentru transmisia asincronă a unui cuvânt de un byte, se transmit 8 biţi de date, un bit de start şi unul sau doi biţi de stop, de unde referirea ca tip de transmisie 8/10 sau 8/11.

Viteza de transmisie pe linie se măsoară în **baud** (rata în Baud, se citeşte bauzi) şi este definită ca inversul duratei T a celui mai scurt element din codul cuvântului transmis pe linie. Unele din frecvenţele (standard) inferioare sunt: 19200, 9600, 4800, 2400, 600, 300, 150, 110,75. De exemplu, pentru cuvinte în format 8/11, la o rată (frecvenţa) de 110 baud, se transmit 8 cuvinte adică numai 80 biţi de informaţie (şi nu 110); iar la o rată de 2400 baud, durata T este de $416\frac{2}{3}\mu s$. Realizarea transmisiei asincrone impune ca rata baud să aibă aceeaşi valoare la emisie şi la recepţie. Termenul asincron, din acest protocol, semnifică faptul că transmisia/(apariţia) unui cuvânt în linie poate fi în oricare moment de timp.

În Figura 3.82-b este structurat un bloc de recepţie asincron conform protocolului expus anterior. La apariţia pe linie a unei tranziţii $H \rightarrow L$ bistabilul de control este înscris în starea $Q = 1$ care comandă declanşarea astabilului sincronizat, cu o întârziere de $\frac{3}{2}T$. Semnalul generat de astabil, cu frecvenţa $1/T$, este utilizat ca semnal de ceas, pe frontul pozitiv,

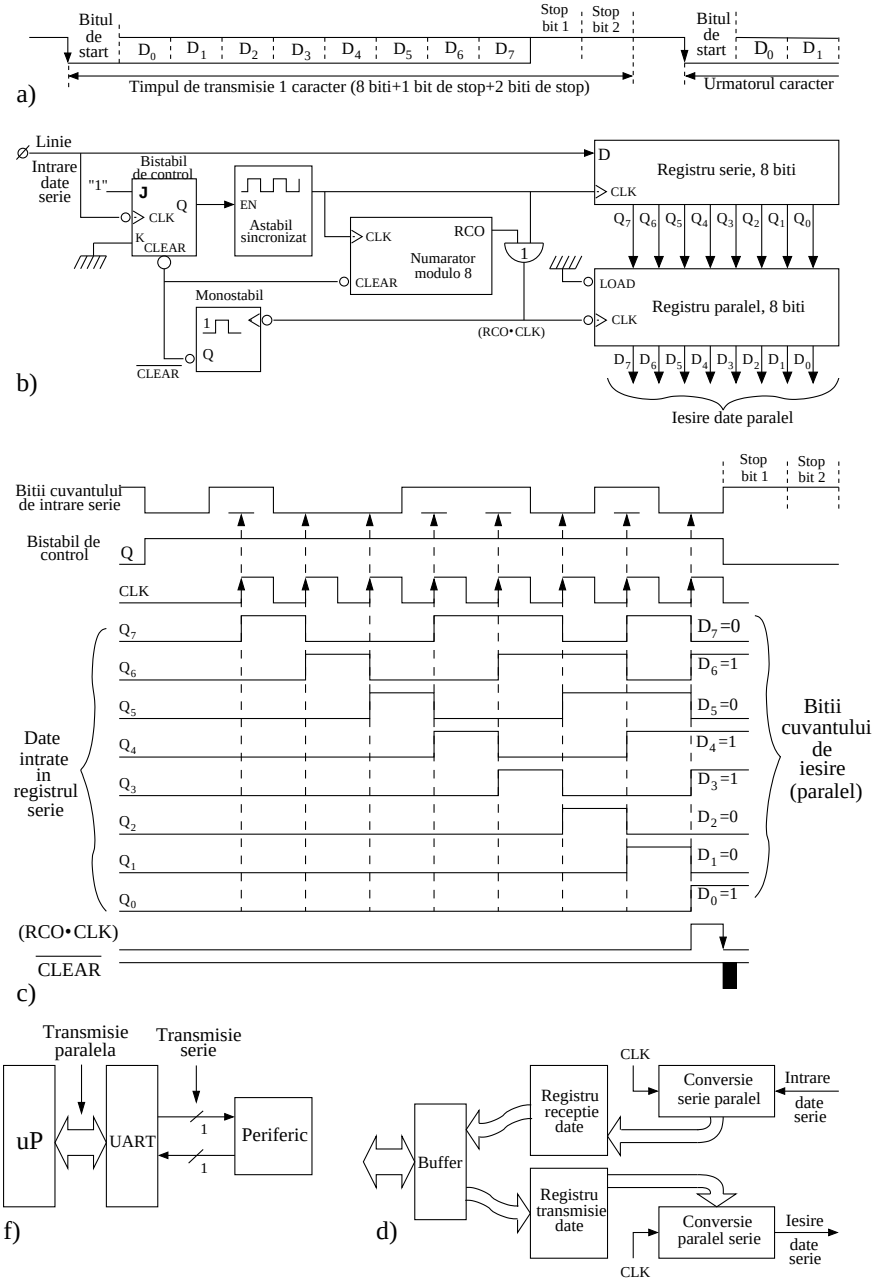


Figura 3.82 Transmisia serială asincronă: a) protocolul tip 8/10 sau 8/11 de transmisie asincronă; b) structurarea unui bloc de recepţie asincronă; c) diagrama semnalelor pentru procesul de conversie serie-paralel; d) structurarea, de principiu, a unui circuit UART şi (e) utilizarea sa ca interfaţă într-un sistem pe bază de microprocesor.

pentru încărcarea serială în registrul serie de 8 biţi a cuvântului recepţionat de pe linie şi pentru numărătorul modulo 8. După opt tacte de ceas registrul serie este plin cu byte-ul recepţionat iar numărătorul genereză semnalul RCO. Prin poarta 1, pe durata $T/2$ al celui de al optulea impuls de ceas, este generat semnalul (RCO $\cdot$ CLK) care comandă:

1 - încărcarea conţinutului registrului serie Q_7, Q_6, ..., Q_1, Q_0 în registrul paralel de 8 biţi, deci la ieşirea acestuia se obţine, în paralel, cuvântul D_7, D_6, ..., D_1, D_0 transmis serial pe linie;

2 - monostabilul care genereză un impuls $\overline{CLEAR}$ ce constituie semnalul de reset pentru bistabilul de control ($Q = 0$ deci astabilul este blocat) şi pentru numărătorul modulo 8. Blocul de recepţie este pregătit pentru recepţia următorului cuvânt pe linie care poate apare oricând după consumarea unuia sau a doi biţi de stop. Diagramele semnalelor pentru procesul de recepţie şi conversie serie-paralel sunt prezentate în Figura 3.82-c.

Circuitul **UART** (**U**niversal **A**synchronous **R**eceiver **T**ransmitter), Figura 3.82-d, obtenabil comercial, include atât partea de recepţie cât şi partea de emisie. Partea de recepţie, cu o structură similară cu cea prezentată anterior, recepţionează serie, converteşte în paralel şi printr-un registru buffer depune datele pe o magistrală paralelă. Partea de emisie, printr-un buffer, preia datele de pe o magistrală paralelă şi după o convesie paralel-serie transmite serial datele pe linie. Un astfel de circiut totdeauna interfaţează, într-un sistem pe bază de microprocesor, conectarea unui periferic ce are intrarea şi ieşirea serie, Figura 3.82-e

---

3.5.6 Circuite liniare cu registre de deplasare

Un sistem, definit ca o aplicaţie f de pe mulţimea intrărilor X pe mulţimea ieşirilor Y, $Y = f(X)$, este referit ca fiind liniar dacă i se poate aplica principiul superpoziţiei (1. răspunsul la două sau mai multe intrări este egal cu suma răspunsurilor la fiecare intrare în parte: dacă $y_1 = f(x_1)$, $y_2 = f(x_2)$ atunci $y_1 + y_2 = f(x_1 + x_2)$); 2. păstrează factorul de scalare al intrărilor (pentru $a \cdot x \rightarrow af(x)$). Pe mulţimea binară elementele liniare sunt: elementul de întârziere (celula bistabil D) şi operatorul XOR (NXOR); nu sunt liniari operatorii OR şi AND. Deci **sistemele digitale realizate cu celule D (registre de deplasare) şi porţi XOR sunt referite ca sisteme liniare**. Pentru sistemele (secvenţiale) liniare, utilizând adunarea modulo 2 şi nu adunarea logică, când mulţimea de definiţie este 0,1, aritmetica cu care se operează este pe câmpul Galois $GF(2)$; se pot concepe astfel de sisteme pe oricare câmp Galois $GF(q)$, vezi secţiunea 1.1.2. Operaţiile pe $GF(2)$ sunt identice cu cele din aritmetica numerelor reale cu deosebire de adunare (care este modulo 2). Pe $GF(2)$ oricare element, pentru adunare, este identic cu inversul său $A = -A$, deci scăderea este identică cu adunarea ($A \oplus A = 0$, $A \oplus 0 = A$, $A \oplus 1 = -A$). Amintim relaţia: dacă $A \oplus B = C$ atunci $A = C \oplus B$ (pentru că $A \oplus B \oplus B = C \oplus B \rightarrow A = C \oplus B$), $B = A \oplus C$ şi $A \oplus B \oplus C = 0$.

Pentru analiza funcţionării circuitelor realizate cu porţi logice XOR şi celule de întârziere cu bistabile D se vor utiliza (pentru comparaţie) patru circuite; Figura 3.83: două compuse numai din din câte o poartă XOR şi o celulă D, dar unul din ele cu reacţie; două circuite mai complexe realizate din cinci celule bistabil D şi patru porţi XOR, dar unul, de asemenea, cu reacţie. Se consideră că toate celulele bistabil comută sincron pe frontul semnalului de ceas. După aplicarea frontului activ de ceas, pe durata perioadei de ceas T, ieşirea oricărui circuit se calculează ca o sumă modulo

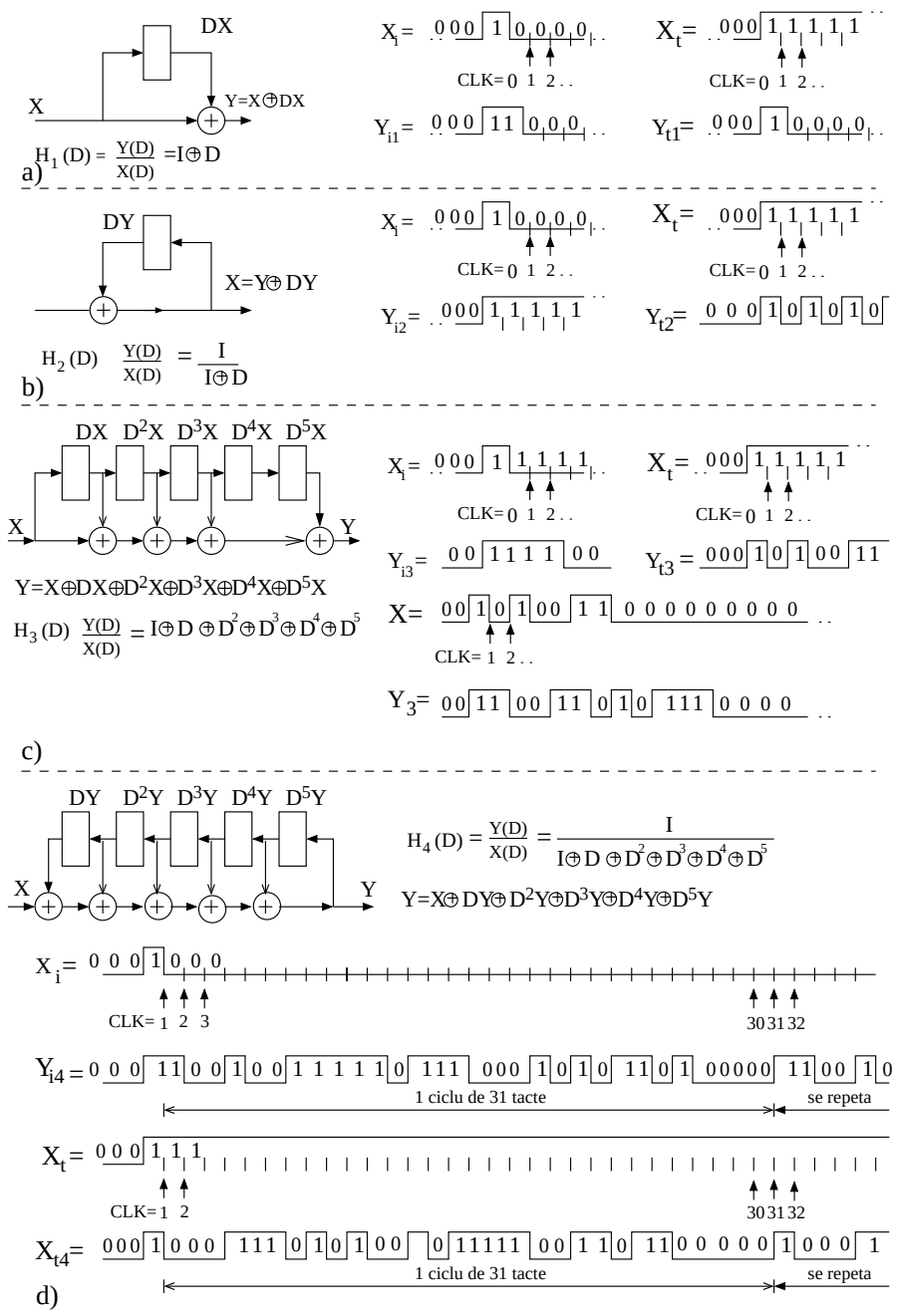


Figura 3.83 Secvenţele de răspuns Y_i şi Y_t pentru semnalele de intrare impuls unitar X_i şi treaptă unitară X_t aplicate la: a),c) circuite secvenţiale liniare fară reacţie; b),d) circuite secvenţiale liniare cu reacţie.

2 între valoarea bitului existent pe intrare şi cu anumite valori anterioare ale biţilor din succesiunea X aplicată pe intrare, sau cu valorile anumitor biţi din succesiunea Y obţinută la ieşire, valori care au fost stocate în celule de memorare D. Rezultă că ieşirea depinde, pe lângă intrarea prezentă, şi de "istoria" funcţionării circuitului. Apare o similitudine, în funcţionare, cu un automat sincron(care la fel depinde de "istoria" sa), de unde pentru denumirea acestor circuite liniare s-a ajuns la sintagma **circuite secvenţiale liniare**.

Semnalele utilizate în testarea acestor circuite, prin care se pot evidenţia anumite particularităţi de structurare ale circuitului respectiv, sunt semnalul impuls unitar şi semnalul treaptă unitară. Pentru sistemele digitale, **semnalul impuls unitar** este format din 1 precedat şi urmat de un şir infinit de zero-uri, $X_i = \ldots 0001000 \ldots$, iar **semnalul treaptă unitară** este format dintr-un şir infinit de 1 precedat de un şir infinit de zero-uri $X_t = \ldots 000111 \ldots$.

Pentru circuitul simplu din Figura 3.83-a, compus dintr-o poartă XOR şi o celulă D se poate deduce simplu modul de calcul al bitului de ieşire $Y_1 = X \oplus DX$. Rezultă că fiecare bit de ieşire, după aplicarea frontului activ de ceas (tactului) pe durata perioadei de ceas, T_{CLK}, se calculează prin sumă modulo 2 între valoarea bitului prezent pe intrare X şi bitul anterior de intrare DX (stocat în celula D), deci $Y = X(I \oplus D)$; I - este operatorul unitar, $D^0 = I$, relaţia 3.48. Răspunsurile Y_{i1} şi Y_{t1}, la aplicarea semnalelor impuls unitar X_i şi treaptă unitară Y_t, sunt prezentate în aceeaşi figură.

Circuitul din Figura 3.83-b, cu aceleaşi componente ca şi cel din Figura 3.83-a, dar pentru elementul de întârziere fluxul de transfer este de la ieşire la intrare, realizează o reacţie, deci se memorează ieşirea, DY. Din structura circuitului se deduce $Y = X \oplus DY$, care se transformă în $Y \oplus DY = X$, deci $X = Y(I \oplus D)$. Fiecare bit de ieşire, pe durata unei perioade de ceas, T_{CLK}, se calculează prin sumă modulo 2 între valoarea bitului prezent pe intrare X şi a bitului de ieşire DY, care a fost în perioada anterioară a semnalului de ceas (înainte de aplicarea tactului). Răspunsurile Y_{i2} şi Y_{t2}, la aplicarea semnalelor impuls unitar X_i şi treaptă unitară X_t, sunt prezentate în aceeaşi figură. Răspunsul la un semnal treaptă Y_{t2} este un semnal cu variaţie dreptunghiulară cu perioada $2T_{CLK}$ (de fapt acest circuit nu este altceva decât o transformare a bistbilului D în bistabil T, Figura 3.50-b, care divizează cu doi frecvenţa semnalului de ceas când $T = 1$).

Analizând circuitul din Figura 3.83-c se poate deduce uşor relaţia pentru semnalul de ieşire, $Y = X \oplus DX \oplus D^2X \oplus D^3x \oplus D^5X = X(I \oplus D \oplus D^2 \oplus D^3 \oplus D^5)$. Bitul de ieşire, pe durata unei perioade de ceas T_{CLK}, se calculează prin sumă modulo 2 între valoarea bitului aplicat pe intrare şi valorile biţilor anteriori (aplicaţi cu unu, două, trei şi cinci tacte înainte) din şirul de intrare, valori memorate sub forma DX, D^2X, D^3X şi D^5X. După această regulă sunt calculate răspunsurile la semnal impuls Y_{i3} şi la semnal treaptă Y_{t3}, prezentate în aceeaşi figură.

La circuitul din Figura 3.83-d, din modul de realizare a conexiunilor se deduce expresia pentru calculul semnalului de ieşire $Y = X \oplus DY \oplus D^2Y \oplus D^3Y \oplus D^5Y \rightarrow Y \oplus Y \oplus X = Y \oplus X \oplus X \oplus DY \oplus D^2Y \oplus D^3Y \oplus D^5Y \rightarrow X = Y(I \oplus D \oplus D^2 \oplus D^3 \oplus D^5)$. Pentru fiecare bit din şirul de intrare, valoarea bitului de ieşire se obţine prin sumare modulo 2 între valoarea bitului aplicat pe intrare şi valorile biţilor de ieşire anteriori DY, D^2Y, D^3Y şi D^5Y. Spre deosebire de circuitul anterior, unde în registru de deplasare sunt introduşi biţii şirului de intrare, aici în registrul de

deplasare sunt introduşi biţii şirului de ieşire. Aplicând regula de calcul a circuitului, pe aceeaşi figură, sunt prezentate şirurile Y_{i4} şi Y_{t4} care sunt răspunsurile la semnalul impuls unitar şi semnalul treaptă unitară. Ambele răspunsuri sunt şiruri periodice de impulsuri, perioadă ce se întinde pe durata a 31 impulsuri de tact. Pentru Y_{i4} se observă că impulsul unitar de pornire nu se mai aplică (la al 31-lea tact de ceas), deci poate fi considerat ca un ciruit cu funcţionare autonomă, cu un ciclu de 31 tacte, pentru amorsare este necesar doar un singur impuls. Circuitele cu funcţionare autonomă sunt utilizate mai mult pentru succesiunea de cuvinte generate în paralel, la ieşirile bistabilelor care formează registrul de deplasare, decât pentru generarea succesiunii serie de biţi la ieşire.

Utilizând principiul superpoziţiei, succesiunea de biţi de la ieşire, pentru oricare succesiune de biţi aplicată la intrare, se poate obţine pe baza răspunsului la semnalul impuls unitar. Se va exemplifica pentru circuitul din Figura 3.83-c. Deoarece un semnal treaptă unitară, X_t, se poate compune printr-o sumare infinită de semnale impuls unitar, X_i, defazat unul de altul cu T_{CLK}, rezultă răspunsul la semnal treaptă prin sumare modulo 2 a răspunsurilor de semnal impuls în felul următor:

$$\begin{array}{r} \ldots 00010000\ldots \\ \ldots 00010000\ldots \\ \ldots 00010000\ldots \\ \ldots 00010000\ldots \\ \hline X_t = \ldots 1111\ldots\ldots\ldots \end{array} \qquad \begin{array}{r} \ldots 00011110100\ldots \\ \ldots 00011110100\ldots \\ \ldots 00011110100\ldots \\ \ldots 00011110100\ldots \\ \hline X_{t3} = \ldots 101001111\ldots \end{array}$$

Invers, deoarece un semnal impuls X_i poate fi compus prin sumă modulo 2 între două semnale X_t defazate cu T_{CLK}, semnalul X_{i3} se obţine prin sumarea modulo 2 între două semnale Y_{t3} (încercaţi!). În Figura 3.83-c este prezentat răspunsul circuitului şi pentru următoarea succesiune de intrare compusă din şapte biţi $X = \ldots 0001010011000\ldots$. Compunând această succesiune din semnale impuls unitar defazate corespunzător se obţine:

$$X = \oplus \left\{ \begin{array}{l} \ldots 00010000\ldots \\ \ldots 0000010000\ldots \\ \ldots 0000000010000\ldots \\ \underline{\ldots 00000000010000\ldots} \\ \ldots 00010100110000\ldots \end{array} \right. \Rightarrow Y_3 = \oplus \left\{ \begin{array}{l} \ldots 0001111010000\ldots \\ \ldots 000001111010000\ldots \\ \ldots 000000001111010000\ldots \\ \underline{\ldots 0000000001111010000\ldots} \\ \ldots 0001100110101110000\ldots \end{array} \right.$$

Această operare, în timp, asupra biţilor de intrare şi asupra biţilor de intrare/ieşire anteriori, stocaţi în celulele bistabil, este destul de laborioasă şi supusă la erori, mai ales dacă sunt multe celule de memorare. Se poate evita această operare, printr-o algebrizare a calculelor, dacă şi şirurile de biţi de intrare şi cele de ieşire sunt reprezentate ca polinoame după puterile variabilei D. De exemplu, pentru secvenţa de 7 biţi, utilizată anterior, $X = \ldots 0001010011000\ldots$ se face reprezentarea $X(D) = I \oplus D^2 \oplus D^5 \oplus D^6$. Bitul b_i din poziţia i a secvenţei de intrare este reprezentat în polinom prin termenul $b_i D^i$ (pentru $b_i = 0 \rightarrow 0 \cdot D^i = 0$ şi pentru $b_i = 1 \rightarrow 1 \cdot D^i = D^i$); primul bit care se aplică circuitului, ce corespunde cu MSB, are poziţia zero şi are totdeauna valoarea 1, deci $1 \cdot D^0 = I$. Pentru secvenţele de impuls unitar X_i şi treaptă

unitară X_t se obţin următoarele reprezentări polinomiale:

$$\begin{array}{ll} X_i(D) = & 1 \cdot D^0 = I \\ X_t(D) = & I \oplus D \oplus D^2 \oplus D^3 \oplus D^4 \oplus D^5 \oplus D^6 \oplus \ldots \end{array} \tag{3.52}$$

Raportul între polinomul secvenţei de ieşire $Y(D)$ şi al polinomului secvenţei de intrare $X(D)$ defineşte **funcţia de transfer a circuitului,** $H(D)$:

$$H(D) = \frac{Y(D)}{X(D)} \tag{3.53}$$

Funcţia de transfer pentru un circuit se deduce din analiza conexiunilor (structurarea) între registrul de deplasare şi porţile XOR. De exemplu, pentru circuitele fără reacţie din Figura 3.83 se deduc

$$H_1(D) = (I \oplus D) \qquad H_3(D) = I \oplus D \oplus D^2 \oplus D^3 \oplus D^5$$

iar pentru circuitele cu reacţie

$$H_2(D) = I/I \oplus D \qquad H_4(D) = I/(I \oplus D \oplus D^2 \oplus D^3 \oplus D^5)$$

Cunoscând polinomul secvenţei de intrare $X(D)$ şi funcţia de transfer a circuitului $H(D)$ cu relaţia 3.53 se poate calcula polinomul secvenţei de ieşire $Y(D)$, care este o mapare a biţilor 1 din secvenţa de ieşire. Prin acest mod de calcul se va exemplifica în continuare determinarea (şi în acelaşi timp verificarea) secvenţelor de ieşire pentru semnalele de intrare prezentate în Figura 3.83-c.

1. Răspunsul la treaptă unitară, $X_t(D)$.

$$\begin{array}{rcl} Y_{t3}(D) & = & H_3(D) \cdot X_t(D) = H_3(D) \cdot (I \oplus D \oplus D^2 \oplus D^3 \oplus D^4 \oplus \ldots) = \\ & = & I \cdot H_3(D) \oplus D \cdot H_3(D) \oplus D^2 \cdot H_3(D) \oplus D^3 \cdot H_3(D) \oplus \ldots = \\ & = & H_3(D) \oplus D \cdot H_3(D)(I \oplus D \oplus D^2 \oplus D^3 \oplus \ldots) = H_3(D) \oplus D \cdot Y_{t3} \end{array}$$

$$\begin{array}{rcl} \Big(Y_{t3}(D) \oplus D \cdot Y_{t3}(D) & = & H_3(D) \oplus D \cdot Y_{t3}(D) \oplus D \cdot Y_{t3}(D) = \\ & = & H_3(D) \rightarrow Y_{t3}(I \oplus D) = H_3(D)\Big) \end{array}$$

$$\begin{array}{rcl} Y_{t3}(D) & = & H_3(D)/(I \oplus D) = (I \oplus D \oplus D^2 \oplus D^3 \oplus D^5)/(I \oplus D) = \\ & = & I \oplus D^2 \oplus D^5 \oplus D^6 \oplus D^7 \oplus \ldots \text{ deci } 0001010011\ldots \end{array}$$

$$\begin{array}{c} \big((I \oplus D \oplus D^2 \oplus D^3 \oplus D^5) \oplus (D^6 \oplus D^6) \oplus (D^7 \oplus D^7) \oplus (D^8 \oplus D^8) = \\ = (I \oplus D) \oplus (I \oplus D)D^2 \oplus (I \oplus D)D^5 \oplus (I \oplus D)D^6 \oplus (I \oplus D)D^7 \oplus \ldots\big) \end{array}$$

2. Răspunsul la impuls unitar, $X_i(D)$.

$$Y_{i3}(D) = H_3(D) \cdot X_i(D) = H_3(D) = I \oplus D \oplus D^2 \oplus D^3 \oplus D^5 \text{ deci } Y_{i3} = 11110100\ldots$$

3. Răspunsul la secvenţa $X = \ldots 0001010011000\ldots$

$$\begin{aligned} Y_3(D) &= H_3(D)\cdot X(D) = (I \oplus D \oplus D^2 \oplus D^3 \oplus D^5)\cdot(I \oplus D^2 \oplus D^5 \oplus D^6) \\ &= I \oplus D \oplus D^4 \oplus D^5 \oplus D^7 \oplus D^9 \oplus D^{10} \oplus D^{11} \end{aligned}$$

deci $Y_3 = 110011010111000\ldots$

Similar, se pot calcula răspunsurile la semnal treaptă şi semnal impuls pentru circuitul cu reacţie cu funcţia de transfer $H_4(D)$. Dificultatea, de data aceasta, constă în faptul că pentru $Y_{i4}(D)$ şi $Y_{t4}(D)$ rezultă polinoame a căror puteri ajung la D^{31}. Dar mai simplu, aceste două polinoame pot fi obţinute, invers, dacă şirul de biţi, în timp, pentru Y_{i4} şi Y_{t4} din Figura 3.83-d este mapat direct în forma polinomială (se consideră numai puterile pentru care există impuls în răspunsul în timp, $0D^i = 0{,}1D^i = D^i$).

La realizarea unui singur circuit din mai multe circuite componente cu funcţiile de transfer $H_1(D)$, $H_2(D)$, $\ldots H_k(D)$, pentru determinarea funcţiei de transfer rezultante $H(D)$, se utilizează algebra funcţiilor de transfer. Pentru circuitul obţinut, prin înserierea de circuite componente, funcţiile de transfer se înmulţesc $H(D) = H_1(D)\cdot H_2(D)\ldots H_k(D)$; prin conectare în paralel de circuite componente funcţiile de transfer se sumează modulo 2 $H(D) = H_1(D) \oplus H_2(D) \oplus \ldots \oplus H_k(D)$, iar pentru serie paralel sau paralel serie se combină înmulţirea cu sumarea.

Din analiza circuitelor liniare cu registre de deplasare rezultă că acestea pot fi utilizate ca **filtre binare** - o succesiune de biţi aplicată pe intrare este modificată într-o altă succesiune de biţi de ieşire în funcţie de funcţia de transfer a circuitului. Dar, abordarea poate fi şi de sinteză, impunându-se transformarea unei anumite secvenţe de intrare $X(D)$ într-o anumită secvenţă de ieşire $Y(D)$ se determină funcţia de transfer $H(D)$ - apoi, din funcţia de transfer se deduce structura circuitului căutat.

Exemplul 3.29 Pentru circuitul cu funcţia de transfer $H(D) = I \oplus D \oplus D^3$ să se determine secvenţa de intrare nulă.

Soluţie. Secvenţa de intrare nulă $X_0(D)$ este acel şir de biţi aplicat pe intrare care produce pe ieşire secvenţa $Y(D) = \ldots 0000\ldots$, deci respectă relaţia

$$X_0(D)\cdot H(D) = X_0(D) \oplus DX_0(D) \oplus D^3X_0(D) = 0 \qquad \rightarrow \qquad X_0(D) = DX_0(D) \oplus D^3X_0(D)$$

Expresia obţinută pentru $X_0(D)$ arată că fiecare bit care trebuie să se aplice pe intrare se calculează ca sumă modulo 2 între biţii registrului de deplasare din celulele D şi D^3. Dar pornind din repaus cu acest filtru, adică toate celulele pline cu 0, rezultă că adevărata secvenţă nulă este un şir de zero-uri (dar care nu va putea scoate filtrul din repaus). Pentru a scoate filtrul din repaus se acceptă ca secvenţa de intrare nulă să aibă primul bit 1, care generează şi în ieşire primul bit 1, ceea ce violează şirul de zero-uri pe ieşire pentru primul bit. Dar procedând în acest mod secvenţa nulă de ieşire este considerată ca fiind răspunsul unitar $Y(D) = I$, deci relaţia anterioară se rescrie.

$$I = X_0(D)\cdot H(D) = X_0(D) \oplus D\cdot X_0(D) \oplus D^3\cdot X_0(D)$$

$$X_0(D) = I/(I \oplus D \oplus D^3) = 1 \oplus D \oplus D^2 \oplus D^4 \oplus D^7 \oplus D^8 \oplus \ldots$$

rezultă secvenţele:

$$\begin{aligned} X_0 &= \ldots 000\dot{:}1110100\dot{:}1110100\dot{:}1110100\dot{:}\ldots \\ Y = X_i &= \ldots 000\dot{:}1000000\dot{:}0000000\dot{:}0000000\dot{:}\ldots \end{aligned} \tag{3.54}$$

Aplicaţii în transmisia codurilor ciclice. Oricare CLC cu ieşiri multiple poate fi privit ca un transcodor, Figura 2.33, conversia într-un alt cod efectuându-se în paralel. Pentru codurile ciclice, ai căror biţi se transmit serial, codificarea şi decodificarea se realizează prin trecerea şirului de biţi prin circuite secvenţiale liniare. Codurile ciclice, care aparţin unei clase generale a codurilor algebrice detectoare de erori, prezintă distinctiv faptul că: dacă vectorul $(b_{n-1}b_{n-2}\dots b_3b_2b_1b_0)$ reprezintă un cuvânt al codului atunci şi combinaţia $(b_0b_{n-1}b_{n-2}\dots b_3, b_2b_1)$ obţinută prin deplasarea ciclică a biţilor, aparţine de asemenea codului.

În principiu, pentru transmisia codurilor ciclice (secvenţa $X(D)$) acestea sunt aplicate la emisie într-un circuit secvenţial cu funcţia de transfer $H(D)$ ce generează secvenţa de ieşire $Y(D)$. Ceea ce de fapt este o înmulţire modulo 2 a polinomului de intrare $X(D)$ (codul de transmis) cu polinomul circuitului (funcţia de transfer). Este posibil ca pe canalul de transmisie să fie injectat un zgomot, N, la recepţie ajungând secvenţa $X'(D) = Y'(D) + N$. Recepţia se face printr-un circuit cu o funcţie de transfer de forma $1/H(D)$ obţinându-se la ieşire secvenţa:

$$Y'(D) = \frac{1}{H(D)}(Y(D) + N) = \frac{1}{H(D)}(X(D)H(D) + N) = X(D) + \frac{1}{H(D)} \cdot N$$

De data aceasta se realizează o împărţire cu acelaşi polinom cu care s-a înmulţit le emisie; la o transmisie corectă restul împărţirii trebuie să fie nul.

Dacă cuvântul de cod nu este modificat de zgomot ($N = 0$) atunci $Y'(D) = X(D)$. Dacă $N \neq 0$ atunci analiza termenului $\frac{1}{H(D)} \cdot N$, la recepţie, va indica transmisia încărcată cu zgomot a cuvântului $X(D)$; dar chiar mai mult, se poate indica poziţia din cuvânt a bitului/biţilor modificat/modificaţi precum şi corectarea pentru obţinerea cuvântului (iniţial) corect [Vlăduţiu '89].

Exemplul 3.30 În Figura 3.84, un cuvânt de patru biţi, 1110, din cele 16 posibile, este transmis printr-o pereche de filtre liniare cu funcţii inverse. La emisie, cuvântul este extins la o lungime de cod de 7 biţi, 1110 000, adăugând pe ultimele trei poziţii trei biţi 000, referiţi ca biţi de control. Se obţine la ieşire din filtrul de la emisie, care are funcţia de transfer $H(D) = I \oplus D^2 \oplus D^3$, următorul şir 1100010. În linia de transmisie, un zgomot injectează în poziţia a patra din dreapta un bit 1, deci la intrarea filtrului de recepţie, cu funcţia $1/H(D)$, se aplică şirul $X' = 1101010$. Se obţine şirul de ieşire $Y' = 1111\,001$ în care ultimii trei biţi de control diferă de 000, ceea ce indică faptul că transmisia s-a realizat cu eroare (pe baza cuvântului de control 001 se poate detecta şi corecta eroarea).)

Circuite liniare cu registre de deplasare cu reacţie, autonome. Aceste circuite după cum reiese şi din această denumire (lungă!) sunt de fapt filtre binare cu reacţie, prezentate anterior, dar la care este eliminată intrarea, referite în literatură ca circuite de deplasare cu reacţie, **LFSR** (autonomous **L**inear-**F**eedback **S**hift-**R**egister). Neexistând intrare, circuitul după amorsare continuă să genereze un semnal periodic, prin aplicarea impulsurilor de ceas, ceea ce poate fi analogul discret al unui oscilator.

De exemplu, la circuitul cu reacţie din Figura 3.83-d răspunsul Y_{i4} la un semnal impuls unitar este o secvenţă cu o perioadă de 31 de tacte. De asemenea, la circuitul cu funcţia de transfer $H(D) = I \oplus D^2 \oplus D^3$ din Exemplul 3.29 aplicând pe intrare secvenţa nulă se obţine pe ieşire secvenţa care este semnalul impuls unitar X_i; dar

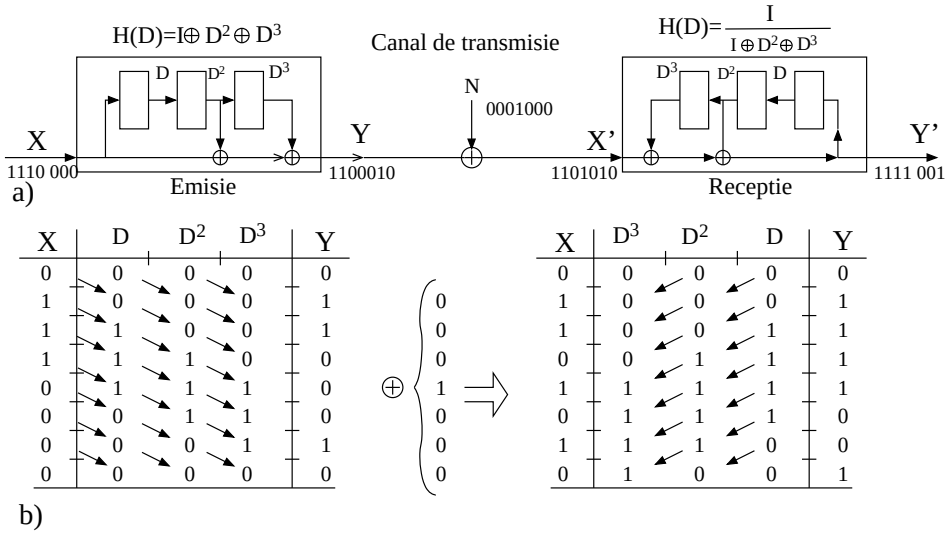


Figura 3.84 Transmisie cu detectare şi corectare de eroare: a) structură de circuit pe bază de filtre secvenţiale liniare cu funcţii de transfer inverse; b) tabele cu modificarea conţinuturilor filtrelor de la emisie şi de la recepţie.

dacă acest circuit este realizat cu reacţie, cu funcţia de transfer $1/H(D)$ care are structura circuitului receptor din Figura 3.84-a, atunci aplicându-i pe intrare (pentru amorsare) secvenţa impuls unitar se obţine pe ieşire un semnal periodic care este secvenţa nulă (invers, în raport cu relaţia 3.53).

$$X_i = \ldots 000\dot{:}1000000\dot{:}0000000\dot{:}\ldots$$
$$Y_0 = \ldots 000\dot{:}1110100\dot{:}1110100\dot{:}\ldots$$

La aceste două circuite cu reacţie se constată că fiind în starea zero, toate celulele pline cu 0, dacă se aplicaă numai o singură dată impulsul unitar, vor genera permanent o secvenţă periodică, chiar dacă intrarea este zero, deci au autonomie faţă de intrare.

Structura de principiu pentru un circuit de deplasare cu reacţie ca n celule este prezentată în Figura 3.85-a. La sumatoarele modulo 2 intrările sunt ieşirile din celulele registrului de deplasare multiplicate cu coeficienţii $a_i, i = 1, 2, \ldots, n-1,\ a_i \in \{0,1\}$. (De fapt aceşti coeficienţi indică dacă ieşirea celulei i a registrului este conectată în sumarea modulo 2, $a_i = 1$, sau dacă nu este conectată, $a_i = 0$; ieşirea din ultima celulă totdeauna este conectată, $a_n = 1$). Un astfel de circuit cu reacţie este un automat deci, pentru analiza sa, este mai potrivită abordarea din punct de vedere al tranziţiei stărilor şi nu descriind şirul biţilor de ieşire Y. În Figura 3.83-d , de exemplu, şirul Y_{i4} introdus în registrul de deplasare va realiza următoarele tranziţii

ale stărilor ($D^5Y = z_5,\ D^4Y = z_4,\ D^3Y = z_3,\ D^2Y = z_2,\ DY = z_1$):

$$
\begin{aligned}
z_5z_4z_3z_2z_1 \quad = \quad & (0) \to (1) \to (3) \to (6) \to (12) \to (25) \to (18) \to (4) \to (9) \\
\to \quad & (19) \to (7) \to (15) \to (31) \to (30) \to (29) \to (27) \to (25) \\
\to \quad & (14) \to (28) \to (24) \to (17) \to (2) \to (5) \to (10) \to (21) \\
\to \quad & (11) \to (22) \to (13) \to (26) \to (20) \to (9) \to (16) \to (1) \to \ldots
\end{aligned}
$$

(cifrele încercuite reprezintă numărul în zecimal al echivalentului binar dat de codul stării $z_5z_4z_3z_2z_1$).

Particularizând structura de principiu la n=4, funcţia de transfer a stărilor se exprimă în felul următor:

$$
\begin{aligned}
w_4 &= z_3 \\
w_3 &= z_2 \\
w_2 &= z_1 \\
w_1 &= a_4z_4 \oplus a_3z_3 \oplus a_2z_2 \oplus a_1z_1 = Y
\end{aligned}
\tag{3.55}
$$

Primele trei ecuaţii descriu deplasările (stânga) din registru iar ultima reacţia liniară prin reţeaua de porţi sumă modulo 2. Ecuaţia pentru calculul semnalului de reacţie Y, care se aplică la intrarea primului bistabil D al registrului, este:

$$a_4D^4Y \oplus a_3D^3Y \oplus a_2D^2Y \oplus a_1DY = Y \ \to\ a_4D^4Y \oplus a_3D^3Y \oplus a_2D^2Y \oplus a_1DY \oplus Y = 0$$

$$a_4D^4Y \oplus a_3D^3Y \oplus a_2D^2Y \oplus a_1DY \oplus Y = 0 \ \to\ a_4D^4 \oplus a_3D^3 \oplus a_2D^2 \oplus a_1D \oplus I = 0$$

iar particularizând $a_4 = 1$, $a_3 = 0$, $a_2 = 0$, $a_1 = 1$, şi pentru obişnuinţă şi generalitate adoptând notaţia în X $(= Y)$, se obţine **polinomul caracteristic**:

$$I \oplus X \oplus X^4 \tag{3.56}$$

care corespunde circuitului din figura 3.85-b. Ordonarea (directă) a termenilor unui polinom de ordinul n pe cele n celule ale registrului de deplasare se face printr-o mapare directă în sensul de creştere a exponentului puterii şi a creşterii numărului de tacte de întârziere pe registru. Termenul polinomului X^k, $1 \le k < n$ îi corespunde în registru celula care realizează o întârziere a intrării cu k tacte (D^k), iar termenilor X^n şi 1, care sunt totdeauna prezenţi în reţeaua de recţie a circuitului, le corespunde în registru respectiv ultima celulă (D^n) şi intrarea primei celule ($D^0 = 1$). Pentru polinomul caracteristic din relaţia 3.56, într-o ordonare directă, în reţeaua sumatoare modulo 2, din exteriorul registrului, sunt conectate ieşirile celulei a patra (X^4) şi celulei a întâia (X) şi rezultatul aplicat la intarea primei celule (D^0); structurarea rezultată este referită ca circuit secvenţial liniar cu reacţie cu sumatoare exterioare. Pentru acelaşi polinom caracteristic (3.56) există şi structurarea cu sumatoare incluse, prezentată în Figura 3.85-c, în care se conectează celulele bistabil ale registrului la reţeaua de sumatoare modulo 2 după aceeaşi regulă expusă anterior. Sub desenul corespunzător fiecărei din cele două structurări, cu sumator exterior şi sumator inclus, este ataşat şi tabelul de tranziţie al stărilor (cu calculul biţilor fiecărei stări).

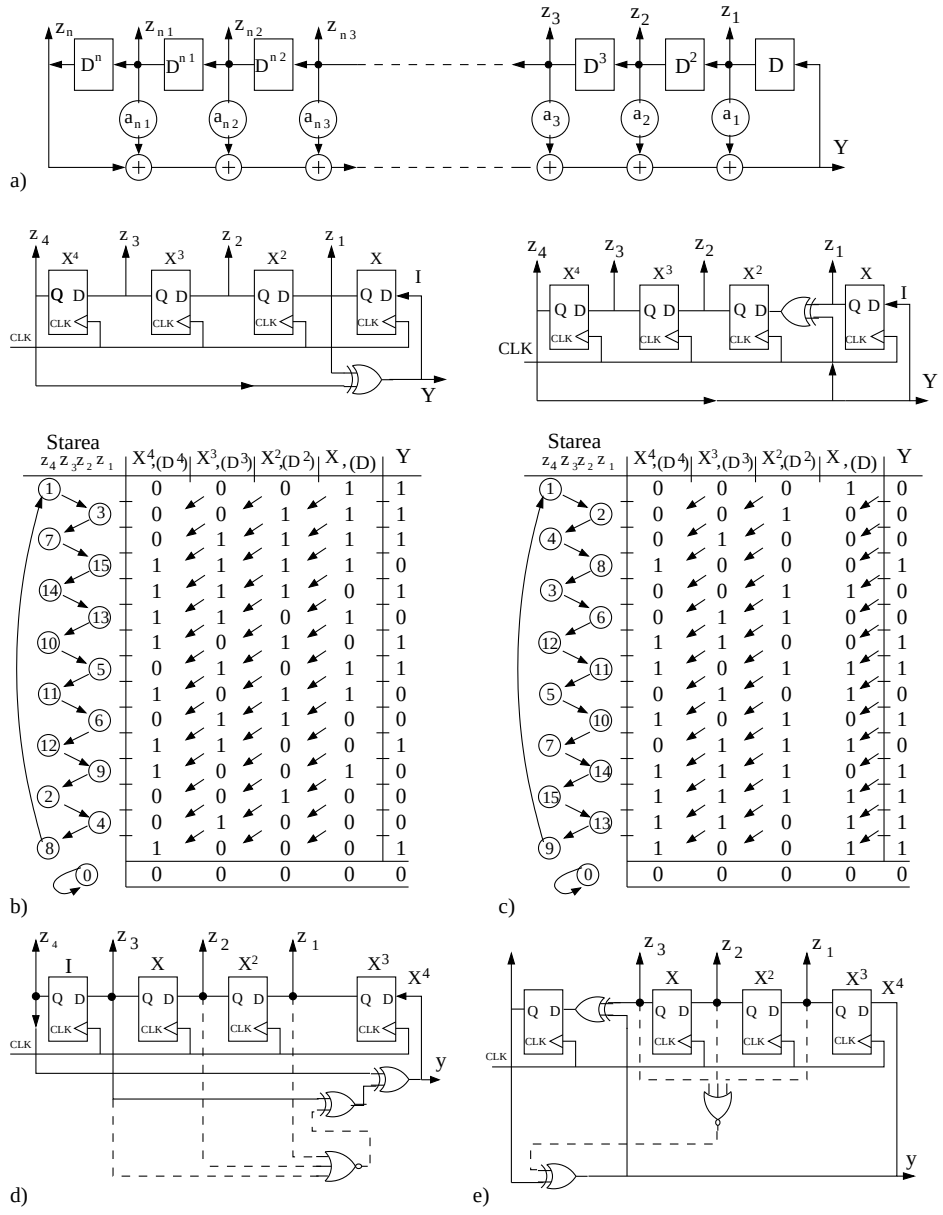


Figura 3.85 Circuitul secvenţial liniar cu reacţie: a) structurare generală, de principiu, pentru un (circuit bazat pe) registru de deplasare cu n celule; structura de circuit şi tabelul de tranziţie al stărilor pentru polinomul $I \oplus X \oplus X^4$, în ordonare directă cu sumator extern (b) şi cu sumator intern (c); structură de circuit pentru polinomul $I \oplus X \oplus X^4$, în ordonare inversă (sau polinomul inversat $X^4 \oplus X^3 \oplus I$ şi ordonare directă), cu sumator extern (d) şi cu sumator intern (e)(Liniile desenate întrerupt corespund introducerii în ciclu normal şi a stării $z_4 z_3 z_2 z_1 = 0000$).

Din analiza celor două tabele de tranziţie ale stărilor rezultă că se generează cicluri succesive ce câte 15 stări, $(2^n - 1)$, starea $z_4z_3z_2z_1 = 0000$ nu este inclusă în ciclu. Succesiunea stărilor în cadrul ciclului are un caracter semi-aleatoriu, prin urmare circuitul poate fi considerat ca un generator de zgomot alb sub forma de scvenţe binare pseudo-aleatoare.

Pentru cazul general, de registru cu n celule, secvenţa binară pseudo-aleatoare generată are un ciclu de $(2^n - 1)$ stări. Ciclu maxim de $(2^n - 1)$ stări se generează numai atunci când polinomul caractristic al circuitului este primitiv, adică este un polinom ireductibil (prim, nu are divizori şi exponentul puterii este de valoare maximă pentru acel circuit). În literatură [Green ′85] sunt date polinoamele primitive până la gradul $n = 100$; în continuare sunt date aceste polinoame primitive până la gradul $n = 13$:

$I \oplus X \oplus X^2$	$I \oplus X \oplus X^6$	$I \oplus X^3 \oplus X^{10}$
$I \oplus X \oplus X^3$	$I \oplus X \oplus X^7$	$I \oplus X^2 \oplus X^{11}$
$I \oplus X \oplus X^4$	$I \oplus X \oplus X^5 \oplus X^6 \oplus X^8$	$I \oplus X^3 \oplus X^4 \oplus X^7 \oplus X^{12}$
$I \oplus X^2 \oplus X^5$	$I \oplus X^4 \oplus X^9$	$I \oplus X \oplus X^3 \oplus X^4 \oplus X^{13}$

Dacă polinomul nu este primitiv (admite divizori) circuitul corespunzător va avea un număr de cicluri egal cu divizorul cel mai mare al lui $(2^n - 1)$, fiecare din aceste cicluri având o secvenţă cu lungimea maximă $< 2^n - 1$; aceste cicluri fiind independente, odată circuitul intrat într-unul din aceste cicluri nu îl mai părăseşte decât numai prin resetare.

Se demonstrează că dacă polinomul caracteristic este primitiv atunci şi polinomul inversat este primitiv; dacă în polinomul caracteristic se substituie $X \to 1/X$ se obţine **polinomul inversat**. **Ordonarea (directă)** a polinomului caracteristic pe registru de deplasare se face în felul următor: $I(D^0)$ la intrarea primei celule; X la ieşirea lui (D); X^2 la ieşirea lui (D^2); $\ldots$;X^k la (D^k); $\ldots$; X^{n-1} la (D^{n-1}); X^n la (D^n). În opoziţie, **ordonarea (inversă)** a polinomului inversat pe registrul de deplasare se face în felul următor: X^n la intrarea primei celule (D^0); X^{n-1} la ieşirea lui (D); X^{n-2} la (D^2); X^{n-3} la (D^3); $\ldots$; X^k la (D^{n-k}); $\ldots$; X la (D^{n-1}); I la ieşirea lui D^n. Corespunzător polinomului caracteristic dat prin relaţia (3.56), polinomul inversat $X^4 \oplus X^3 \oplus I$, ordonat direct pe un registru de deplasare cu patru celule, produce o structurare de circuit secvenţial cu reacţie cu sumatoare exterioare ca în Figura 3.85-d şi produce şi o altă structurare cu sumatoare incluse ca în Figura 3.85-e. Rezultă că pentru un polinom primitiv există patru structurări: două cu sumatoare exterioare, una corespunde polinomului direct, $X^4 \oplus X \oplus I$, Figura 3.85-b, iar cealaltă corespunde polinomului inversat $X^4 \oplus X^3 \oplus I$, Figura 3.85-d; două cu sumatoare incluse, una corespunzătoare polinomului direct, Figura 3.85-c, iar cealaltă polinomului inversat, Figura 3.85-e. Deşi toate cele patru structurări provin din acelaşi polinom primitiv, fiecare din ele generează propria secvenţă cu lungimea maximă de 15 tacte (în general $2^n - 1$ tacte). Uzual, nu se mai calculează polinomul inversat iar pentru implementarea sa se utilizează tot polinomul direct dar se consideră ordonarea inversată.

Circuitul secvenţial liniar cu reaţie cu ciclul de lungimea maximă $(2^n - 1)$ nu cuprinde starea zero, toate celulele pline cu zero, deoarece prin reţeaua sumatoare modulo 2 de reacţie s-ar produce tot zero; în registru se injectează zero, deci circuitul rămâne în starea zero, închizându-se un ciclu permanent în jurul stării zero (nu poate fi amorsat). Pentru evitarea blocării, accidentale, în starea zero, la punerea circuitului

sub tensiune trebuie prevăzută setarea automată a unei celule în starea unu. (O astfel de soluţie constă într-un circuit de integrare RC a cărei intrare este tensiunea de alimentare a registrului (V_{DD}, V_{CC}) iar ieşirea acestui circuit se conectează la intrarea asincronă PRESET a celulei în care se va înscrie 1 şi la intrarea asincronă CLEAR a tuturor celorlalte celule care se vor înscrie în zero.)

Se poate extinde ciclul circuitului secvenţial liniar cu reacţie la 2^n dacă se introduce ca stare normală şi starea zero. Pentru această extensie, intrarea unei porţi XOR, din reţeaua de reacţie, se obţine ca ieşire de la o poartă NOR cu $n-1$ intrări, aceste intrări sunt colectate de la ieşirile tuturor celulelor registrului de deplasare mai puţin ultima (D^n). Dacă circuitul este în starea zero, $z_n z_{n-1} \ldots z_1 z_0 = 00 \ldots 00$, ieşirea porţii NOR generează un 1 prin care reţeaua de reacţie cu XOR, la următorul impuls de ceas, va injecta 1 în prima celulă şi/sau într-o altă celulă a registrului de deplasare (când sumatoarele sunt incluse). Dacă circuitul este în starea $z_n z_{n-1} \ldots z_1 z_0 = 10 \ldots 00$ ieşirea 1 a porţii NOR, plus $D^n = 1$, prin reţeaua de reacţie va injecta 0 în registrul de deplasare, deci următoarea stare este $z_n z_{n-1} \ldots z_1 z_0 = 00 \ldots 00$. Se obţine atât extensia la 2^n stări cât şi amorsarea din starea zero. Pentru structurile din Figura 3.85-d şi 3.85-e sunt desenate punctat conexiunile pentru aceste circuite de extensie.

Aplicaţiile circuitelor secvenţiale liniare cu reacţi sunt numeroase:

- generator de secvenţe binare pseudo-aleatoare pentru generare de stimuli în testarea circuitelor;
- circuite de codificare şi decodificare pentru detectarea şi corectarea erorilor, tehnica comunicaţiilor;
- numărătoare modulo $2^n - 1$ sau 2^n (pot fi considerate numărătoare în cod arbitrar, 3.4.2.2). Consumă cea mai puţină suprafaţă decât oricare alt numărător cu excepţia celui asincron, dar acesta nefiind sincron ridică problema la interfaţare. Este mai rapid decât oricare numărător cu excepţia numărătorului Johnson, dar acesta este numai modulo $2n$. (Pentru structurarea cu sumatoare incluse perioada de ceas minimă este egală cu propagarea printr-o celulă D plus o poartă XOR.)

3.5.7 Distribuţia şi aplicarea semnalului de ceas

Într-un sistem digital sincron semnalul de ceas (tactul) este utilizat pentru a introduce o referinţă de timp faţă de care se raportează realizarea unei funcţii în sistem. Deoarece acestă referinţă este vitală, pentru funcţionare sistemului sincron, trebuie acordată o deosebită atenţie parametrilor semnalului de ceas şi a reţelei de distribuţie, care este suportul fizic pentru aplicarea semanalului de ceas. Semnalul de ceas, uzual, este considerat ca un semnal de control, dar este un semnal de control cu caracteristici şi atribute speciale. Semnalele de ceas, tipic, asigură comanda unor sarcini mari, se propagă pe cele mai lungi trasee, operează la frecvenţele cele mai ridicate în raport cu oricare semnal de control sau de date din sistem. Deoarece semnalele de date sunt furnizate pe baza referinţei de timp, forma de variaţie în timp a semnalelor de ceas trebuie să fie "curată". Ori, păstrarea formei "curate" produsă de generatorul de ceas, până la punctele de aplicare, este puternic influenţată şi întârziată de către reţeaua de distribuţie. Proiectarea reţelei de distribuţie a semnalelor de ceas determină în mod esenţial performanţele sistemului digital mai ales la sistemele de viteză ridicată.

În concluzie, semnalul de ceas are o contribuţie majoră în obţinerea performanţelor şi funcţionării corecte pentru sistemele digitale sincrone.

În oricare sistem digital, cu procesare de tip clasic sau în pipeline, circuitele combinationale sunt interfaţate pe intrare şi pe ieşire cu elemente de memorare (latch-uri, bistabile, registre). Transferul din interfaţa de intrare, prin partea combinaţională, în interfaţa de ieşire se realizează între două tacte ale semnalului de ceas, a cărui perioadă minimă, T_{CLKmin}, se calculează cu relaţia 3.45. Structurarea clasică, Figura 3.76-a, pe baza căreia s-a dedus acestă relaţie este reluată şi în figura 3.86-a. De data acesta s-au introdus în relaţia pentru calculul timpului de propagare date, τ_{pD}, între cele două interfeţe şi timpul de propagare pe interconexiuni τ_{pInt}, obţinându-se relaţia:

$$\begin{aligned} \tau_{\text{pD}} &= \tau_{\text{pCQ}} + \tau_{\text{pCLC}} + \tau_{\text{SU}} + \tau_{\text{pInt}} \\ T_{CLKmin} &\geq \tau_{\text{pCQ}} + \tau_{\text{pCLC}} + \tau_{\text{SU}} + \tau_{\text{pInt}} \end{aligned} \tag{3.57}$$

Evident, τ_{pInt} la circuitele realizate cu componente discrete poate fi neglijat dar pentru sistemele integrate de mare viteză unde prin procesul de scalare, permanent, dimensiunile sunt multiplicate cu $1/s$ ($s > 1$, factorul de scalare, vezi Tabelul 1.11), această componentă a timpului de propagare devine din ce în ce mai puţin de neglijat. Timpii de propagare, dependenţi de dispozitiv: τ_{pCQ}, τ_{SU}, τ_{pCLC}, se micşorează prin scalarea dimensiunilor dar nu se micşorează şi τ_{pInt}. Micşorarea dimensiunilor traseelor de interconectare duce la micşorarea secţiunilor, la creşterea rezistenţei, deci nu la o micşorare a timpului de propagare τ_{pInt}, în plus apar capacităţi parazite (distribuite) şi efectul de electromigraţie (vezi secţiunea 1.5.1). Mai ales, pentru traseele din reţeaua de distribuţie a semnalului de ceas, scalarea se aplică cu foarte mare prudenţă, ori se păstrează aceleaşi dimensiuni (vezi secţiunea 4.5).

În structura din Figura 3.86 se consideră că există sincronizare/(suprapunere) între momentele de aplicare a semnalelor de ceas la cele două registre, $t_{CLKi} = t_{CLKj}$. În prctică, cele două momente nu se suprapun $t_{CLKi} \neq t_{CLKj}$; acestă nesincronizare are două componente: defazajul şi fluctuaţia fronturilor semnalelor de ceas.

Cauzele care duc la un defazaj de ceas (la întârzieri diferite) sunt reprezentate în Figura 3.86-b. De exemplu, cele două momente de aplicare ale semnalelor de ceas la bistabilele consecutive D_i, D_j, din lanţul de bistabile, pot să nu se suprapună, $t_{CLKi} \leq t_{CLKj}$ deoarece din motiv de încărcare cele două semanle se obţin de la două ramuri diferite ale reţelei de distribuţie a semnalului de ceas, iar pe cele două ramuri întârzierile nu sunt egale. Dacă se notează cu t timpul la **generatorul de ceas general**, **GCLK** (un generator PLL, Phase Lock Loop), iar întârzierile de propagare prin reţeaua de distribuţie, ale celor două semnale de ceas, până la D_i şi D_j respectiv cu τ_{i} şi τ_{j} rezultă momentele în timp $t_{CLKi} = t + \tau_{\text{i}}$ şi $t_{CLKj} = t + \tau_{\text{j}}$.

Definiţia 3.11 Defazajul de ceas τ_{df} (clock skew), între două registre/bistabile i, j, consecutive în sensul de propagare a datelor pentru procesare, este diferenţa în timp între momentul aplicării semnalului activ de ceas la registrul/bistabilul j şi momentul aplicării semnalului activ de ceas la registrul/bistabilul i $\diamond$

$$\tau_{\text{df}} = t_{CLKj} - t_{CLCi} = \tau_{\text{j}} - \tau_{\text{i}} \tag{3.58}$$

De notat faptul că defazajul de ceas este relevant numai între două registre/bistabile consecutive, deci între care există un transfer de date.

τ

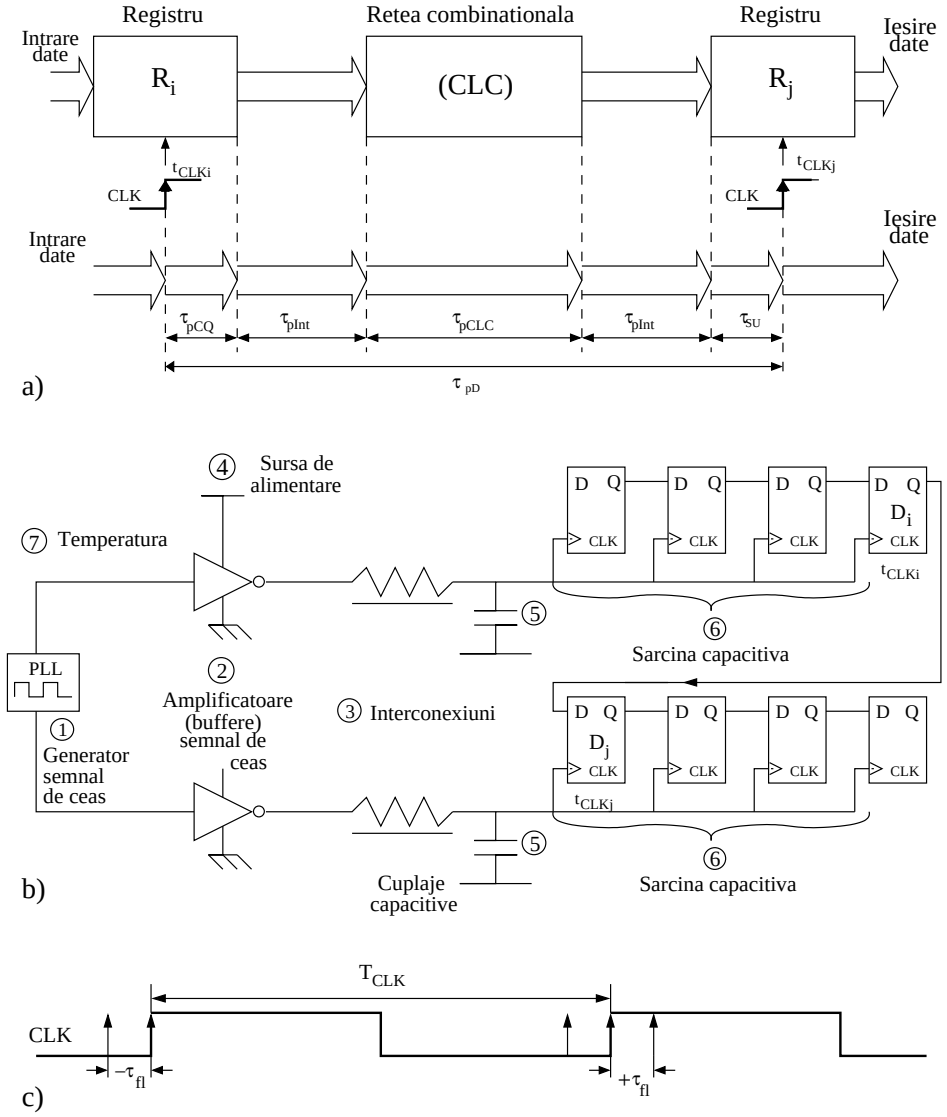


Figura 3.86 Nesincronizarea semnalelor de ceas între două registre/bistabile consecutive dintr-o cale de propagare a datelor: a) structură de principiu a unei etape în propagarea datelor cu evidenţierea timpilor (componente) de propagare; b) reprezentarea cauzelor, într-o reţea de distribuţie a semnalelor de ceas, care pot determina defazaj de ceas şi fluctuaţia fronturilor; c) explicativă pentru apariţia efectului de fluctuaţie al fronturilor (clock jitter).

Cauzele care pot duce ca întârzierile de propagare τ_{j} şi τ_{i} să nu fie egale sunt:

- Bufferele (amplificatoarele) de semnal (2) din reţeaua de distribuţie. Aceste amplificatoare sunt o cauză principală pentru $\tau_{\text{df}} \neq 0$. Datorită dispersiei din procesul de fabricaţie bufferele rezultă cu variaţii pentru valorile timpilor de propagare (care se păstrează chiar daca sunt incluse în reţele echilibrate şi cu încărcări identice).

- Interconexiunile (3) din reţeaua de distribuţie a ceasului. Pentru fiecare interconexiune intervine lungimea şi rezistivitatea traseului, constanta şi grosimea dielectricilor, rezistenţele de contact şi ale găurilor de pătrundere (vias) între straturile metalizate, capacităţile parazite.

- Variaţia valorii tensinii de alimentare (4)(vezi Figura 1.78).

- Cuplajele capacitive (5). Se recomandă ca stratul metalizat pentru reţeaua de distribuţie a semnalului de ceas să fie între straturile metalizate pentru V_{DD} şi V_{SS} (sursă şi masă).

- Sarcini de comandă (6)diferite pentru cele două semnale de ceas.

- Variaţia temperaturii (7). Defazajul de ceas are o valoare constantă în timp dacă temperatura mediului nu se modifica pronunţat.

A doua componentă care contribuie la nesincronizare este **efectul de fluctuaţie** (**în apariţie**) **al fronturilor** semnalelor semnalelor de ceas (clock jitter). Ca o consecinţă a acestui efect fronturile consecutive apar când la intervale de timp mai mari decât T_{CLK}, când la intervale mai mici decât T_{CLK}, Figura 3.86-c. Dacă se notează pentru semnalul de ceas cu perioada T_{CLK} abaterile de apariţie (fluctuaţiile) ale fronturilor semnalelor de ceas, faţă de momentul corect de apariţie al frontului, cu $-\tau_{\text{fl}}$ şi $+\tau_{\text{fl}}$ atunci perioada reală a semnalului de ceas poate avea valori în intervalul $[T_{CLK} - 2\tau_{\text{fl}}, T_{CLK} + 2\tau_{\text{fl}}$. Cauzele acestor fluctuaţii sunt: generatorul de ceas (1); încărcarea capacitivă (6) şi cuplajele capacitive (5); variaţiile de temperatură (7) şi în tensiunea de alimentare (4).

În funcţionarea reală a sistemelor semnalul de ceas aplicat la două registre/bistabile consecutive poate fi afectat simultan atât de defazaj cât şi de fluctuaţia fronturilor. Defazajul este o variaţie spaţială reflectată în momentele de aplicare a semanlelor de ceas şi nu se modifică de la perioadă la perioadă pe când fluctuaţia fronturilor, se poate modifica de la ciclu la ciclu, este un efect pasager.

Transferul sincron al datelor de la registrul R_i la registrul următor R_j trebuie asigurat atât pentru valoarea maximă a timpului de propagare al datelor τ_{pDmax} cât şi pentru timpul de propagare minim τ_{pDmin} în condiţiile în care intervin factorii de nesincronizare τ_{df} şi τ_{fl}. Valoarea τ_{pDmax} determină T_{CLKmin}, adică frecvenţa maximă a semnalului de ceas, deci performanţa de viteză. Valoarea τ_{pDmin} determină transferul corectat al datelor în registrul R_j, deci funcţionarea corectă a sistemului. Dacă timpul de propagare minim, de la registrul R_i la R_j, este mai mc decât timpul de menţinere τ_{H} al registrului R_j, $\tau_{\text{pDmin}} < \tau_H$, adică datele înscrise în registrul R_i ajung la registrul

R_j înainte ca datele anterioare aplicării frontului activ de ceas să fi fost înscrise în registrul R_j, în registrul R_j se înscriu datele de la registrul R_i; deci pe un singur tact datele parcurg două etape în calea de transfer (înscriere în registrul R_i şi propagare la registrul R_j precum şi înscriere în registrul R_j pentru că timpul de menţinere τ_{H} de la registrul R_j nu s-a consumat). Pentru a evalua modul cum se poate asigura atât performanţa cât şi funcţonarea corectă a sistemului sincron se vor analiza tipurile de defazaj de ceas.

Defazajul de ceas pozitiv, $\tau_{\mathbf{df}} > 0$. O valoare pozitivă pentru τ_{df}, conform relaţiei 3.53, apare când frontul activ de ceas la registrul R_j se aplică cu un interval de timp τ_{df} după aplicare aceluiaşi front activ de ceas la registrul R_i, $t_{CLKi} < t_{CLKj}$.

Practic, o astfel de situaţie în aplicarea semnalelor de ceas poate apare când sensul de aplicare al semnalelor de ceas coincide cu sensul de deplasare al datelor şi între registrele consecutive R_i şi R_j, pe traseul de ceas, apare o întârziere τ_{df}, Figura 3.87-a.

În prezenţa defazajului τ_{df} pozitiv, pentru întârzierea minimă în propagarea datelor τ_{pDmin}, înscrierea corectă in registrul R_j este asigurată numai când se respectă relaţia $\tau_{\mathrm{df}} + \tau_{\mathrm{H}} \leq \tau_{\mathrm{pDmin}}$, adică:

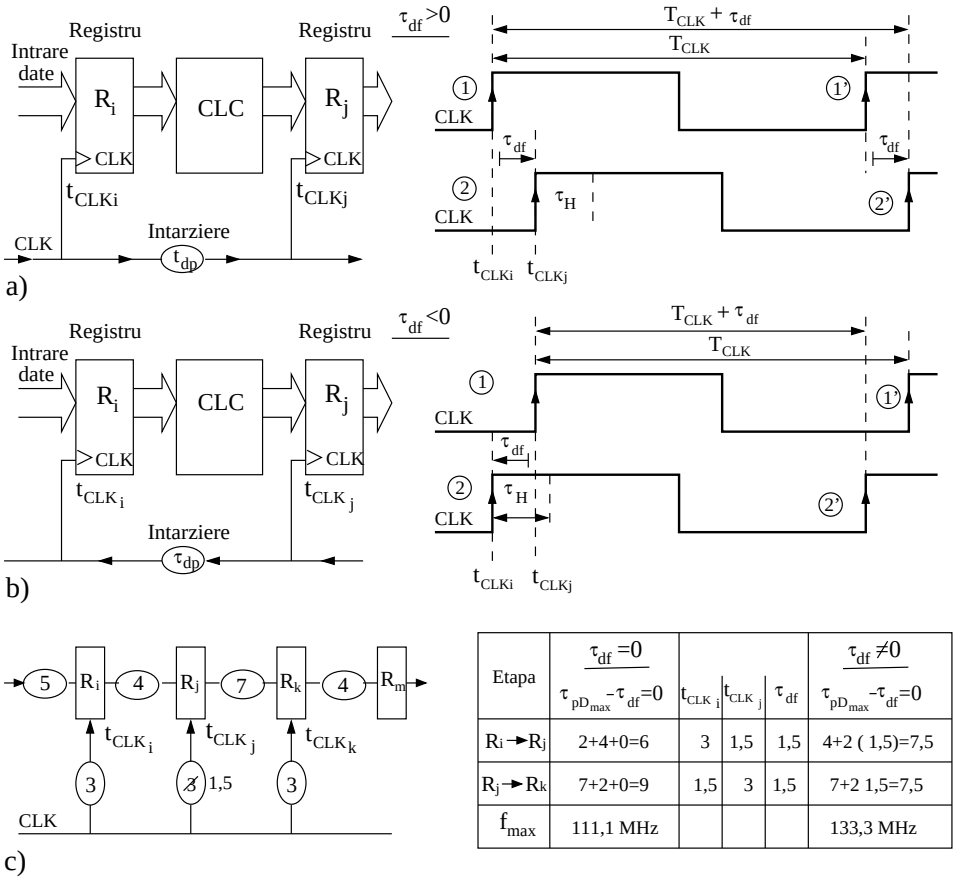
$$\tau_H \leq \tau_{\mathrm{pDmin}} - \tau_{\mathrm{df}} = \tau_{\mathrm{pCQ}} + \tau_{\mathrm{pCLC}} + \tau_{\mathrm{pInt}} - \tau_{\mathrm{df}} \tag{3.59}$$

Această relaţie când defazajul devine semnificativ, prin diferenţa $\tau_{\mathrm{pDmin}} - \tau_{\mathrm{df}}$ de valoare mică, poate impune pentru timpul de menţinere τ_{H} să aibă o valoare mai mică decât valoarea minimă prescrisă unui bistabil D pentru o funcţinare corectă (evitarea metastabilităţii, Figura 3.46). Deci creşterea defazajului poate determina ca funcţionarea să ajungă la limita critică pentru care valoarea prescrisă pentru τ_{H} să fie $\tau_{\mathrm{H}} \leq \tau_{\mathrm{pDmin}} - \tau_{\mathrm{df}}$, deci o înscriere incorectă a bistabilului R_j. Inegalitatea exprimată de relaţia 3.59 poate ajunge la limita critică pentru circuitele care cascadează bistabile, cum sunt structurile de registre de deplasare sau de numărătoare; la aceste circuite $\tau_{\mathrm{pCLC}} = 0$ (în general), $\tau_{\mathrm{pInt}} \to 0$ (prin geometria de proiectare), deci se poate ajunge la $\tau_{\mathrm{H}} \leq \tau_{\mathrm{pCQ}} - \tau_{\mathrm{df}}$. Un registru sau un numărător care funcţionează la limita acestei inegalităţi poate avea o funcţionare corectă la testare dar, în utilizare, datorită variaţiilor de mediu (temperatură, tensiune), poate inversa inegalitatea. Pentru o funcţionare sigură a acestor tipuri de circuite se recomandă o comandă cu un defazaj de ceas negativ.

În prezenţa defazajului τ_{df} pozitiv de valoare mult mai mică decât τ_{pD} pericolul înscrierii eronate nu poate apare dar, în schimb, la o valoare mare a timpului de propagare a datelor aceasta trebuie corelată cu perioada minimă impusă pentru semnalul de ceas. Din diagrama de semnale din figură rezultă că durata perioadei de ceas plus timpul de defazaj, $T_{CLK} + \tau_{\mathrm{df}}$, nu poate fi mai mică dacât timpul de propagare a datelor plus timpul de stabilizare al registrului R_j, $T_{CLK} + \tau_{\mathrm{df}} \geq \tau_{\mathrm{pD}} + \tau_{\mathrm{SU}} = \tau_{\mathrm{pDmax}}$. Din această relaţie rezultă valoarea minimă a prioadei de ceas:

$$T_{CLKmin} \geq \tau_{\mathrm{pDmax}} - \tau_{\mathrm{df}} = \tau_{\mathrm{pCQ}} + \tau_{\mathrm{pCLC}} + \tau_{\mathrm{pInt}} + \tau_{\mathrm{SU}} - \tau_{\mathrm{df}} \tag{3.60}$$

De notat faptul că, faţă de cazul de inexistenţă a defazajului, $\tau_{\mathrm{df}} = 0$, când comanda se face cu o perioadă minimă dată de relaţia 3.57, în prezenţa defazajului de ceas pozitiv, prin relaţia 3.60, comanda se poate realiza cu o perioadă de ceas mai mică, deci o frecvenţă mai ridicată (un efect benefic al defazajului pozitiv! , vezi Exemplul 3.31).



Etapa	$\tau_{df}=0$ $\tau_{pD_{max}}-\tau_{df}=0$	$t_{CLK\ i}$	$t_{CLK\ j}$	τ_{df}	$\tau_{df}\neq 0$ $\tau_{pD_{max}}-\tau_{df}=0$
$R_i \rightarrow R_j$	2+4+0=6	3	1,5	1,5	4+2 (1,5)=7,5
$R_j \rightarrow R_k$	7+2+0=9	1,5	3	1,5	7+2 1,5=7,5
f_{max}	111,1 MHz				133,3 MHz

Figura 3.87 Defazajul de ceas: a) defazajul de ceas pozitiv (poate fi generat de structuri la care coincide sensul fluxului de date cu cel de aplicare al semnalelor de ceas); b) defazajul de ceas negativ (cele două sensuri sunt opuse); c) exempificare de utilizare a defazajului pozitiv pentru mărimea frecvenţei se sincronizare într-o cale de date.

Defazajul de ceas negativ, $\tau_{\mathbf{df}} < 0$. Defazajul negativ de ceas apare când frontul activ se aplică întâi la R_j şi apoi la R_i, adică $t_{CLKj} < t_{CLKi}$. O astfel de comandă poate apare când sensul de aplicare al semnalelor de ceas este opus sensului de deplasare al datelor şi între registrele consecutive R_i şi R_j, pe traseul de ceas, apare întârzierea τ_{df}, Figura 3.87-b . Relaţiile pentru valoarea critică a timpului de stabilizare τ_{H} şi a perioadei minime de ceas T_{CLKmin}, se deduc prin acelaşi raţionament ca şi pentru $\tau_{\text{df}} > 0$, sunt respectiv (3.59) şi (3.60) cu diferenţa că pentru calcul se introduce o valoare negativă pentru defazaj. Se observă că defazajul negativ, în raport cu cel pozitiv, este mai puţin restrictiv pentru timpul de menţinere τ_{H} (poate avea valori mai mari care asigură o înscriere corectă a datelor), dar, în schimb, măreşte valoarea pentru T_{CLKmin} (ceea ce poate fi o limitare în sistemele de viteză rdicată).

În concluzie, creşterea defazajului pozitiv are ca efect micşorarea perioadei minime de ceas aplicabile dar poate duce la o degradare a înscrierii corecte (funcţionării sigure), pe când crşterea defazajului negativ, în valoare absolută, are ca efect mărirea perioadei minime de ceas aplicabile dar cu o înbunătăţire pentru înscrierea corectă. Dar într-o cale de transfer de date fiecare registru/bistabil R_j, în afară de primul şi ultimul, realizează o etapă cu următorul R_k şi o altă etapă cu cel anterior R_i ($i < j < k$). Considerând defazaj zero în cele două etape, la mărirea/micşorarea timpului de aplicare a semnalului de ceas t_{CLKj} la registrul R_j se va genera un defazaj negativ/pozitiv în etapa $R_j \rightarrow R_k$ şi un defazaj pozitiv/negativ în etapa $R_i \rightarrow R_j$, deci efectele care apar, prin aceasta, asupra perioadei de ceas aplicabile şi asupra funcţionării sigure vor fi în sensuri opuse în cele două etape. Rezultă că modificarea timpului t_{CLKj}, fără modificarea timpilor t_{CLKi} şi t_{CLKk}, poate modifica, în sens contrar, valoarea minimă, T_{CLKmin}, aplicabilă în cele două etape vecine $R_i \rightarrow R_j$ şi $R_j \rightarrow R_k$. Această concluzie poate fi utilizată pentru o metodă de optimizare globală a unei căi de date printr-o optimizare/modificare locală ("deskewing data pulses", "cycle stealing"); metodă aplicabilă când în calea de date există diferenţe mari între valorile timpilor de propagare τ_{pD} din două etape consecutive, dar să concretizăm printr-un exemplu.

Exemplul 3.31 În Figura 3.87-c este prezentat un segment dintr-o cale de date, cu cele trei registre R_i, R_j, R_k, cu specificarea timpilor de propagare maximă din fiecare etapă din calea de date, prin ovale orizontale şi prin ovale verticale, întârzierile (considerate egale) de pe fiecare traseu de aplicare a semnalelor de ceas. Frecvenţa maximă de ceas, cu care se poate sincroniza această cale de date (considerând că registrele sunt identice, cu $\tau_{\text{pCQ}} = 2ns$), este impusă de etapa cu propagarea maximă, adică etapa $R_j \rightarrow R_k$ unde $\tau_{pDmax} = 7ns + 2ns = 9ns$, deci $f_{max} = 1/9ns = 111.1MHz$.

Dacă numai timpul de aplicare al semnalului de ceas t_{CLKj}, la registrul R_j, este micşorat de la $3ns$ la $1.5ns$, prin ajustarea întârzierii de pe traseul respectiv, apare, conform relaţiei 3.58, un defazaj pozitiv pentru etapa $R_j \rightarrow R_k$, $3ns - 1.5ns = 1.5ns$, şi un defazaj negativ pentru etapa $R_i \rightarrow R_j$, $1.5ns - 3ns = -1.5ns$. Aceasta înseamnă, conform relaţiei 3.60, că perioada minimă de ceas aplicabilă în etapa $R_j \rightarrow R_k$ poate fi micşorată, $2ns+7ns-1.5ns = 7.5ns$, iar în etapa $R_i \rightarrow R_j$ trebuie să fie mărită, $2ns + 4ns - (-1.5ns) = 7.5ns$, deci se ajunge la o egalitate între perioadele de ceas minime aplicabile în cele două etape vecine; rezultă o frecvenţă maximă de comandă $f_{max} = 133.3MHZ$ (o creştere cu 19.98%). Prin micşorarea timpului t_{CLKj} de la $3ns$ la $1.5ns$ nu se afectează înscrierea corectă a datelor în registrul R_k; în etapa $R_j \rightarrow R_k$ defazajul este pozitiv dar valoarea acestuia este mult mai mică decât timpul maxim de propagare ($\tau_{\text{df}} = 1.5ns < 9ns = \tau_{pDmax}$), deci nu devine critică

valoarea impusă pentru τ_H, vezi relaţia 3.59.

Această modalitate de modificare locală într-o cale de date, dar cu efect de optimizare globală deoarece creşte frecvenţa maximă pentru GCLK, depinde de posibilitatea de modificare în circut a întârzierii pentru a obţine un τ_{df} ajustabil şi evident, de variaţiile de mediu (temperatură şi tensiune).

Reţeaua de distribuţie a semnalelor de ceas. Mulţimea de trasee de la sursa de ceas general, GCLK, până la punctele de aplicare a semnalelor de ceas pentru sincronizare formează reţeaua de distribiţie. Performanţele care se urmăresc, pentru o reţea, şi care sunt fundamentale pentru funcţionarea corectă a circuitului sunt: timpul de propagare (întârzierea introdusă) de la GCLK până la punctele de aplicare, adică sincronizarea; defazajul de ceas şi puterea disipată. Există două modalităţi, de structurare a unei reţele: de arbore bufferat şi de arbore simetric.

Reţeaua de distribuţie sub formă de arbore bufferat are, prin analogie o structură de arbore, un traseu principal pornind din sursa de GCLK (rădăcină) care apoi se ramifică pentru fiecare registru, acestea fiind "frunzele", Figura 3.88-a. Pe acest traseu de la rădăcină la frunze se introduc buffere care au rolul de amplificare şi de izolare. Numărul de buffere înseriate depinde de sarcina capacitivă totală (interconexiuni şi punctele de sincronizare) ce trebuie comandată. Ieşirea buferului trebuie să genereze un curent suficient de încărcare şi descărcare al capacităţilor într-un timp scurt pentru ca semnalele de ceas să prezinte fronturi abrupte necesare procesului de sincronizare; acestă cerinţă impune ca bufferul să aibă o funcţionare de generator de curent, rezistenţa sa de ieşire să fie mult mai mare decât rezistenţa conexiunilor reţelei comandate (restricţie ce poate fi îndeplinită relativ uşor dacă reţeua de distribuţie este realizată în strat metalizat). Dar bufferele sunt şi principala sursă de introducere de defazaj deoarece caracteristicile elementelor active din reţea (buffere) variază mult mai pronunţat în raport cu caracteristicile elementelor pasive (interconexiuni) atât datorită procesului de fabricaţie cât şi datorită mediului.

Uneori pentru a micşora rezistenţa interconexinilor, şi a uniformiza disipaţia de putere pe toată suprafaţa de integrare, reţeaua de distribuţie este realizată sub forma unei plase de trasee pe o suprafaţă cât mai mare, în figură este prezentată, ca medalion, o structurare de principiu pentru o plasă. Pentru reducerea controlată a puterii disipate în reţeaua de distribuţie sunt introduse porţi care, comandate cu un semnal de condiţionare, pot să elimine alimentarea cu un semnal de ceas a anumitor zone locale pentru anumite intervale de timp.

A doua structură de **reţea de distribuţie** pentru semnalele de ceas poate fi considerată tot un arbore dar **sub formă de arbore simetric**, echilibrat, având forma de H sau X, Figura 3.88-b; evident fiecare vârf H sau X se poate continua cu un alt H sau X şi acesta să se continue cu un alt H sau X ş.a.m.d. Bifurcarea traseelor, din cel anterior, duce şi la înjumătăţirea lăţimii acestor trasee pentru ca în propagarea semnalelor să se evite reflexia în punctele de bifurcaţie. Această structurare realizează ca atingerea oricărei frunze să se facă prin trasee de aceeaşi lungime şi secţiune, obţinându-se întârzieri egale faţă de GCLK, deci, teoretic, un defazaj de ceas nul. În raport cu arborele bufferat, arborele simetric H trebuie să comande o capacitate mai mare deoarece traseele necesare sunt mai lungi. În plus, arborele H este mai puţin potrivit pentru VLSI care au un layout mai puţin simetric. Combinaţia între cele două tipuri de structuri pare a fi compromisul indicat: o reţea H, pentru distribuţia

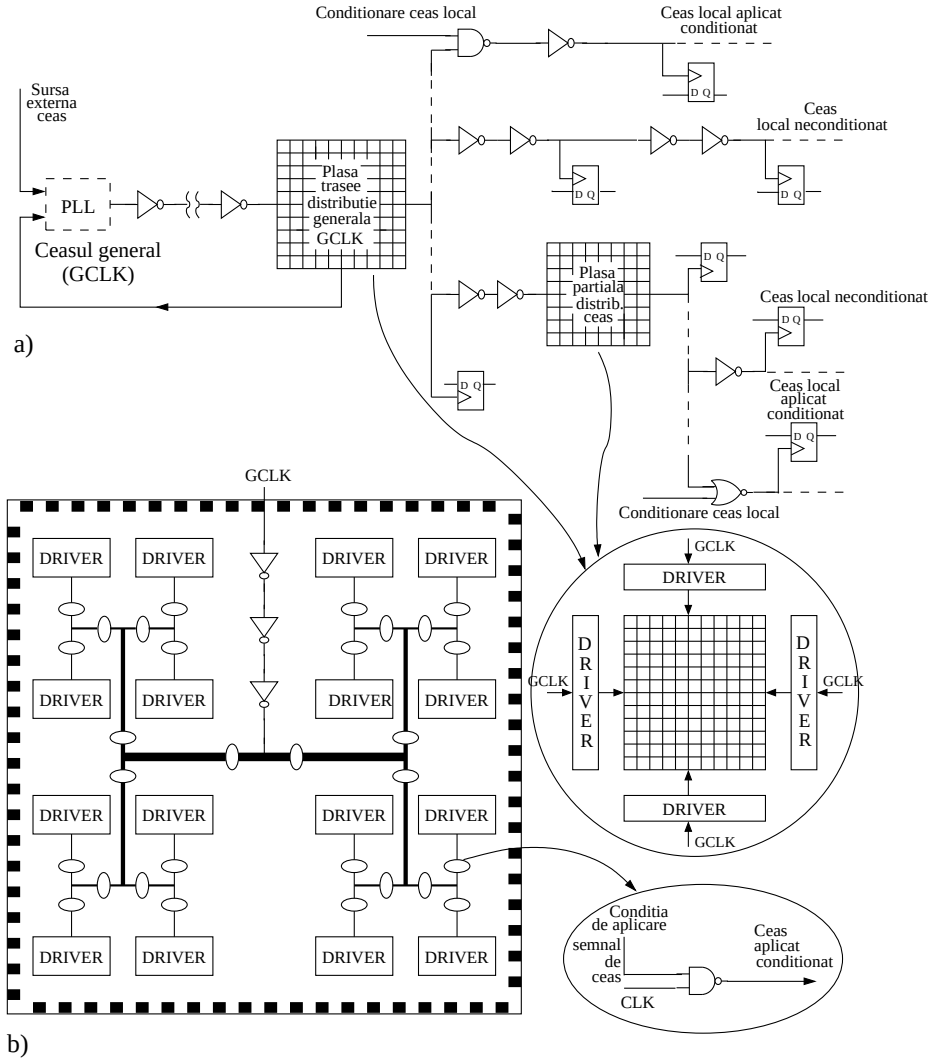


Figura 3.88 Modalităţi de structurare a reţelei de distribuţie pentru semnalul de ceas: a) sub forma unei reţele de arbore bufferat; b) sub forma unei reţele de arbore simetric H.

globală de semnal de ceas, continuată din vârfuri cu reţele bufferate pentru distribuţia locală.

În circuitele VLSI actuale reţeaua de distribuţie a semnalului de ceas ajunge să comande zeci de mii de registre, deci a unei sarcini capacitive ce se apropie de $100cm$! Într-o reţea de distribuţie de ceas fiecare tranziţie schimbă starea în fiecare nod capacitiv, spre deosebire de o reţea combinaţională unde activitatea de comutaţie a elementelor este determinată de funcţia logică. Combinaţia de sarcină capacitivă mare, C_L, şi frecvenţa ridicată determină o valoare ridicată pentru componenta dinamică a puterii disipate, $V_{DD}^2 \cdot C_L \cdot f$; la unele procesoare actuale mai mult de 30% din puterea consumată se regăseşte în puterea disipată în reţeaua de distribuţie a ceasului. Soluţiile pentru reducerea componentei dinamice a puterii disipate sunt: 1 - micşorarea tensiunii de alimentare (implementări la $\frac{1}{2}V_{DD}$ numai pentru alimentarea reţelei de ceas, prin aceasta degradarea performanţei de viteză este nesemnificativă); 2 - micşorarea capacităţii echivalente totale, printr-o proiectare adecvată, (3 - micşorarea frecvenţei se exclude deoarece viteza este o cerinţă a majorităţii circuitelor VLSI). De asemenea, pentru nealimentarea temporară locală, există porţi care comandă condiţionat aplicarea semnalului de ceas (vezi medalionul de la Figura 3.88-b).

Pentru circuitele VLSI actuale, care lucrează la frecvenţe de peste $1GHz$, realizarea distribuţiei semnalului de ceas şi menţinerea puterii disipate în limita de siguranţă sunt două aspecte ce ridică dificultăţi. La valori $f_{CLK} > 1GHz$, perioada semnalului de ceas T_{CLK} se micşorează până la valori care nu depăşesc timpul de propagare, $\tau_{\text{p poartă}}$, prin 10 poţi logice, $T_{CLK} < 10\tau_{\text{p poartă}}$, iar mărimea defazajului de ceas trebuie menţinut sub $30ps$ ($1ps = 10^{-12}s$) [Friedman '01]. Obţinerea acestor mărimi reduse de defazaj ridică dificultăţi datorită faptului că valorile întârzierilor semnalului de ceas au variaţii greu de controlat. Aceste variaţii greu de controlat ale valorilor întârzierilor se datorează: 1 - procesul de fabricaţie nu poate asigura o dispersie care să ducă la obţinerea unor parametri cu abateri foarte strânse; 2 - variaţiile de mediu (temperatură, tensiune); 3 - utilizarea în proiectare a unor modele cu o precizie încă nesatisfăcătoare pentru schemele echivalente de circuit. Pentru frecvenţe peste $(1 \div 2)GHz$ şi efectul de linie de transmisie trebuie luat în considerare, deci pe lângă caracterul RC al sarcinii apare şi componenta de inductivitate L rezultând caracteristici de tip RLC. Caracterul RLC în funcţionare modifică puternic întârzierea semnalului şi puterea disipată (uneori aceasta poate chiar să descrească).

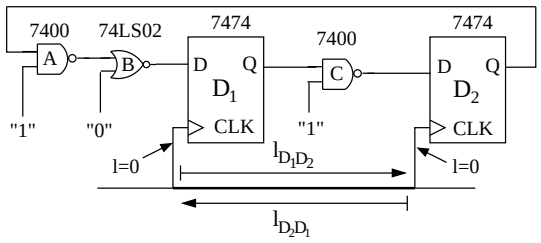
Pentru frecvenţe peste $1GHz$ structurarea reţelei de distribuţie a semnalelor de ceas cuprinde, în succesiune, următoarele trei componente: 1 - global, o reţea pentru distribuţia semnalului GCLK, la care se conectează; 2 - regional, o serie de circuite de compensare a defazajului (deskew circuits), fiecare dintre acestea comandă, printr-o reţea arbore echilibrat; 3 - local, o multitudine de buffere ce alimentează punctele de sincronizare de la registre. În plus, în paralel cu reţeua globală, GCLK, mai există (separat) încă o reţea care distribuie un semnal GCLK etalon, dar încărcarea acestei reţele este mult mai mică decât cea a reţelei globale.

Reţeaua globală, GCLK, structurată ca un arbore H, distribuie semnalul de ceas de la PLL până la circuitele regionale de compensare a defazajului. Pentru a minimiza efectele de cuplare capacitivă şi inductivă între liniile adiacente de semnal şi traseele reţelei GCLK, acestea, din urmă, sunt complet ecranate prin plasarea între linii de masă şi de alimentare V_{DD}. Componenta de compensare a defazajului care se bazează,

în principal pe un sistem de reglare constă din (1) un detector de fază, dintre semnalul GCLK etalon şi un semnal de reacţie cules local. Semnalul diferenţă de fază comandă digital o linie de întârziere analogică a cărei ieşire se aplică la (2) un buffer ce produce un semnal de ceas fără întârziere şi care se aplică la o (3) - reţea arbore echilibrat ce alimentează bufferele locale. De la un buffer local se distribuie semnale de sincronizare printr-o reţea de porţi comandate condiţionat (deci se poate reduce puterea disipată prin nealimentare temporară).

Toate sistemele electronice, fundamental, sunt de natură asincrone, totuşi printr-o precisă inserare a unei relaţii de temporizare locală în realizarea funcţiilor şi utilizarea elementelor de memorare un sistem asincron poate fi adaptat să aibă o funcţionare sincronă. Atâta timp cât relaţia de temporizare locală este îndeplinită (sincronizarea), modul de abordare sincron poate fi aplicat, implementarea de sisteme, controlabile în funcţionare, este posibilă. Oricum, sistemele sincrone vor fi încă mult timp modul comun de implementare până vor ceda locul implementărilor asincrone(care, în principiu, pot obţine viteze de procesare mai ridicate).

Exemplul 3.32 Pentru circuitul din Figura 3.89-a, în raport cu sensul de transfer prin poarta C în etapa D_1D_2, semnalul de ceas se aplică: a - în acelaşi sens; b - în sens opus. Pentru aceste două variante de aplicare a semnalului de ceas să se calculeze lungimea maximă a conductorului ($l_{D_1D_2}$ şi $l_{D_2D_1}$) care asigură un transfer de date corect între bistabile precum şi frecvenţa maximă a semnalului de ceas. Parametrii de timp pentru circuitele utilizate sunt daţi în Figura 3.89-b. Pentru viteza de propagare a semnalului se consideră valoare $v_p = 20cm/ns$.



Circuit	t_{pHL} [ns]			t_{pHL} [ns]		
	Min	Tipic	Max	Min	Tipic	Max
7400	–	11	22	–	7	15
74LS02	–	10	20	–	10	20
7474*	10	14	25	10	20	40

*τ_{SU}=20 [ns] , τ_H =5 [ns]

Figura 3.89 Explicativă pentru exemplul 3.32

Soluţie. Valorile maxime şi minime ale timpilor de propagare pentru transferul $D_1 \rightarrow D_2$ prin poarta C sunt:

$$\tau_{pD_1D_2(LH)max} = \tau_{pCQ(LH)max} + \tau_{pC(HL)max} = 25ns + 15ns = 35ns$$

$$\tau_{pD_1D_2(LH)min} = \tau_{pCQ(LH)min} + \tau_{pC(HL)min} = 10ns + 7ns = 17ns$$

$$\tau_{pD_1D_2(HL)max} = \tau_{pCQ(HL)max} + \tau_{pC(LH)max} = 40ns + 22ns = 62ns$$

$$\tau_{pD_1D_2(HL)min} = \tau_{pCQ(HL)min} + \tau_{pC(LH)min} = 10ns + 11ns = 21ns$$

de asemenea pentru transferul $D_2 \rightarrow D_1$ prin porţile A şi B se calculează:

$$\tau_{pD_2D_1(LH)max} = \tau_{pCQ(LH)max} + \tau_{pA(HL)max} + \tau_{pB(LH)max} = 25ns + 15ns + 20ns = 60ns$$

$$\tau_{pD_2D_1(LH)min} = \tau_{pCQ(LH)min} + \tau_{pA(HL)min} + \tau_{pB(LH)min} = 10ns + 7ns + 10ns = 27ns$$

$\tau_{\mathrm{pD_2D_1(HL)max}} = \tau_{\mathrm{pCQ(HL)max}} + \tau_{\mathrm{pA(LH)max}} + \tau_{\mathrm{pB(HL)max}} = 40ns + 22ns + 20ns = 82ns$

$\tau_{\mathrm{pD_2D_1(HL)min}} = \tau_{\mathrm{pCQ(HL)min}} + \tau_{\mathrm{pA(LH)min}}\tau_{\mathrm{pB(HL)min}} = 10ns + 11ns + 10ns = 31ns$

Pentru cele două variante a şi b aplicând relaţiile (3.59) şi (3.60) se obţin respectiv valorile lungimilor maxime şi frecvenţei maxime.

a) $t_{CLKD_1} < t_{CLKD_2} \rightarrow \tau_{\mathrm{dfD_1D_2}} > 0,\ \tau_{\mathrm{dfD_2D_1}} < 0,\ |\tau_{\mathrm{dfD_2D_1}}| = \tau_{\mathrm{dfD_1D_2}}$
$\tau_{\mathrm{dfD_1D_2max}} \leq \tau_{\mathrm{pD_1D_2(LH)min}} - \tau_{\mathrm{H}} = 17ns - 5ns = 12ns;\ l_{D_1D_2max} \leq 12ns \times 20cm/ns = 2.4m$

Defazajul negativ $\tau_{\mathrm{dfD_2D_1}} = -12ns$ pentru transferul $D_2 \rightarrow D_1$ nu pericliteză înscrierea corectă a datelor în D_1, dimpotrivă ajută!
$T_{CLKD_1D_2min} \leq \tau_{\mathrm{pD_1D_2(HL)max}} + \tau_{\mathrm{SU}} - \tau_{\mathrm{dfD_1D_2}} = 62ns + 20ns - 12ns = 90ns$
Iar pentru transferul $D_2 \rightarrow D_1$ prin porţile A şi B se calculează
$T_{CLKD_2D_1min} \leq \tau_{\mathrm{pD_2D_1(HL)max}} + \tau_{\mathrm{SU}} - \tau_{\mathrm{dfD_2D_1}} = 82ns + 20ns - (-12ns) = 114ns$
deci frecvenţa maximă $f_{max} \leq 1/114ns = 87.71MHz$ este limitată de acest transfer.

b) $t_{CLKD_2} < t_{CLKD_1} \rightarrow \tau_{\mathrm{dfD_2D_1}} > 0,\ \tau_{\mathrm{dfD_1D_2}} < 0,\ |\tau_{\mathrm{dfD_1D_2}}| = \tau_{\mathrm{dfD_2D_1}}$
$\tau_{\mathrm{dfD_2D_1}} \leq \tau_{\mathrm{pD_2D_1(LH)min}} - \tau_{\mathrm{H}} = 27ns - 5ns = 22ns;\ l_{D_2D_1max} \leq 22ns \times 20cm/ns = 4.4m$

Defazajul negativ $\tau_{\mathrm{dfD_1D_2}} = -22ns$, pentru transferul $D_1 \rightarrow D_2$ ajută la înscrierea corectă a datelor în bistabilul D_2.
$T_{CLKD_2D_1min} \geq \tau_{\mathrm{pD_2D_1(HL)max}} + \tau_{\mathrm{SU}} - \tau_{\mathrm{dfD_2D_1}} = 82ns + 20ns - 22ns = 80ns$
Iar pentru tansferul $D_1 \rightarrow D_2$ prin poarta C se calculează
$T_{CLKD_1D_2min} \geq \tau_{\mathrm{pD_2D_1(HL)max}} + \tau_{\mathrm{SU}} - \tau_{\mathrm{dfD_1D_2}} = 62ns + 20ns - (-22ns) = 104ns$
deci frecvenţa maximă $f_{max} \leq 1/104ns = 96.153MHz$ este limitată de acest transfer.

Frecvenţa maximă, pentru cazul când bistabilele D_1 şi D_2 se comandă sincron, este fixată de transferul cel mai lung $T_{CLKmin} \leq \tau_{\mathrm{pD_2D_1(HL)max}} + \tau_{\mathrm{SU}} = 82ns + 20ns = 102ns$, $f_{max} \leq 98.039MHz$.

3.6 MEMORIA CU ACCES ALEATORIU

Circuitul **RAM** (**R**andom-**A**ccess-**M**emory), ca şi circuitul ROM (2.4.6), este un suport pentru stocarea informaţiei sub formă de cuvinte binare. Spre deosebire de ROM, la RAM, pe lângă posibilitatea de citire a imformaţiei, există şi facilitatea de modificare a informaţiei, adică de înscriere. Referirea acestor circuite cu termenul de memorie reflectă o similitudine cu memoria naturală. Totuşi, procesul de extragere/regăsire a informaţiei în aceste circuite electronice, în comparaţie cu memoria naturală, este destul de diferit. La memoria naturală se regăseşte o informaţie pe baza unei mai mici informaţii ("un fir") care are o "conexiune" cu informaţia căutată, deci un proces de asociere. În aceeaşi abordare, putem spune că şi circuitele ROM sau RAM sunt o memorie pentru că regăsirea unei informaţii se face tot printr-o asociere, acest "fir" fiind o adresă, a locaţiei unde a fost stocată informaţia. Dar există şi circuite electronice la care regăsirea informaţiei se face prin asociere la aceasta a unei informaţii parţiale, similar memoriei naturale, acestea sunt memoriile de tip asociativ [Stefan ′00].

Termenul de memorie RAM exprimă faptul că accesul la oricare locaţie (random) se poate face fără nici o restricţie de timp indiferent care este adresa locaţiei accesate; posibilitate care este evidentă şi la circuitul ROM. Justificarea termenului de random (aleator) este de nuanţă istorică. Primele memorii cu citire şi înscriere aveau accesul serial, adică accesul la o locaţie nu putea fi realizat decât numai după parcurgerea

tuturor adreselor locaţiilor anterioare; un exemplu curent de memorie serială este caseta magnetică unde accesul la o anumită informaţie (date, sunet sau imagine) se face prin derularea benzii până la poziţia respectivă. Un alt exemplu de memorie serială este registrul inel la care înscrierea sau citirea unui bit se face prin recicularea cuvântului în inel până când poziţia din cuvântul respectiv ajunge să fie transferată din ultima celulă în prima celulă. La apariţia memoriei cu acces aleator, această caracteristică de accesare la oricare locaţie fiind un salt deosebit din punct de vedere al vitezei de acces, în raport cu memoriile cu acces serial, s-a imprimat în denumire (abreviaţia) RAM, care a devenit un termen utilizat în exclusivitate. Denumirea corectă ar fi de memorie RWM (Read/Write Memory).

Spre deosebire de circuitul ROM memoria RAM este de tip volatil, adică la pierderea tensiunii de alimentare informaţia stocată "se volatilizează". Această volatilitate se datorează faptului că stocarea unui bit are ca suport fizic o stare a unui circuit, stare ce dispare la nealimentarea circuitului, şi nu prezenţa sau absenţa unui element fizic (fuzibil, diodă, tranzistor) este suportul fizic, cum apare la circuitul ROM.

Structurarea de principiu pentru memoria RAM poate fi realizată printr-o extensie a celei pentru memoria ROM,Figura 2.49-a. La fel ca şi la ROM se adoptă, pe suprafaţa de Si, pentru memoria RAM, o formă pătratică sau o formă dreptunghiulară apropiată de un pătrat. De exemplu, pentru o memorie RAM de capacitate $2^n \times m$ biţi (n biţi de adrese şi un cuvânt de date de m biţi pe I/O), din cuvântul de adresare, $A_{n-1}A_{n-2}\dots A_1A_0$, subcuvântul de n_2 biţi, $A_{n-1}A_{n-2}\dots A_{n_1+1}A_{n_1}$, se aplică decodificatorului $DCDn_2 : 2^{n_2}$ pentru activarea **liniilor de cuvânt**, iar subcuvântul de n_1 biţi, $A_{n_1-1}A_{n_1-2}\dots A_1A_0$, se aplică pentru selectarea coloanelor, $n_1 + n_2 = n$, Figura 3.90-a . Valorile care duc la o formă pătratică se calculează cu relaţiile $n_2 = (n + k)/2$, $n_1 + k = (n + k)/2$; lungimea cuvântului de date, m, este în general o putere a lui doi, $m = 2^k$.

Pe fiecare linie de cuvânt activată prin una din cele 2^{n_2} ieşiri de la $DCDn_2 : 2^{n_2}$ există $m \cdot 2^{n_1}$ celule, adică **linii (coloane) de bit**, care formează 2^{n_1} cuvinte, fiecare cuvânt cu lungimea de m biţi. La aplicarea subcuvântului de n_1 biţi se selectează simultan, din linia activată de DCD, m linii de bit, selectare care se poate realiza cu un grup de $m \times MUX2^{n_1} : 1$. Deoarece cele m linii de bit (celule) trebuie să fie atât înscrise cât şi citite impune pentru cuvântul de date, $D_{m-1}D_{m-2}\dots D_1D_0$, să poată fi aplicat la terminalele de I/O în ambele sensuri. Acest dublu sens de transfer al datelor pe I/O impune pentru citire selectarea liniilor de bit să se realizeze cu un grup de $m \times MUX2^{n_1} : 1$; iar pentru înscriere selectarea să se realizeze cu un grup de $m \times DMUX1 : 2^{n_1}$; în practică selectările pentru cele două sensuri sunt incluse într-un grup de $m \times (MUX/DMUX)$. Realizarea acestei duble selectări se bazează pe elemente care pot conduce în ambele sensuri: tranzistoare de trecere sau porţi de transmisie.

Transferul în ambele sensuri, ale grupului $m \times (MUX/DMUX)$, poate fi realizat de $m \times MUX2^{n_1} : 1$ cu o structură arborescentă cu tranzistoare de trecere, Figura 2.35-b. Numărul tranzistoarelor de trecere pe oricare ramură, în sensul de la I/O la o linie de bit, sau în sens invers, este egal cu n_1. Această structură arborescentă de MUX, când n_1 are valoare ridicată, poate duce la valori mari pentru timpul de transfer al datelor la memorie, deoarece timpul de transfer pe o ramură între linia de bit şi I/O este proporţional cu rezistenţa echivalentă a tranzistoarelor pe acea

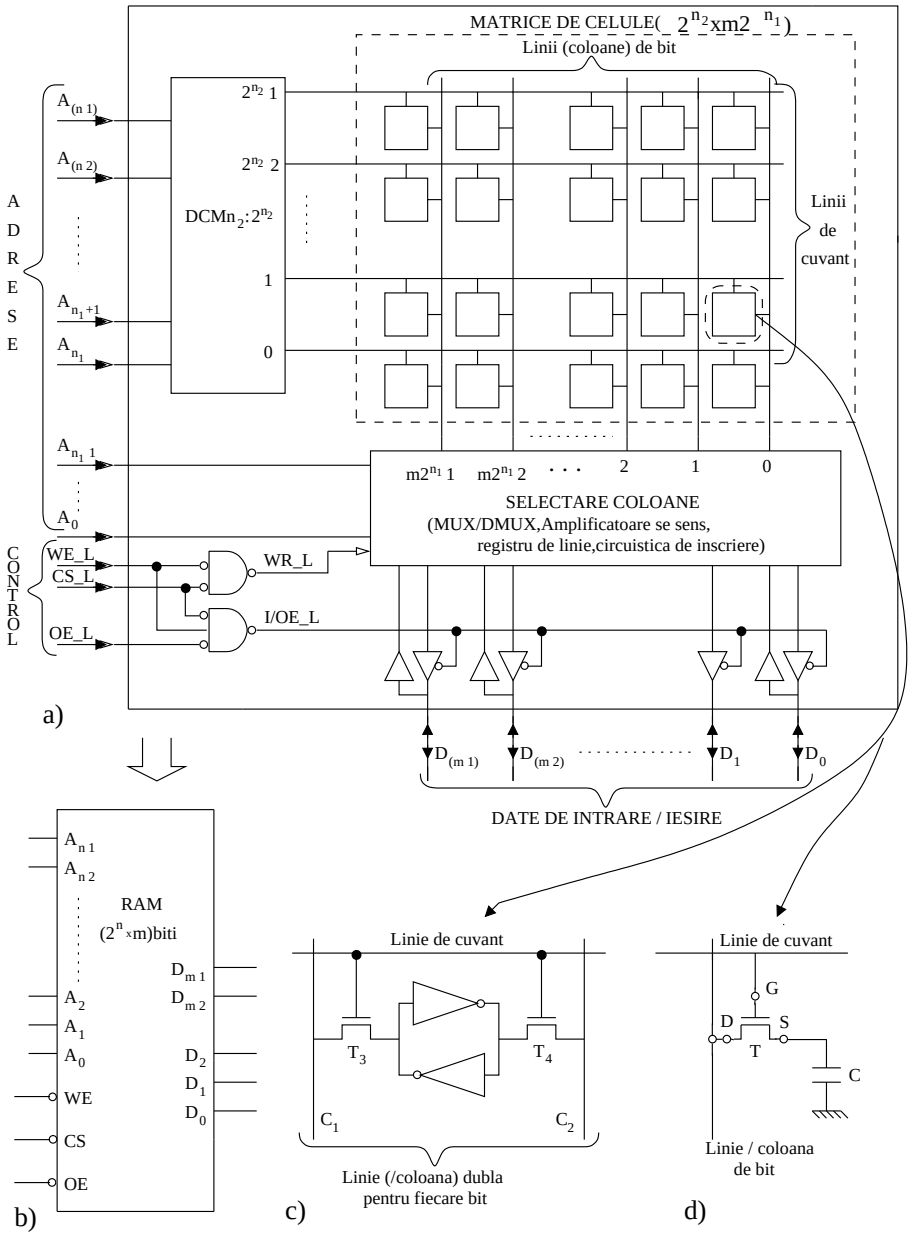


Figura 3.90 Memoria RAM: a) structurare de principiu cu o adresare bidimensională (prin linii de cuvânt şi linii (coloane) de bit); b) simbol de reprezentare pentru un circuit RAM; c),d) structurare de principiu respectiv pentru o celulă statică şi pentru o celulă dinamică.

ramură. O altă variantă care realizează valori mai reduse pentru timpul de transfer al datelor utilizează doar câte un singur tranzistor de trecere pe fiecare traseu care duce la cele $m \cdot 2^{n_1}$ linii de bit, iar comanda acestor tranzistoare se obţine în urma aplicări subcuvântului $A_{n_1-1}A_{n_1-2}\ldots A_1A_0$ la un $DCDn_1 : 2^{n_1}$. Fiecare din cele 2^{n_1} ieşiri ale DCD va comanda m tranzistoare de trecere, ceea ce poate constitui o încărcare ridicată. De asemenea, numărul de tranzistoare necesar pentru selectarea coloanelor de bit este $(m + n_1)2^{n_1}$; $m2^{n_1}$ tranzistoare de trecere şi $n_12^{n_1}$ tranzistoare pentru circuitul decodificator. Se pot concepe şi variante hibride între aceste variante cum ar fi: arbori parţiali cu un număr p de tranzistoare, $p << n_1$, apoi fiecare arbore parţial înseriat cu un tranzistor de trecere comandat de la ieşirea unui $DCD(n_1-p) : 2^{(n_1-p)}$ poate duce la valori rezonabile pentru timpii de acces şi numărul de tranzistoare [Kang ′ 96]. Dimeniunea memoriei RAM structurată cu o adresare bidimensională, când $n_1 \simeq n_2 \simeq n/2$, deoarece k are valori mici ≤ 4, $m = 1, 4, 8, 16$ se poate calcula cu relaţia 2.16, rezultă în $O(2^n)$.

Semnalele de control pentru memoria RAM sunt: selectare circuit CS_L (Chip Select), validarea ieşirii OE_L (Output Enable), care sunt prevăzute şi la memoria ROM, plus semnalul de validare a înscrierii WE_L (Write Enable). Structurarea anterioară a memoriei RAM, plecând de la memoria ROM, se reflectă şi în semnalele de control; o memorie RAM (statică) poate fi privită ca o memorie ROM înzestrată cu facilitatea de înscriere.

Semnalul de înscriere WR_L într-o celulă a matricei RAM se obţine, în interiorul circuitului, pa baza conjuncţiei a două semnale exterioare: cel de validare a înscrierii şi cel de selectare a circuitului$\overline{WR}_L = \overline{\overline{WE}_L \cdot \overline{CS}_L}$. Pentru o înscriere corectă a unui cuvânt de date, $D_{m-1}D_{m-2}\ldots D_1D_0$, într-o locaţie determinată de cuvântul de adresă, $A_{n-1}A_{n-2}\ldots A_1A_0$, trebuie respectată cu stricteţe o secvenţialitate în aplicarea semnalelor de control; întâi se aplică cuvântul de adresă, apoi cuvântul de date şi ambele trebuie să fie stabile în momentul apicării frontului activ al semnalului intern de înscriere WR_L. Deoarece semnalul WR_L se obţine din conjuncţia celor două semnale externe CS_L şi WE_L trebuie luate în considerare şi durata intervalelor de activare ale acestora în raport cu valorile satbile al cuvântului de date şi ale cuvântului de adresă, aceste restricţii vor fi explicate în secţiunea următoare.

Citirea unui cuvânt de date din matricea RAM, la fel ca şi la ROM, este mai puţin exigentă faţă de stabilitatea adresei în momentul activării semnalului intern de validare a ieşirii I/OE_L (I/O Enable); schimbarea adresei când $I/OE\_L = 0$ (activ) nu va produce o modificare greşită în matricea RAM ci doar, eventual, citirea altei locaţii. Semnalul intern I/OE_L pentru validarea bufferelor de ieşire TSL se obţine prin următoarea conjuncţie $\overline{I/OE}\_L = \overline{\overline{CS}\_L \cdot WE\_L \cdot \overline{OE}\_L}$. Pentru citire se aplică cuvântul de adresă, se activează semnalele de control selectare circuit, CS_L, de validare ieşire OE_L, iar cuvântul citit se obţine la ieşire prin bufferele TSL. Activarea semnalului de înscriere WE_L=0, va trece bufferele de ieşire în starea HZ (deci se exclude probabilitatea citirii unei locaţii); bufferele de intrare nu trebuie comandate deoarece un cuvânt de date nu poate fi înscris decât atunci când se comandă operaţia de înscriere prin semnalul de înscriere, WE_L=0.

Celula de memorie RAM poate fi cu o funcţionare statică sau dinamică. Celula RAM statică este de fapt o celulă bistabilă care prin cele două stări ale sale poate stoca bitul 0 sau 1. Ca celulă bistabilă poate fi un latch sau un bistabil. Se preferă totuşi celula pe bază de latch, Figura 3.90-c, pentru că necesită mai puţine compo-

nente decât un bistabil deci se pot realiza memorii de capacitate mai ridicată. Dar utilizarea unui latch în loc de bistabil, de exemplu de tip D, prezintă dezavantajele de transparenţă şi de funcţionare asincronă; **memoria RAM având în consecinţă funcţionare asincronă**. (Asincronismul trebuie privit prin faptul că aplicarea semnalelor de control nu sunt raportate la apariţia semnalului semnalului de ceas). Accesul la un latch dintr-un nod se realizează prin două tranzistoare de trecere T_3, T_4 comandate de potenţialul liniei de cuvânt care trece prin acel nod. Cele două tanzistoare conectează latch-ul, pentru realizarea operaţiilor de citire şi înscriere, la două linii de bit notate cu C_1 şi C_2, deci pentru fiecare nod există nu una ci două linii de bit. Există variante cu o singură linie de bit dar implementările uzuale sunt cu două linii de bit deoarece operaţiile de înscriere şi citire sunt mult mai sigure. Latch-ul dintr-un nod comandat prin tranzistoarele de trecere T_3 şi T_4 poate fi comparat cu latch-ul cu ceas, Figura 3.40-a; semnalul de pe linia de cuvânt la o celulă de memorie ar fi echivalentul semnalului de ceas al latch-ului cu ceas care validează porţile 3 şi 4 pentru aplicarea intrărilor de date R şi S (la celula de memorie se acceptă două inversoare în loc de porţi pentru creşterea numărului de biţi integraţi pe suprafaţa de Si).

Celula RAM dinamică, Figura 3.90-d, are ca suport pentru memorarea bitului 1 sau 0 prezenţa sau absenţa unei sarcini pe un condensator C realizat în fiecare nod al matricei. Condensatorul este conectat la linia de bit printru-un tranzistor de trecere T care este comandat pe poartă de către potenţialul aplicat pe linia de cuvânt. Deoarece sarcina cu care este încărcat condensatorul C, corespunzătoare memorării bitului 1, se micşorează în timp prin curentul rezidual al tranzistorului T se impune refacerea (reîmprospătarea) acestei sarcini prin reîncărcarea condensatorului; în general timpul după care se impune reîncărcarea este de ordinul ms. Denumirea de celulă dinamică s-a acceptat prin similitudinea funcţionării cu cea a circuitelor dinamice, dar de data acesta nu se utilizează o capacitate parazită a circuitului ci această capacitate C este implementată în fiecare nod împreună cu tranzistorul T.

Parametrii care se analizează pentru compararea circuitelor de memorie sunt: 1 - costul/bit, care este direct legat de densitatea de integrare (numărul de biţi pe unitatea de suprafaţă de Si); 2 - timpul de acces, care determină viteza de lucru a circuitului; 3 - puterea disipată. Comparativ, memoriile RAM statice realizează viteze mult mai bune de ordinul ns (pot ajunge până la 2ns timp de acces) faţă de cele dinamice care se situează în domeniul zeci de ns, in schimb memoriile RAM dinamice au performanţe superioare în ceea ce priveşte puterea disipată şi densitatea de integrare.

3.6.1 Memoria RAM statică

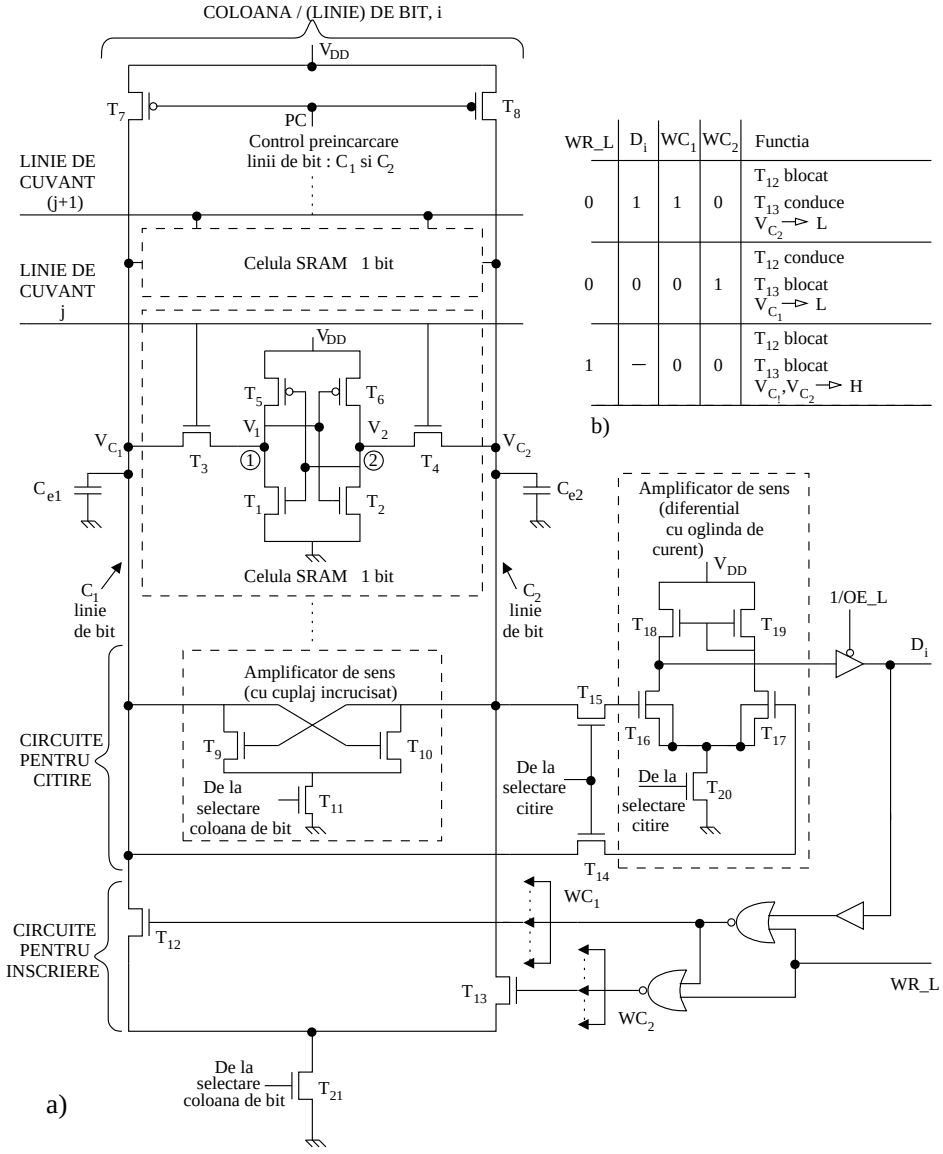
Memoria cu acces aleatoriu statică, **SRAM**, utilizează ca celulă de memorie un latch; inversoarele latch-ului pot fi cu sarcină rezistivă (realizată în polisiliciu) sau cu sarcină activă (tranzistoare cu canal iniţial, Figura 1.54-a, sau cu tranzistor complementar, Figura 1.33-a). În funcţie de existenţa sau neexistenţa unui tranzistor de sarcină, în structura inversorului, celula statică de memorare dintr-un nod al matricei poate fi formată respectiv din 6 sau 4 tranzistoare (sunt incluse şi cele două tranzistoare de trecere prin care se conectează la linia de bit); de unde denumirea de celulă 6T-SRAM sau 4T-SRAM. Practic, actual, majoritatea implementărilor VLSI

de memorii SRAM se bazează pe o celulă 6T-CMOS; aceată impunere a memoriilor CMOS-SRAM în raport cu alte implementări SRAM se datorează următoarelor avantaje: 1 - putere redusă (puterea disipată în regim staţionar este practic nulă, determinată doar de curentul rezidual prin inversoarele CMOS); 2 - imunitate ridicată la zgomot, relaţia 1.19, deoarece admite margine de zgomot de valoare ridicată, relaţia 1.18; 3 - posibilitatea de funcţionare într-o largă plaje de tensiuni de alimentare (mai ales la tensiuni reduse). Dezavantajele memoriei CMOS-SRAM sunt: 1 - consum mai ridicat de suprafaţă pe aria de Si; 2 - proces tehnologic mai complex; 3 - tendinţa de apariţie a fenomenului de "zăvorâre"(vezi secţiunea 1.6). Prin tehnologiile actuale cu multiple straturi de polisiliciu şi multiple straturi de metal (pentru realizarea conexiunilor) se reduce dezavantajul consumului mai ridicat de suprafaţă. Implementările de putere şi tensiune redusă utilizează o celulă 6T-CMOS, în consecinţă în continuare se va analiza structurarea şi funcţionarea memoriei cu o astfel de celulă. O astfel de structurare este prezentată în Figura 3.91-a cu o celulă 6T desenată la intersecţia liniei de cuvânt j (activată de ieşirea j a decodificatorului de linii, $DCDn_2 : 2^{n_2}$, Figura 3.90-a) cu coloana de bit i (compusă din cele două coloane de bit C_1, C_2) care este selectată prin cuvântul $A_{n_1-1}A_{n_1-2}\dots A_1A_0$ aplicat grupului de MUX/DMUX.

Operaţia de citire a celulei. Se consideră că celula este înscrisă în starea logică 0 care, prin convenţie, corespunde valorilor de tensiune: în nodul (1), $V_1 = 0V$, deci T_1 conduce, T_5 blocat; iar în nodul (2), $V_2 = V_{DD}$, T_2 blocat, T_6 conduce. La activarea liniei de cuvânt j, cu o tensiune V_{DD}, tranzistoarele de acces T_3 şi T_4 intră in conducţie conectând liniile de bit C_1 şi C_2, care au potenţialele $V_{C_1} = V_{C_2} = V_{DD}$, la ieşirile latch-ului. Datorită diferenţei de potenţial între linia C_1 şi nodul (1), prin tranzistorul T_3 şi T_1, condensatorul C_{e1}, (capacitatea echivalentă a liniei C_1) se descarcă producând o mică scădere (maxim de ordinul sutelor de mV, deoarece capacitatea C_{e1} este destul de mare) a potenţialului V_{C_1}, rezultând o diferenţă de potenţial între cele două linii $\Delta V = V_{C_1} - V_{C_2} < 0$. În timp ce V_{C_1} scade puţin, tensiunea în nodul (1) creşte, tensiunea V_1 nu trebuie să devină mai mare decât tensiunea de prag V_{pT_2} (când T_2 ar intra în conducţie), $V_{1max} \leq V_{pT_2}$. Din această relaţie, în proiectarea latch-ului, rezultă valorile pentru raporturile dimensiunilor $(W/L)T_3$ şi $(W/L)T_1$, pentru care se poate realiza citirea fără comutarea latch-ului în starea opusă.

Amplificarea în prima etapă a diferenţei ΔV se realizează cu un amplificator de sens, care, de fapt, este un latch ale cărei ieşiri sunt conectate la liniile de bit C_1 şi C_2 deci potenţialele pe porţile celor două tranzistoare T_9 şi T_{10} sunt egale cu $V_{C_1} = V_{C_2} = V_{DD}$. În momentul când potenţialul V_{C_1} al liniei C_1 începe să descrească, şi tranzistorul T_{11} este comandat în conducţie (prin selectarea coloanei), tranzistorul T_{10} este comandat înspre blocare, care prin reacţie comandă pe T_9 înspre conducţie realizându-se astfel bascularea acestui latch-amplificator. Pe traseul T_9, T_{11}, înspre masă condensatorul C_{e1} se descarcă rezultând o mărire a diferenţei $V_{C_1} - V_{C_2}$. Această diferenţă pronunţată de tensiune, dintre cele două linii C_1 şi C_2, rezultată în urma procesului de citire a celulei de memorie, se aplică prin intermediul tranzistoarelor de trecere T_{14} şi T_{15} pe intrările amplificatorului de sens.

Amplificatorul de sens (diferenţial) va genera (prin driverul de ieşire TSL) un semnal 0 logic pentru $\Delta V < 0$ şi un semnal 1 logic pentru $\Delta V > 0$. Tranzistorul T_{20} este realizat cu o lungime de canal mărită pentru a avea o funcţionare de generator



WR_L	D_i	WC_1	WC_2	Functia
0	1	1	0	T_{12} blocat T_{13} conduce $V_{C_2} \rightarrow$ L
0	0	0	1	T_{12} conduce T_{13} blocat $V_{C_1} \rightarrow$ L
1	—	0	0	T_{12} blocat T_{13} blocat $V_{C_1}, V_{C_2} \rightarrow$ H

Figura 3.91 Memoria CMOS-RAM statică: a) structurarea unei coloane dintr-o matrice RAM. Pe lângă structura unei celule sunt prezentate circuitele necesare operaţiilor de înscriere şi citire; b) tabelul de adevăr pentru generarea semnalelor, de comandă a liniilor de bit WC_1 şi WC_2, în efectuarea operaţiei de înscriere.

de curent. Terminalele de substrat ale tranzistoarelor T_{16} şi T_{17} sunt conectate la nodul de sursă comună, şi nu la masă, pentru a se evita variaţia tensiunilor de prag ale acestor tranzistoare datorită polarizării de substrat. Pentru aceste variaţii mici ΔV amplificatorul de sens diferenţial realizează un semnal de răspuns cu un timp de creştere de valoarea $\tau_r \simeq 1ns$ (vezi Figura 1.15-a).

La citirea stării logice 1, înscrisă în celulă (în nodul (1), $V_1 = V_{DD}$, deci T_5 conduce, T_1 blocat; iar în nodul (2), $V_2 = 0V$, deci T_2 conduce, T_6 blocat) analizând, similar, ca în cazul anterior se generează $\Delta V = V_{C_1} - V_{C_2} > 0$, iar la ieşirea driverului TSL se obţine semnalul 1 logic.

În utilizarea memoriei SRAM poate apare cazul când se citeşte o celulă de pe o coloană, înscrisă în 0 deci $\Delta V < 0$, iar citirea următoare este de la o celulă tot de pe aceeaşi coloană, dar înscrisă în 1 deci $\Delta V > 0$. Pentru aceste cazuri, pentru a citi corect şi rapid, trebuie încărcate, în intervalul dintre cele două citiri, cele două capacităţi echivalente C_{e1} şi C_{e2} cu aceeaşi sarcină deci liniile C_1 şi C_2 aduse la acelaşi potenţial. În acest scop este introdusă posibilitatea de egalizare (/preîncărcare), PC, de aducere la acelaşi potenţial a liniilor de bit.

Operaţia de înscriere în celulă. Să considerăm, ca şi anterior, că valorile la ieşirea latch-ului $V_1 = 0$, $V_2 = V_{DD}$ reprezintă 0 logic, iar $V_1 = V_{DD}$ şi $V_2 = 0$ reprezintă 1 logic. Pentru înscrierea celulei de memorie în starea 1 se aplică pe linia de bit C_1 potenţialul $V_{C_1} = V_{DD}$ şi pe linia C_2 potenţialul $V_{C_2} = 0V$, iar celula iniţial este în starea zero. La activarea liniei de cuvânt j tranzistoarele T_3 şi T_4 intră în conducţie, în nodul (1) şi pe poarta tranzistoarelor T_2, T_6 se aplică $V_{C_1} = V_{DD}$, iar în nodul (2) şi pe poarta tranzistoarelor T_1, T_5 se aplică $V_{C_2} = 0V$; prin conexiunile care realizează o reacţie pozitivă starea inversorului din dreapta, cu T_2 blocat T_6 în conducţie, va trece în starea cu T_2 în conducţie şi T_6 blocat, iar inversorul din stânga va trece din starea cu T_1 în conducţie şi T_5 blocat în starea cu T_1 blocat şi T_6 în conducţie ceea ce reprezintă înscrierea stării 1 în celulă. Dacă celula era în starea 1, aplicarea pe liniile de bit $V_{C_1} = V_{DD}$, $V_{C_2} = 0$ nu produce bascularea latch-ului. În mod similar, dacă pe liniile de bit C_1, C_2 se aplică respectiv $V_{C_1} = 0V$ şi $V_{C_2} = V_{DD}$, iar celula era în stare logică 1 se va produce bascularea celulei în starea logică 0, iar dacă era în starea logică 0 nu se produce nici o modificare.

Aplicarea celor două tensiuni de niveluri logice diferite pe cele două linii de bit se realizează cu circuitele desenate în partea de jos a figurii. Practic, pentru înscrierea în 1, coloana C_2 este conectată la masă prin tranzistoarele T_{13} şi T_{21}, iar pentru înscrierea în 0 coloana C_1 este conectată la masă prin tranzistoarele T_{12} şi T_{21}; T_{21} intră în conducţie când coloana de bit j este selactată prin semicuvântul de adresă $A_{n_1-1}A_{n_1-2}\ldots A_1A_0$. Semnalele complementare WC_1 şi WC_2, care comandă respectiv tranzistoarele T_{12} şi T_{13}, se obţin din valoarea bitului D_i, a cuvântului de date, aplicat pe intrare pentru a fi înscris şi din semnalul de înscriere în celulă WR_L, prin intermediul a două porţi NOR, conform tabelului de adevăr din Figura 3.91-b. Tranzistoarele T_{12}, T_{13} şi T_{21} trebuie să fie dimensionate cu valori ridicate pentru raportul (W/L) pentru ca să forţeze aproape la zero potenţialele coloanelor în operaţia de înscriere. Semnalele WC_1 şi WC_2 pot fi utlizate pentru comanda şi a altor coloane, în cazul nostru de 2^{n-1} coloane (dar atenţie la fan-out care creşte mult!).

Parametrii de timp pentru operaţiile de citire şi înscriere în memorie. Pentru efectuarea corectă a operaţiilor de citire şi înscriere semnlele care concură

(cuvântul de adresă $A_{n-1} \div A_0$, cuvântul de date $D_{m-1} \div D_0$ pe I/O şi semnalele de control: validare înscriere WE_L; selectare circuit CS_L; validare ieşire OE_L, Figura 3.90-b) trebuie să aibă o anumită succesiune în timp şi sunt restricţionate de anumite valori limită de timp. Aceste restricţii sunt impuse de înscrierea latch-ului care are o funcţionare transparentă. Semnalul intern WR_L care înscrie datele în latch, şi care determină durata de transparenţa a latch-ului, se obţine ca o conjuncţie între semnalele de intrare, $\overline{WR\_L} = \overline{\overline{WE\_L} \cdot \overline{CS\_L}}$, deci acest semnal este activ în L pe o durată τ_{WP} egală cu intervalul în care ambele semnale de intrare sunt active. Latch-ul devine transparent la începutul duratei τ_{WP}, pe frontul H-L, şi rămâne în stare de transparenţă până la frontul L-H. Pe durata de transparenţă, τ_{WP}, nu se impune pentru cuvintele de date şi de adresă aplicate să fie stabile, dar este necesar ca acestea să fie stabile faţă de frontul L-H, pe un interval τ_{SU} înainte şi τ_{H} după; această restricţie se va reflecta în definirea parametrilor de timp următori [Wakerly ′00].

Operaţia de citire în memorie, Figura 3.92-a (semnalul de înscriere se consideră neactivat, WE_L=H)

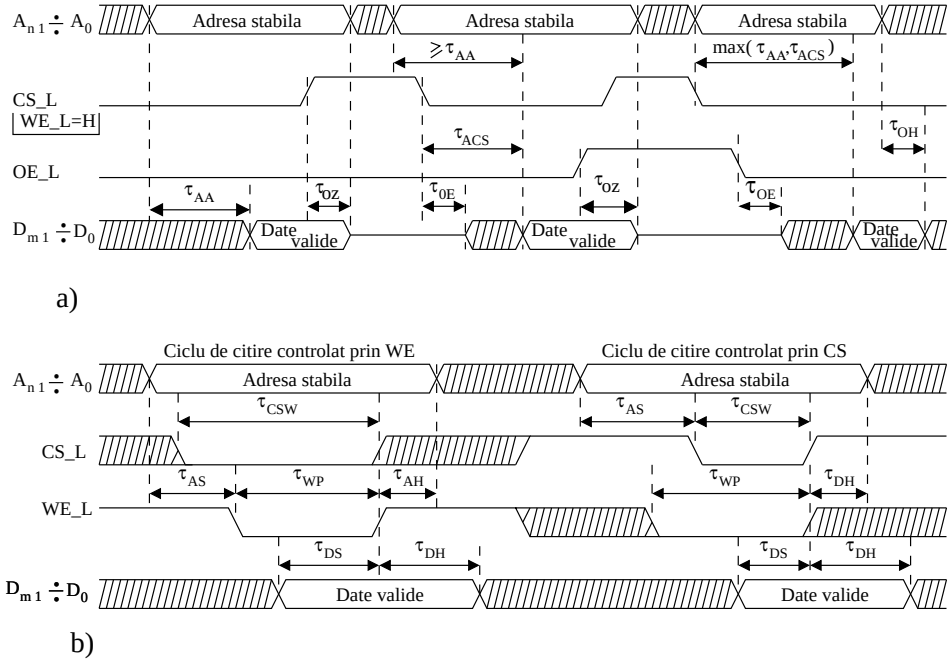


Figura 3.92 Definirea parametrilor de timp pentru o memorie SRAM: a) pentru operaţia de citire (WE_L, inactiv); b) pentru operaţia de înscriere

- τ_{AA}, timpul de acces (faţă de aplicarea adresei). Este intervalul de timp din momentul aplicării la intrare a cuvântului de adresă până în momentul când cuvântul de date citit este valid la pinii I/O (se presupune că semnalul intern I/OE_L a fost deja activat). În exprimarea profesională când se referă la o memorie, de exemplu de 60ns se înţelege τ_{AA}.

- τ_{ACS}, timpul de acces de la selectarea circuitului. Este intervalul de timp din momentul aplicării semnalului CS până în momentul când cuvântul de date citit este valid la pinii I/O (se presupune deja cuvântul de adresă şi semnalul OE că au fost aplicate şi sunt stabile). În general $\tau_{AA} = \tau_{ACS}$ dar nu şi în cazul circuitelor la care neactivarea semnalului CS introduce circuitul în regimul de aşteptare (**standby**), Figura 2.49-a.

- τ_{OE}, timpul de validare a ieşirii. Este intervalul din momentul în care ambele semnale OE şi CS sunt activate (care generează semnalul intern I/OE) până când driverele de ieşire TSL trec în starea de funcţionare normală; $\tau_{OE} < \tau_{ACS}$.

- τ_{OZ}, timpul de devalidare a ieşirii. Acest parametru caracterizează trecerea în regim de HZ a driverelor TSL; este intervalul de timp din momentul dezactivării semnalelor OE şi CS până când driverele TSL ajung în HZ.

- τ_{OH}, timpul de menţinere a datelor. Specifică intervalul de timp în care cuvîntul citit mai poate rămâne valid la I/O după ce cuvântul de adresă, al locaţiei de unde s-a citit, a fost anulat.

Operaţia de înscriere în memorie, Figura 3.92-b

- τ_{AS}, timpul de stabilizare a adresei. Specifică cu cât timp înainte de aplicarea semnalului de înscriere (generat prin conjuncţia CS·WE) cuvântul de adresă trebuie să fie stabil.

- τ_{AH}, timpul de menţinere a adresei. Specifică cât timp după aplicarea semnalului de înscriere cuvântul de adresă mai trebuie să nu se modifice. τ_{AS} şi τ_{AH} determină, pentru înscriere, intervalul interzis Δ, centrat pe frontul de înscriere în care biţii cuvântului de adresă nu trebuie să se modifice.

- τ_{DS}, timpul de stabilizare a datelor. Specifică cu cât timp înainte de frontul activ al semnalului de înscriere, WR_L, datele aplicate pe teminalele I/O trebuie să fie stabile.

- τ_{DH}, timpul de menţinere a datelor (nemodificate după frontul de înscriere). τ_{DS} şi τ_{DH}, la fel ca şi τ_{AS} şi τ_{AH} pentru adrese, determină pentru date intervalul interzis Δ, centrat pe frontul de înscriere în care biţii cuvântului de date nu trebuie să se modifice.

- τ_{CSW}, timpul de stabilizare pentru CS. Specifică cu cât timp înainte de frontul de înscriere semnalul CS trebuie să fie stabil.

- τ_{WP}, lăţimea palierului semnalului de înscriere. Specifică cu cât timp înainte de frontul de înscriere trebuie să se aplice semnalul WE. De fapt transparenţa latch-ului, după cum s-a spus, este determinată de durata palierului semnalului WR (generat prin conjuncţia $CS\_L \cdot WE\_L$). Considerând că aceste două semnale sunt dezactivate simultan atunci se pot distinge două cicluri de înscriere: controlat prin WE sau controlat prin CS, după cum WE_L este activat ultimul sau CS_L este activat ultimul.

Din analiza funcţionării memorie SRAM şi din prezentarea parametrilor se observă că operaţiile de înscriere şi de citire sunt asincrone (apariţia lor nu este determinată de un semnal de ceas). De această observaţie trebuie ţinut cont când o memorie SRAM este introdusă într-un sistem în care există alte componente (registre, numărătoare, ASM) care au o funcţionare sincronă.

3.6.2 Memoria RAM dinamică

Principala caracteristică ce a impus pentru capacităţi de stocare mari, chiar Gb ($1G = 10^9$), **memoria dinamică, DRAM**, este densitatea ridicată de integrare ce se poate obţine cu acest tip de memorie deşi prezintă circuistică suplimentară faţă de SRAM. Există variante de DRAM care se bazează pe celule de memorie compuse dintr-o capacitate de stocare şi 1,3 sau 4 tranzistoare, dar, evident, s-a generalizat structura de DRAM care prezintă în fiecare nod al matricei celulă doar un singur tranzistor, notată prin 1-T DRAM; numai structura bazată pe această celulă se va prezenta. Întâi se va prezenta modalitatea de realizare a proceselor de înscriere şi de citire/regenerare la nivelul unei celule dintr-o matrice de memorie iar, apoi, modul de comandă, al operaţiilor de înscriere şi citire/regenerare la nivel de circuit DRAM, prin aplicarea la pinii circuitului a semnalelor: de control, adresă şi date.

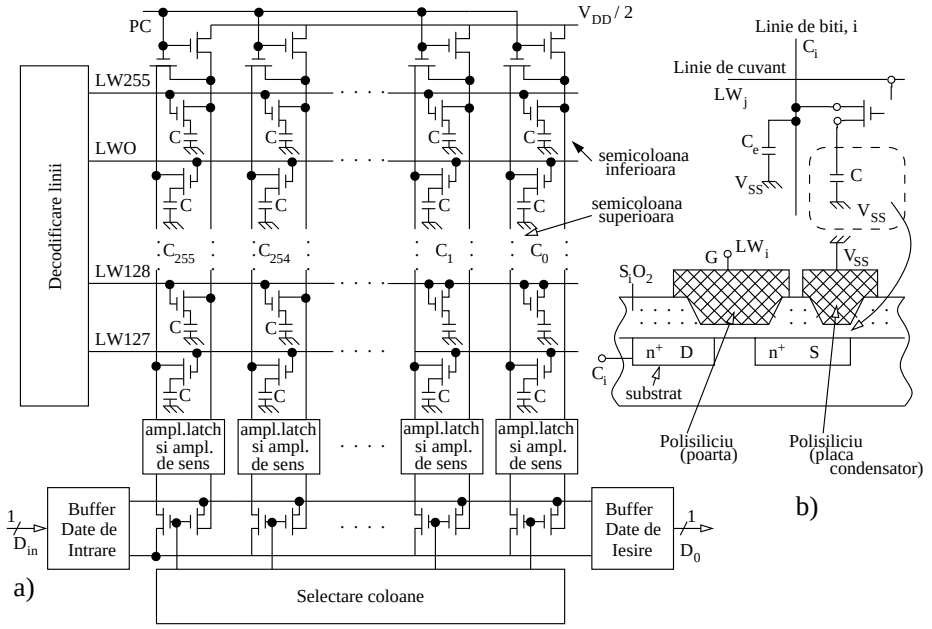


Figura 3.93 Memoria DRAM:a) structurarea de principiu a unei matrice de dimensiune 256 linii × 256 colone; b) structura unei celule de tipul 1-T şi layout-ul corespunzător

Se consideră o matrice de 256 linii de cuvânt ($LW_{255}, \ldots, LW_1, LW_0$) şi 256 coloane/ linii de bit ($C_{255}, \ldots C_1, C_0$), Figura 3.93-a. În fiecare nod al matricei

(256×256), celula constă dintr-un tranzistor nMOS şi un condensator C, Figura 3.93-b. Condensatorul de stocare C se realizează între zona difuzată n^+ din substrat, ca sursă a tranzistorului, şi o placă din polisiliciu realizată deasupra stratului de SiO_2, conectată la masă (V_{SS}). Într-un nod ji al matricei, conectarea condensatorului C la coloana de bit C_i prin intermediul tranzistorului nMOS, se realizează prin activarea (V_{DD}) a liniei de cuvânt LW_j. Prin această conectare sunt puse în paralel două capacităţi de valori sensibil diferite: capacitatea C de stocare din nod (valori 30-10fF, 1fF=10^{-15}F) şi capacitatea echivalentă $C_e(\geq 300\text{fF})$ a coloanei. Prin transferul de sarcină între cele două condensatoare, când sunt puse în paralel, deci prin valoarea tensiunii rezultate pe coloana de bit, se pot realiza operaţiile de înscriere, citire/regenerare.

Înscrierea celulei din nodul ji cu bitul D_{in} aplicat pe intrare se realizează simplu. Întâi, se activează prin semicuvântul superior de adresă aplicat la intrarea decodificatorului de linii, linia de cuvânt LW_j, apoi, prin semicuvântul inferior de adresă, aplicat decodificatorului de coloană, se selectează coloana de bit C_i, iar pe acestă coloană se aplică valoarea bitului de intrare, D_{in}, deci prin tranzistorul în conducţie T din nodul ji se încarcă condensatorul cu sarcină zero (pentru D_{in}=0) sau la potenţialul V_{DD} (pentru D_{in}=1).

Există o paricularitate în structurarea acestei matrice, fiecare coloană de bit este secţionată în două lungimi egale: semicoloana superioară, care cuprinde nodurile de intersecţii cu liniile de cuvânt LW_{255}, $\ldots LW_{129}$, LW_{128} şi semicoloana inferioară care cuprinde nodurile de intersecţie cu liniile de cuvânt LW_{127}, $\ldots LW_1$, LW_0. (Această secţionare explică de ce în Figura 3.93-a numărul liniilor de cuvânt LW_j sunt alternate în numărare corespunzător celor două intervale: de la $LW_0 \div LW_{127}$ cu $LW_{128} \div LW_{255}$.) Această secţionare a coloanelor apare şi pe aria de Si prin realizare a două subarii iar în spaţiul dintre acestea este implementat, corespunzător fiecărei coloane de bit, câte un amplificator latch şi un amplificator de sens diferenţial, Figura 3.94-a. (În această figură este desenat doar amplificatorul latch, circuitul complet amplificator latch şi amplificator de sens diferenţial sunt similare cu cele prezentate în Figura 3.91-a.) Fiecare din cele două semicoloane (S – superioară, I – inferioară) este caracterizată de o capacitate electrică echivalentă, C_{eI} ,C_{eS} ($C_{eI} = C_{eS}$). În figură sunt prezentate pentru fiecare semicoloană tranzistoarele şi condensatoarele din nodurile corespunzătoare, care sunt comandate de semnale de la liniile de cuvânt LW_j; în plus, sunt, pentru fiecare semicoloană, introduse câte o celulă martor. Cele două celule martor de pe o coloană sunt, ca structură, identice cu cea a unei celule din oricare nod dar au un condensator de stocare de valoare $C/2$, prezintă pentru comandă pe porţile tranzistoarelor semnalele CM_I şi CM_S, care nu sunt obţinute de la linii de cuvânt, iar potenţialele în punctele X şi Y pot fi puse la masă prin intermediul a două tranzistoare comandate prin semnalul de preîncărcare, PC.

Citirea/regenerarea celulei. Această operaţie este realizată, într-o succesiune temporală compusă din trei faze, Figura 3.94-b.

Faza 1 - preîncărcare. Prin activarea semnalului de preîncărcare, PC, capacităţile echivalente C_{eI} şi C_{eS} ale semicoloanei superioare şi inferioare de bit se încarcă la tensiuni (de nivel H) egale, iar potenţialele în punctele X şi Y sunt fixate la masă, Figura 3.94-c. Este necesară această egalizare a potenţialelor pentru cele două semicoloane deoarece după o citire, pentru refacerea datei citite, semicoloanele vor fi forţate, în urma comutaţiei amplificatorului latch, una în nivel H iar cealaltă în nivel L (vezi

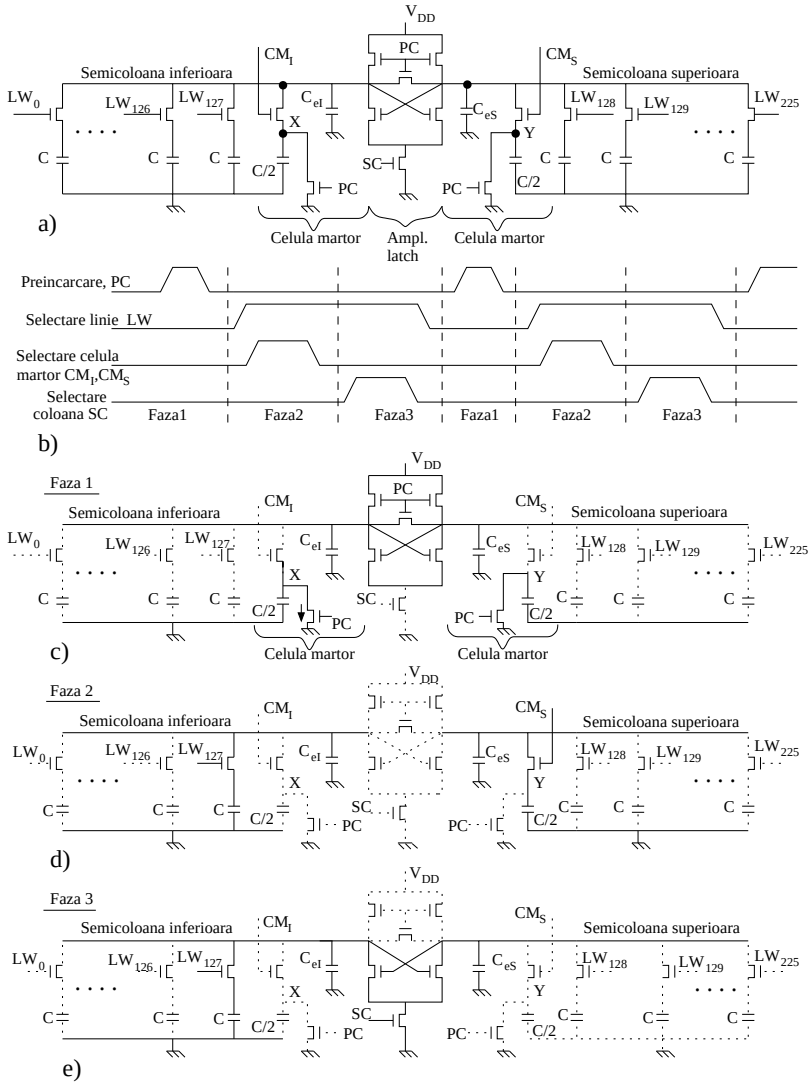


Figura 3.94 Realizarea ciclului citire/regenerare pe o memorie 1-T DRAM prin succesiunea a trei faze: a) structurarea unei coloane de bit a matricei cu 256 de noduri în două semicoloane egale separate printr-un amplificator latch şi două celule martor; b) succesiunea semnalelor de comandă pentru cele trei faze; c) Faza 1, preîncărcarea coloanei de bit şi descărcarea celulelor martor; d) Faza 2, activarea liniei de cuvânt, LW_{127} $(= V_{DD})$, şi a celulei martor din semicoloana opusă; e) Faza 3, selectarea coloanei de bit, SC_i $(= V_{DD})$, citirea celulei şi regenerarea celulei din nodul (127, i). (Partea de circuit care nu participă în faza respectivă este desenată cu linie punctată)

după Faza 3). (Elementele din figură care nu sunt implicate în realizarea acestei faze sunt desenate cu linie punctată).

Faza 2 - activarea liniei de cuvânt şi comanda celulei martor, Figura 3.94-d. Se presupune că prin aplicarea semicuvântului superior de adresă la decodificatorul de linii se activează linia de cuvânt LW_{127} $(= V_{DD})$ din semicoloana inferioară şi semnalul CM_S $(= V_{DD})$ din semicoloana superioară. Dacă în celula de la intersecţia coloanei i cu linia de cuvânt LW_{127} era înscris 1 logic, atunci potenţialul semicoloanei inferioare va creşte puţin, iar potenţialul semicoloanei superioare va scădea puţin, pentru că condensatorul $C/2$ al celulei martor se va încărca; deci apare o diferenţă de potenţial ΔV (maximum câteva sute de mV) între cele două semicoloane, care se aplică între ieşirile amplificatorului latch. Iar dacă în celulă era înscris 0 logic potenţialul semicoloanei inferioare va scădea mai mult decât al semicoloanei superioare, deoarece capacitatea celulei 127 este dublă faţă de capacitatea celulei martor. Evident, prin conectarea celulei la coloana de bit valoarea potenţialului pe condensatorul C din nodul accesat se modifică, deci o distrugere a nivelului de bit care era înscris.

Faza 3 - selectarea coloanei de bit, Figura 3.94-e. Prin aplicarea semicuvântului inferior de adresă la decodificatorul de coloane, semnalul pentru selectarea coloanei i devine activ, SC $(= V_{DD})$, amplificatorul latch basculează forţând pentru semicoloana inferioară potenţialul V_{DD}, deci condensatorul nodului (127, i) se reîncarcă cu 1 logic, iar potenţialul semicoloanei superioare de bit este forţat la masă. În consecinţă data stocată în nodul (127,i), deteriorată ca nivel în urma citirii, este regenerată, rezultă că operaţia realizată de citire este urmată automat de regenerare. Totodată, este comandat amplificatorul de sens diferenţial care va genera pe pinii de ieşire bitul citit în celula nodului (127, i). În continuare poate urma un nou ciclu de citire/regenerare, cu trei faze succesive, pentru alt nod.

În Faza 2, a ciclului de citire/regenerare, sarcina pe condensatorul de stocare dintr-o celulă este modificată dar este refăcută în Faza 3. Dar sarcina stocată pentru valoare logică 1 pe condensatorul C, dintr-o celulă care nu este supusă la un ciclu citire/regenerare, se micşorează (datorită curentului rezidual prin tranzistorul nMOS) încât după un anumit interval de timp nu mai reprezintă nivelul de 1 logic. Deci, dacă nu se efectuează un ciclu de citire/regenerare se impune ca fiecare celulă a matricei sa fie supusă unei operaţii de **regenerare pentru refacere (refreshment)** după un anumit interval de timp (de ordinul câtorva ms). Ceea ce este, totuşi, avantajos, în acest proces de refershment, care complică structura şi lucrul cu o memorie DRAM, constă în faptul că regenerarea nu se face pentru fiecare celulă în parte ci simultan pentru toate celulele din nodurile de pe o linie de cuvânt LW. Pentru matricea anterioară de 256 linii $\times$ 256 coloane, nivelul pentru bitul 1 nu este deteriorat, dacă celula este regenerată la un interval de 4ms, ceea ce implică pentru fiecare linie de cuvânt să fie activată, $LW = V_{DD}$, la un interval $4ms : 256 = 15625ns$. Dacă timpul de ciclu, pentru operaţia de citire/regenerare a unei linii, este de 100ns, rezultă că timpul consumat de o regenerare în bloc a tuturor celulelor matricei, prin activarea succesivă a celor 256 linii de cuvânt, este de $256 \times 100 = 2.56 \times 10^4 ns$ adică 0.64% din 4ms; deci 99.36% din timp memoria poate fi folosită efectiv pentru operaţiile utile de scriere sau citire. La unele circuite DRAM trebuie aplicate din exterior comenzile pentru regenerare, la altele, având în interior circuistica necesară, îşi autogenerează procesul de regenerare; circuistica necesară constă dintr-un numărător care, după consumarea unui anumit interval de timp prestabilit, în cazul anterior 4ms, generează

prin numărarea în sens direct/invers toate adresele liniilor de cuvânt.

Modalitatea de realizare a operaţiilor de înscriere, citire/regenerare, sau regenerare separată, la nivel de circuit DRAM, prin aplicarea semnalelor din exterior depinde de structurarea circuitului respectiv. O structurare de principiu pentru un circuit DRAM de 64Kbit este prezentată în Figura 3.95-a.

O particularitate în funcţionarea unei memorii DRAM, care înfluenţează structurarea sa, rezultă din timpul necesar pentru accesarea unei linii de cuvânt care este mult mai lung decât timpul necesar pentru accesarea unei coloane de bit. În consecinţă, se poate accesa o celulă a matricei aplicând întâi semicuvântul de adresare superior la decodificatorul pentru liniile de cuvânt şi numai după aceea se aplică semicuvântul inferior pentru generarea selecţiei de coloană, deci o adresare în doi paşi: întâi linia (care necesită un timp de acces mai lung)şi apoi coloana. Neaplicarea simultană a celor două semicuvinte a determinat ca lăţimea magistralei de adrese pentru o memorie de capacitate 2^n biţi să nu fie de n biţi ci de $n/2$ biţi. Aceasta explică de ce capacităţiile memoriilor DRAM sunt multiplu de patru; la creşterea (semi)cuvântului de adresă cu un bit atât numărul liniilor de cuvânt cât şi numărul coloanelor de bit se dublează. Pentru memoria de 256×256 biţi se aplică întâi 8 biţi la pinii de adresă, care reprezintă semicuvântul superior de adresă, acest cuvânt este înscris într-un registru latch pentru adresa de linie, iar prin intermediul decodificatorului de linii se activează una din cele 256 linii. Înscrierea semicuvântului în registru pentru adresa de linie se face pe frontul negativ al **semnalului de strob linii, RAS_L (Row Address Strob)**. Apoi, se aplică la pinii de adresă alţi 8 biţi care reprezintă semicuvântul inferior de adresă iar acest cuvânt este înscris într-un registru pentru adresa de coloană; prin intermediul decodificatorului se activează o cale din grupul de MUX/DMUX pentru selectarea unei coloane în matricea de celule. Înscrierea în registru pentru adresa de coloană se face pe frontul negativ al **semnalului de strob coloane, CAS_L (Collumn Address Strob)**. Pe durata CAS_L=0 bufferul de ieşire TSL este trecut din HZ în starea de funcţionare normală. Mai există încă două semnale de control: unul necesar numai pentru validarea operaţiei de înscriere WE_L iar celălalt (nefigurat în această structură) pentru selectarea circuitului CS_L.

Citirea memoriei, Figura 3.95-a. Semicuvântul superior de adresă, aplicat cu valori stabile pe pinii circuitului, este strobat/înscris în registrul pentru adresa de linie pe frontul negativ al semnalului RAS_L; se selectează linia corespunzătoare din matrice iar conţinutul acestei linii este înscris într-un registru latch de linie. Apoi, după ce semicuvântul inferior de adresă aplicat pe pinii circuitului devine stabil acesta este înscris în registrul pentru adresa de coloană pe frontul negativ al semnalului de strob CAS_L. Pe baza acestei adrese de coloană prin intermediul multiplexoarelor este selectat bitul corespunzător, din cuvântul înscris temporar în registrul latch de linie, şi aplicat pe pinul de ieşire D_O (bufferul de ieşire TSL este în stare normală de funcţionare atât timp cât CAS_L este activat). La dezactivarea semnalului RAS_L, conţinutul din registrul latch de linie este reînscris (/regenerare) înapoi în linia de cuvânt care a fost selectată, deci la sfârşitul operaţiei de citire a unei locaţii se realizează regenerarea întregii linii în care se află locaţia respectivă citită.

Operaţia numai de regenerare a informaţiei din întreaga matrice este similară cu cea de la citire cu diferenţa că nu se mai comandă selectarea coloanei (CAS_L nu se activează, se încarcă linia selectată în registrul latch de linie şi apoi se înscrie

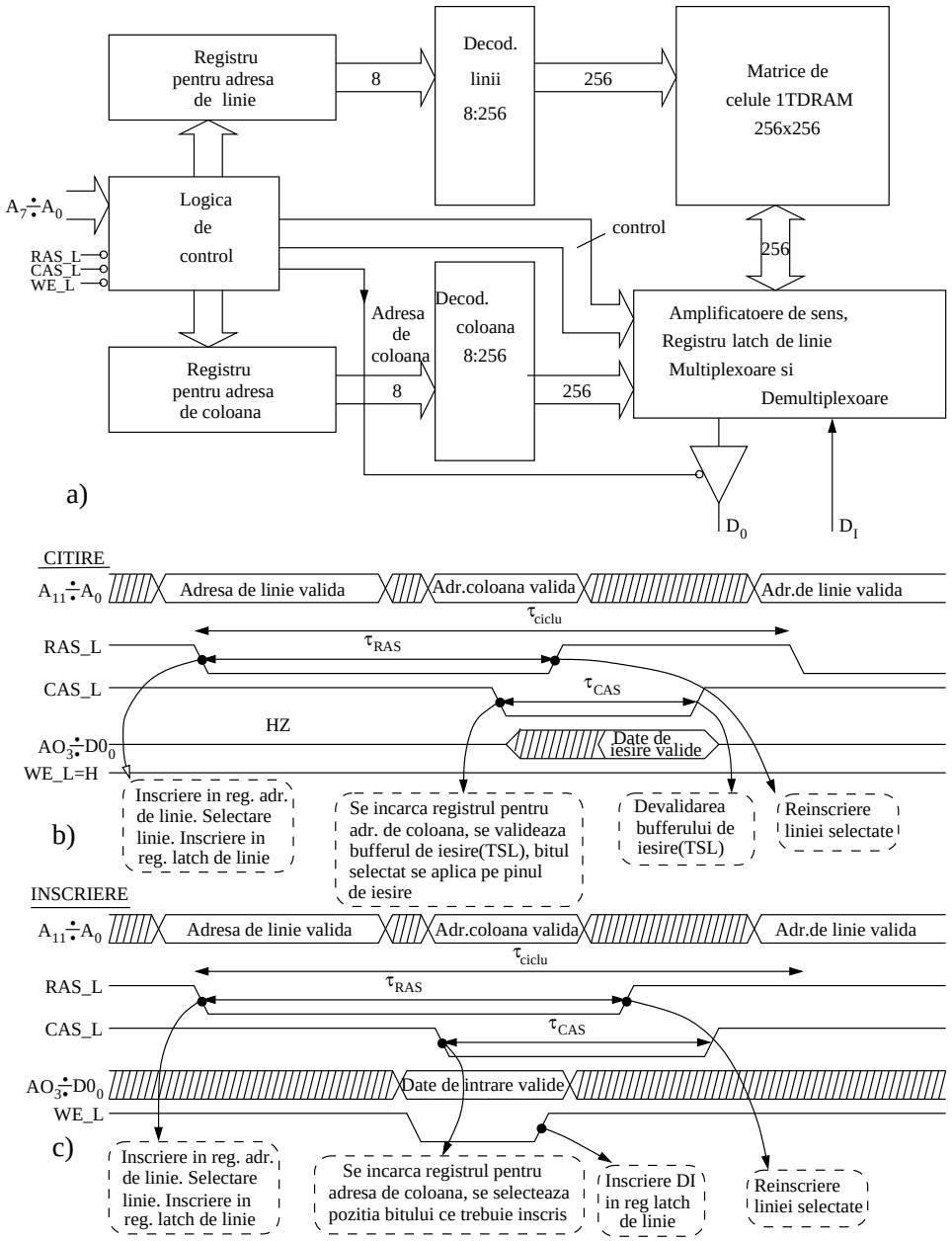


Figura 3.95 Circuitul de memorie DRAM: a) structurare de principiu pentru o memorie de 64Mb ($4096 \times 4096 \times 4$); succesiunea aplicării cuvântului de adresă, a datelor de ieşire/intrare şi a semnalelor de control în efectuarea operaţiilor de citire/regenerare (b) şi de înscriere (c).

înapoi linia cu acelaşi conţinut). Se aplică succesiv toate adresele liniilor de cuvânt, pe frontul negativ al semnalului RAS_L conţinutul unei linii se înscrie în registrul latch de linie iar la dezactivarea acestui semnal conţinutul din registrul latch de linie este înscris înapoi în linia respectivă; intervalul de timp după care o linie trebuie reîmprospătată este de ordinul ms. În principiu, partea de reîmprospătare trebuie să aibă un circuit timer care să producă intervale de timp după care trebuie pornită operaţia de reîmprospătare (Timer - un circuit numărător prestabil, comandat cu o frecvenţă stabilă) şi un numărător ce generează succesiv toate adresele de linie.

Înscrierea memoriei. Succesiunea de selectare a locaţiei din matricea de memorie este similară ca la operaţia de citire, dar semnalul WE_L trebuie activat înaintea semnalului CAS_L; pe durata RAS_L=0 bufferul de ieşire TSL va rămâne în starea HZ chiar dacă CAS_L devine activ. După ce conţinutul liniei selectate a fost înscris în registrul latch de linie (la fel ca la citire), poziţia din acest registru, determinată de selectare coloană, este înscrisă cu bitul de intrare DI la activarea semnalului de înscriere WE_L. La dezactivarea semnalului RAS_L se va înscrie în linia de cuvânt conţinutul din registrul latch de linie dar cu valoarea bitului DI în poziţia selectată.

Un procedeu prin care se simpifică procesul de reîmprospătare a memoriei este cel referit prin CAS înainte de RAS (CAS before RAS). Aplicarea acestui procedeu necesită existenţa unui numărător în interiorul circuitului de memorie care să genereze succesiv adresele de linii de cuvât (se elimină astfel aplicarea din exterior a adreselor de linie). După cum şi denumirea procedeului indică, se aplică semnalul CAS înaintea semnalului RAS, este reîmprospătată linia de cuvânt selectată de numărător şi, apoi, se incrementează numărătorul.

La DRAM, ca şi la SRAM, funcţionarea nu este dirijată de ceas, funcţionarea este de tip asincron şi este fixată prin fronturile H-L şi L-H ale semnalelor de control RAS_L şi CAS_L.

Sintetic funcţionarea memoriei dinamice se compune din (parametrii de timp din paranteze corespund unei memorii actuate de 4MB):

Adresarea liniei de cuvânt ($\sim 50ns$)

- se aplică adresa de linie şi se activează semnalul de strobare RAS_L
- întregul conţinut al liniei selectate/citite este înscris în registrul latch de linie
- conţinutul liniei citite este deteriorat

Adresarea coloanei ($\sim 10ns$)

- se aplică adresa de coloană şi se activează semnalul de strobare CAS_L
- pentru coloana selectată bitul:
  - la citire este transferat din registrul latch de linie la ieşire, D_O
  - la înscriere este înscris cu valoare DI

Reîmprospătare ($\sim 30ns$)

- se înscrie înapoi în linie întreg conţinutul din registrul latch de linie

Pentru o memorie dinamică principalii parametri din catalog sunt (valorile din paranteze corespund pentru o memorie de 4MB):

- τ_{RAC}, este intervalul de timp minim de la frontul de activare a semnalului RAS_L până în momentul când datele sunt valide la ieşirea bufferului TSL (60 ns). Viteza unui cip de memorie este specificată prin această valoare (de acces).

- τ_{RC}, este intervlul de timp minim din momentul (pornirii) accesării unei linii de cuvânt până în momentul accesării liniei de cuvânt următoare, adică distanţa mini-mă, în timp, între două activări ale semnalului RAS_L; este un timp de ciclu (110 ns).

- τ_{CAC}, este intervalul de timp minim din momentul activării semnalului se strob CAS_L până la obţinerea datelor valide la ieşierea bufferului TSL (15 ns).

- τ_{PC}, este intervalul de timp minim din momentul (pornirii) accesării unei coloane până în momentul accesării coloanei următoare; este un timp de ciclu (35 ns)

Următoarele valori reflectă dinamica memoriilor de tip DRAM: creştere de capacitate+60%/an; cost -30%/an; densitatea de integrare 2.5×celule/suprafaţă; creşterea suprafeţei waferului de 1.5 ori în 3 ani.

Memoria este o componentă foarte importantă în sistemele de calcul, unde, din ce în ce mai mult, i se impune un volum mare de schimb de date în unitatea de timp. Pentru creşterea volumului schimbului de date se acţionează: fie prin mărirea lungimii cuvântului cu care se lucrează, fie prin micşorarea latenţei memoriei (valorile date anterior arătau că timpul minim de acces consecutiv la liniile de cuvânt este $\tau_{\text{RC}} = 110ns$) sau fie, simultan, prin ambele modalităţi. Există diferite modalităţi prin care se obţine o latentă micşorată pentru memorie, amintim doar două: accesarea tip pagină şi accesarea tip cu ieşirea de date extinsă EDO (Extended Data Out). Acestea de fapt, realizează cât mai multe operaţii pe o aceeaşi accesare a unei linii de cuvânt, astfel încât operaţiile să nu mai fie o repetare de tipul CAS, RAS, CAS, RAS ....

Modul de accesare de tip pagină, prin adresa de linie, activează o întreagă pagină/linie de cuvânt (poate fi până la 8096 biţi) şi apoi prin schimbarea numai a adresei de coloană, conjugată cu activarea semnalului CAS, poate accesa oricare celulă cuprinsă în acea linie; trecerea la o altă pagină impune aplicarea noii adrese de linie şi reactivarea semnalului RAS. Succesiunea semnallor de strobare în aplicarea adreselor pentru o operaţie în modalitatea de acces tip pagină este: RAS, CAS, CAS, ..., CAS; RAS, CAS, CAS, ...

Modul de accesare tip EDO este de fapt tot o adresare de tip pagină numai că, după cum şi denumirea sugerează, se extinde timpul cât datele sunt valide pe ieşire. De data aceasta se exclude activarea succesivă a semnalului CAS pentru fiecare citire, iar starea normală a bufferului de ieşire TSL este comandată de un semnal de validare a ieşirii, OE_L. Deci datele, din registrul latch de linie, sunt valide la ieşire cât timp OE_L = 0, pe durata dintre începutul marcat de activarea semnalului CAS_L până la activarea următoare a acestui semnal. Pe această durată sunt generate succesiv adresele de coloană ale biţilor stocaţi în registrul latch de linie.

Din prezentările anterioare, atât la memoria statică cât şi la cea dinamică, rezultă că funcţionarea acestor circuite este de tip asincron. Pentru integrarea lor într-un sistem sincron este necesară "aducerii funcţionării în acelaşi timp"/(sincronizarea) cu ceasul sistemului. În modul sincron funcţionarea internă a circuitului de memorie rămâne aceeaşi dar semnalele cu exteriorul (cuvântul de adresă, semnalele de control, datele de intrare) sunt citite/(eşantionate) pe frontul pozitiv al semnalului de ceas şi

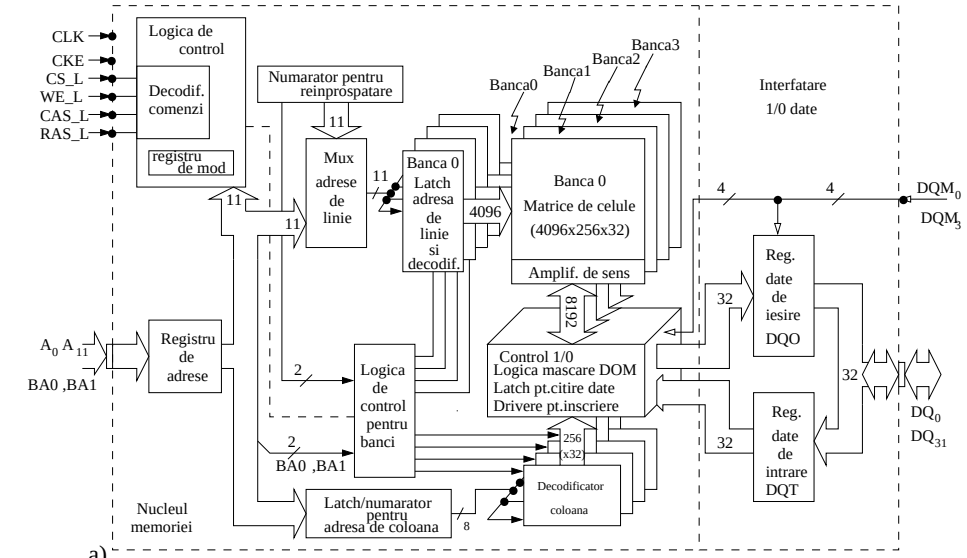
înscrise în registre interne ale circuitului de memorie (aceste circuite registru interne sunt componente ale interfeţei de sincronizare); de asemenea datele de ieşire sunt generate sincron (pe durata dintre două fronturi pozitive consecutive ale semnalului de ceas). Circuitele de memorie SRAM, DRAM înzestrate cu aceste interfeţe de Sincronizare sunt referite respectiv prin abreviaţiile SSRAM şi SDRAM.

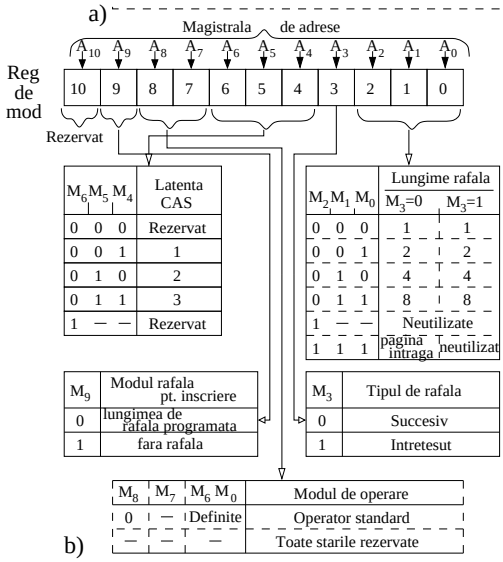
3.6.2.1 Memoria DRAM sincronă, SDRAM

Integrarea unei memorii DRAM, a cărei funcţionare de tip asincron este fixată prin fronturile H-L şi L-H ale semnalelor de control RAS_L şi CAS_L, într-un sistem sincron, comandat de ceas, necesită o anumită circuistică de interfaţare. Dar în ultimii ani, această circuistică de interfaţare a fost integrată pe acelşi cip cu memoria DRAM, iar pentru exterior (pentru utilizator) apare ca un circuit cu funcţionare sincronă, pe baza unui semnal de ceas, circutul fiind referit ca memorie dinamică Sincronă, SDRAM. Pentru prezentarea organizării, funcţionării şi comenzilor exterioare ale unei memorii dinamice sincrone s-a ales circuitul SDRAM 128Mb (×32) (Micron Technology), care, cu anumite simplificări, poate fi utilizat ca un circuit generic.

Organizarea circuitului de capacitate 128Mb este prezentată în Figura 3.96-a. Pentru circuitele DRAM de capacitate mare, datorită dificultăţilor de realizare a unei matrice pătratică de dimensiune ridicată, matricea este segmentată în mai multe submatrice, referite prin temenul de bancă sau de plan; aceste bănci funcţionează în paralel dar independent una de alta. Organizarea din această figură, de capacitatea 128Mb (2^{27} =134 217 728 biţi), este compusă din patru bănci acoperind un spaţiu de adresare de 4M adrese; numărul de 33 554 432 (2^{25}) biţi dintr-o bancă are ca suport o matrică cu 4096 linii ($A_{11} \div A_0$) şi 256 coloane ($A_7 \div A_0$), la fiecare locaţie de la intersecţia unei linii cu o coloană fiind un cuvânt de 32 biţi. Selectarea uneia din cele patru bănci se realizează cu biţii $BA1$ şi $BA0$.

La nivelul de celulă de stocare într-o bancă operaţiile de citire, scriere, reîmprospătare se fac prin succesiunea semnalelor CAS, RAS şi biţilor de intrare şi ieşire, la fel ca la memoria dinamică prezentată în secţiunea anterioară, diferenţa apare în modul cum aceste semnale interne se obţin, prin circuistica de interfaţare, din semnalele externe aplicate la pinii circuitului de memorie sincronă, deoarece acest circuit are o funcţionare programată care este sub controlul semnalului de ceas, CLK. Pe lângă semnalul de ceas mai există un semnal de validare a semnalului de ceas, CKE, deci efectul de sincronizare se realizează numai când acest semnal este activ, $CKE = H$. Pe fiecare front pozitiv al semnalului CLK, de frecvenţă maximă 166MHz, semnalele externe de control CS_L, WE_L, RAS_L şi CAS_L sunt înregistrate în blocul de decodificare comenzi, cuvântul de adresă $A_{11} \div A_{10}$ şi biţii de adresare a băncilor BA1, BA0 sunt înscrişi în registrele de adrese, iar cuvântul de date $DQ_{31} \div DQ_0$ este înscris în registrul de date de intrare DQI (pentru înscriere) sau este generat la ieşirea registrului de date de ieşire DQO (pentru citire) şi, de asemenea, cuvântul mască $DQM_3 \div DQM_0$, pentru datele de intrare/ieşire este înregistrat. Pentru fiecare byte din cuvântul de date intrare/ieşire corespunde un bit în cuvântul de mascare în felul următor: DQM_0 pentru $DQ_7 \div DQ_0$; DQM_1 pentru $DQ_{15} \div DQ_8$; DQM_2 pentru $DQ_{23} \div DQ_{16}$ şi DQM_3 pentru $DQ_{31} \div DQ_{24}$. Valoarea H a bitului de mascare DQM_i, constituie pentru byte-ul i corespunzător din cuvântul de date de intrare/ieşire o mască, adică bufferele de ieşire sunt în starea HZ deci nu se obţin date





lungime rafala	adresa coloanei de inceput	ordine de acces in rafala: Succesiv(M_3=0)	ordine de acces in rafala: Intretesut(M_3=1)
2	A_0		
		0 1	0 1
		1 0	1 0
4	A_1 A_0		
	0 0	0 1 2 3	0 1 2 3
	0 1	1 2 3 0	1 0 3 2
	1 0	2 3 0 1	2 3 0 1
	1 1	3 0 1 2	3 2 1 0
8	A_2 A_1 A_0		
	0 0 0	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
	0 0 1	1 2 3 4 5 6 7 0	1 0 3 2 5 4 7 6
	0 1 0	2 3 4 5 6 7 0 1	2 3 0 1 6 7 4 3
	0 1 1	3 4 5 6 7 0 1 2	3 2 1 0 7 6 5 4
	1 0 0	4 5 6 7 0 1 2 3	4 5 6 7 0 1 2 3
	1 0 1	5 6 7 0 1 2 3 4	5 4 7 6 1 0 3 2
	1 1 0	6 7 0 1 2 3 4 5	6 7 4 5 5 2 3 0 1
	1 1 1	7 0 1 2 3 4 5 6	7 6 5 4 3 2 1 0
pagina intreaga (256)	n=$A_0 \div A_7$ Locatii: $0 \div 256$	C_n, C_{n+1},C_{n+2}, C_{n+3},C_{n+4},.... ...$C_{n\,1}$,C_n,...	Neimplementat

c)

Figura 3.96 Memoria sincronă DRAM: a) organizare de principiu pentru un SDRAM de capacitate 4M × 32 compusă din patru bănci fiecare de dimensiune 4096 × 256 × 32; b),c) modalitatea de programare, prin cuvântul înscris în registrul de mod, a tipurilor de rafale pentru operaţiile de înscriere şi de citire.

de ieşire iar datele aplicate pentru înscriere nu vor fi stocate în memorie; datele de intrare/ieşire vor fi valide pentru înscriere/citire numai când valoarea testată a bitului respectiv de mască, DQM_i, este în L. Când cei patru biţi de mascare sunt consideraţi toţi în aceeaşi stare (H sau L), adică aceeaşi comandă pentru toţi cei patru bytes ai cuvântului de date, sunt referiţi în comun prin notaţia DQM.

Pentru realizarea unei operaţii de preîncărcare, citire, înscriere sau reîmprospătare comenzile necesare se obţin din valorile semnalelor aplicate din exterior pe pinii circuitului, eşantionate pe frontul pozitiv de ceas, dar aceste valori nu sunt interpretate individual ci interpretate împreună sub forma unui cuvânt de control; deci pentru SDRAM există un cuvânt de control, extern, format din biţii CS_L, WE_L, CAS_L, RAS_L, un cuvânt de adresare format din biţii $A_{11} \div A_0$ şi $BA1$, $BA0$ şi un cuvânt de mascare format din biţii $DQM_3 \div DQM_0$. (Toate semnalele aplicate pe pini sunt compatibile LVTTL, $V_{DD} = 3.3V$, vezi Figura 1.48-c.)

Operaţiile de citire şi înscriere sunt pipelinizate, adică pe fiecare front al impulsului de ceas se generează sau se înscrie câte un cuvânt de 32 biţi. Această pipelinizare este realizată atât la o **accesare aleatorie** a locaţiilor cât şi la o **accesare de tip rafală (burst)** - accesarea porneşte de la o locaţie selectată continuând apoi, fără întrerupere (rafală), cu un număr prescris de locaţii într-o succesiune programată. Programarea succesiunii, conţinută în cuvântul de mod $M_{10} \div M_0$, pentru modul de accesare de tip rafală se realizează în exterior şi se înscrie, prin intermediul cuvântului de adresă $A_{10} \div A_0$, în registrul de mod, Figura 3.96-b; această programare se păstrează până la o nouă încărcare cu un cuvânt de mod sau până la anularea tensiunii de alimentare. Pipelinizarea cuvintelor de date pentru înscriere sau pentru citire este asigurată şi de existenţa a patru bănci, selectate cu biţii $AB1$, $AB0$, care operează independent şi în paralel; de exemplu în timp ce într-o bancă se realizează o operaţie de înscriere în alta se iniţiază o operaţie de citire.

Pentru o succesiune de tip rafală a accesărilor, programarea prin cuvântul, $M_{10} - M_0$, înscris în registrul de mod, prescrie următoarele caracteristici: lungimea rafalei, tipul de rafală, latenţa CAS, modul de operare şi modul de înscriere. Rafala este efectuată după ce o linie de cuvânt dintr-o bancă a fost activată şi deci se realizează numai prin activarea într-o anumită succesiune a coloanelor(în cadrul unei linii de cuvânt).

- Lungimea rafalei determină, prin subcuvântul $M_2 \div M_0$, numărul maxim de locaţii (de coloane) care sunt selectate pentru a fi accesate la o comandă de înscriere sau de citire. Pentru o lungime de 2 (locaţii) a rafalei, $M_2M_1M_0 = 001$, este selectat, de pe linia activată, blocul de coloane cu lungimea 2, fixat prin cuvântul de adresă $A_7 \div A_1$, iar în interiorul acestui bloc citirea sau înscrierea porneşte de la coloana cu numărul 0 sau 1, fixat prin valoarea lui A_0; pentru o lungime a rafalei de opt locaţii, $M_2M_1M_0 = 011$, este selectat un bloc cu lungimea de opt coloane fixat prin cuvântul de adresă $A_7 \div A_3$ iar în interiorul blocului citirea sau înscrierea porneşte de la coloana specificată de subcuvântul $A_2A_1A_0$. Pentru o lungime de o pagină se porneşte de la coloana de adresă fixată de cuvântul $A_7 \div A_0$, Figura 3.96-c. Dacă în cadrul blocului sau paginii respective se ajunge la adresa superioară de coloană se sare la prima adresă de coloană dacă nu s-au parcurs toate adresele (fixate prin lungimea programată a rafalei) din blocul sau pagina respectivă. Ordinea de parcurgere a adreselor, programată prin tipul de rafală, M_3, poate fi în mod succesiv sau în mod întreţesut (ordinea întreţesută nu este permisă la tipul de rafală pagină).

- Latenţa CAS, programată prin $M_6M_5M_4$, fixează întârzierea, în număr de tacturi de ceas, din momentul înregistrării comenzii (frontul pozitiv al semnalului de ceas) aplicată la pinii circuitului SDRAM până în momentul când cuvântul citit din memorie este disponibil ca dată validă de ieşire, $DQ_{32} \div DQ_0$; latenţa poate avea valorile 1, 2 sau 3 tacturi de ceas.

- Modul de operare standard, atât pentru citire cât şi pentru înscriere. este fixat prin subcuvântul $M_8M_7 = 00$, alte valori pentru acest subcuvânt fixează moduri de operare pentru testare.

- Modul de rafală pentru înscriere. Când $M_9 = 0$ atât pentru înscriere cât şi pentru citire se efectuează rafală de lungime programată. Pentru $M_9 = 1$ se aplică rafală de lungime programată numai pentru citire, înscrierea nu se face în mod rafală (se înscrie o singură locaţie).

Pentru circuitul SDRAM se obţine cu flux continuu (pipeline) de date de intrare (înscriere) sau de ieşire (citire), câte un cuvânt de date, $DQ_{31} \div DQ_0$, pe fiecare tact de ceas, indiferent dacă accesarea coloanelor se face în modul rafală sau aleatoriu; această operare este programată prin aplicarea din exterior a cuvintelor: de comandă, de adresă şi de mască la pinii circuitului. Pe durata unui tact de ceas circuitul realizează operaţiile exprimate prin cuvintele aplicate din exterior la pinii circuitului, înregistrate/eşantionate pe frontul pozitiv al tactului respectiv de ceas. Conţinutul acestor cuvinte şi succesiunea lor în timp, pentru realizarea anumitor operaţii, se efecturază în exterior de către circuitul controller pentru SDRAM. În continuare, se vor prezenta, pentru realizarea operaţiilor pe o memorie sincronă DRAM, cuvintele de control.

În tabelul din Figura 3.97-a sunt sintetizate comenzile pentru un SDRAM de capacitate 4M $\times$ 32 biţi.

- Comanda INHIBARE (NOP). Prin această comandă se elimină posibilitatea ca circuitul să mai primească oricare altă comandă, chiar dacă semnalul de ceas CLK este validat (CKE = H); dar operaţiile în curs pe care le efectuează SDRAM sunt continuate. De fapt prin comanda INHIBARE este deselectat circuitul SDRAM, CS_L = H.

- Comanda Nici-o-OPeraţie, NOP. Pentru SDRAM, care este deja selectat (CS_L=0), comanda NOP face ca circuitul să nu accepte o altă comandă; operaţiile în curs sunt continuate (se introduce când se doreşte doar consumarea unui timp).

- Comanda ÎNCĂRCARE REGISTRUL DE MOD (Load Mod Register). Conţinutul cuvântului de mod $M_{10} \div M_0$, vezi Figura 3.96-b, este încărcat, din exterior, în registrul de mod prin intermediul cuvântului de adresă $A_{10} \div A_0$. Încărcarea registrului de mod se poate realiza numai când toate cele patru bănci nu sunt activate; după încărcare o următoare comandă pentru memorie se poate aplica numai după intervalul de timp τ_{MRD}.

- Comanda ACTIVARE. Pentru a realiza o operaţie de citire sau înscriere de pe o linie dintr-o bancă aceasta trebuie întâi deschisă (activată acea linie), iar această deschidere se realizează prin comanda ACTIVARE. Prin conţinutul cuvintelor $BA1\,BA0$ şi $A_{11} \div A_0$, din momentul înregistrării/eşantionării comenzii ACTIVARE, este deschisă una din cele patru bănci ($BA1\,BA0$) şi în acea bancă o linie de cuvânt; apoi pot urma operaţii de citire şi/sau înscriere la locaţiile de pe linia deschisă. Se impune un interval τ_{RCDmin} din momentul înregistrării comenzii ACTIVARE până la comanda unei operaţii de citire sau înscriere. De exemplu, dacă $\tau_{RCDmin} = 20ns$ iar

frecvenţa la care se comandă memoria este $f_{CLK} = 125MHz$ ($T_{CLK} = 8ns$) rezultă că, raportat la frontul de ceas care a înregistrat comanda ACTIVARE, următoarea operaţie de citire sau înscriere se poate aplica numai la al treilea front pozitiv de ces, $\lceil \tau_{RCDmin}/T_{CLK} \rceil = 3$, Figura 3.97-b. Succesiunea cuvintelor de comnadă generate de controllerul de memorie este: T_0 - ACTIVARE; T_1 - NOP; T_2 - NOP; T_3 - READ/WRITE ...

Următoarea comandă ACTIVARE, pentru deschiderea unei alte linii de cuvânt din aceeaşi bancă se poate aplica numai după ce linia deschisă anterior a fost închisă/dezactivată (prin aplicarea unei operaţii de preîncărcare); se impune un interval minim de timp τ_{RCmin} între două comenzi ACTIVARE consecutive. Dar o următoare comandă ACTIVARE la o altă bancă se poate genera chiar când banca prezentă este deschis ceea ce duce la o reducere/(ascundere) a timpului total de acces la o linie de cuvânt.

- Comanda READ. Această comandă este utilizată pentru iniţierea unei operaţii de citire de tip rafală într-o linie de cuvânt deja activată. În momentul înregistrării comenzii READ se eşantionează şi cuvântul de adresă: biţii $BA1$, $BA0$ fixează banca iar biţii $A_7 \div A_0$ selectează coloana de la care se începe citirea; iar bitul A_{10}, din cuvântul de adresă, determină modul de preîncărcare: $A_{10} = H$, preîncărcarea automată este selectată; $A_{10} = L$ preîncărcarea automată ste devalidată. Dacă preîncărcarea automată este selectată atunci linia deja activată va fi preîncărcată (dezactivată) la sfârşitul efectuării rafalei de citire, iar dacă preîncărcarea automată este deselectată atunci linia deja activată rămâne în continuare activată şi pentru următorul acces.

Odată iniţiată comanda READ rafală, pornind de la coloana selectată, se obţine pe ieşire, după o latentă CAS prescrisă (vezi Figura 3.96-b), un flux continuu de date DQ_{out}, câte un cuvânt pe fiecare front pozitiv de ceas, evident dacă, cuvântul de mască DQM a fost în L. Cuvântul de mască va trece ieşirea memoriei în HZ cu o întârziere de două tacte după ce valoarea acestui cuvânt a fost înregistrată în starea H. După completarea unei citiri în mod rafală, considerând că nu a fost iniţiată o altă comandă, ieşirea va trece în HZ; la o pagină citită în mod rafală după citirea ultimei coloane din pagină (255) se continuă cu prima coloană până se ajunge la coloana din faţa primei coloane selectate. Datele unei citiri în rafală pot fi continuate cu alte date citite de către o următoare/(nouă) rafală, iar noua rafală poate fi aplicată începând de la terminarea datelor rafalei curente sau chiar de la oricare front pozitiv de ceas din interiorul rafalei curente (trunchiere); dar noua comandă READ rafală trebuie să fie aplicată cu un număr x de tacturi înainte de tactul pe care se vor obţine date valide pe ieşire, unde $x =$ (latenţa CAS) $- 1$. Deci prin comenzi READ succesive se poate menţine un flux de date continuu pe ieşire indiferent dacă aceste comenzi accesează aceeaşi bancă sau bănci diferite. Se poate obţine pe ieşire un flux continuu de date citite care pot alterna cu date înscrise dacă se intercalează şi comenzi WRITE rafală (cu condiţia să nu apară concurenţa de date la intrare/ieşire); dar comanda WRITE trebuie să fie aplicată chiar pe frontul pozitiv de ceas pe care se înscriu datele.

Operaţia de citire sub o rafală de lungime fixă sau de o pagină se poate termina/(trunchia) prin aplicarea la banca respctivă a comenzii PREÎNCĂRCARE (PA), această aplicare trebuie efectuată cu x tacturi înainte de tactul care mai citeşte date valide. De exemplu, în Figura 3.97-c s-a aplicat comanda READ rafală pe T_0, aici de lungime nespecificată, din banca 2, începând de la coloana n, cu latenţa CAS = 3,

Cuvintele de comanda pentru memoria SDRAM, 1Mb x 32 x 4 banci

Denumirea comenzii	CUVANTUL DE : comanda CS_L	RAS_L	CAS_L	WE_L	Masca DQM	ADRESA	Date DQ
Comanda de INHIBARE	H	–	–	–	–	–	–
NICI O OPERATIE (NOP)	L	H	H	H	–	–	–
ACTIVARE (selecteaza o banca si activeaza o linie)	L	L	H	H	–	Banca ← BA1,BA0 Linia ← $A_{11} \div A_0$	–
READ (Selecteaza o banca si o coloana, porneste operatia READ in rafala)	L	H	L	H	L/H	Banca ← AB1,AB0 Coloana ← $A_7 \div A_0$	Date valide
WRITE (Selecteaza o banca si o coloana, porneste operatia WRITE in rafala)	L	H	L	L	L/H	A_{10}=1 Preincarcare automata A_{10}=0 Preincarcare automata devalidata	Date active
TERMINARE RAFALA	L	H	H	L	–	–	–
PREINCARCARE (Dezactiveaza linia deschisa din banca/banci),PA	L	L	H	L	–	A_{10}=H dezact. bancile A_{10}=H dezact. banca	–
REFRES sau AUTO REFRES	L	L	L	H			–
INCARCARE REGISTRU DE MOD	L	L	L	L	–	$A_{11}A_{10}$.....A_0 – M_{10}.....M_0	–

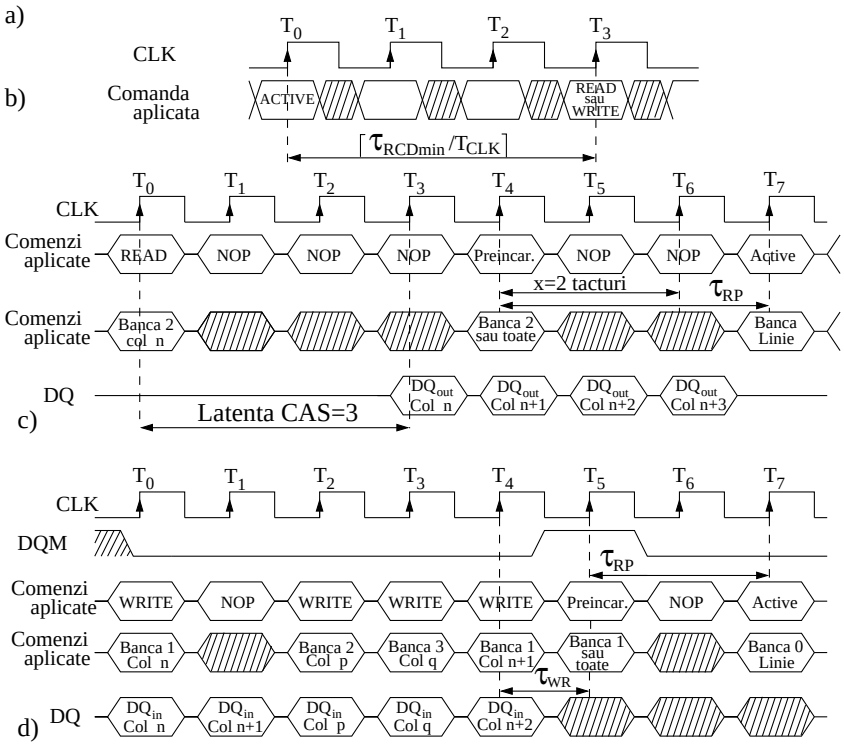


Figura 3.97 Comanda memoriei sincrone SDRAM: a) comenzile aplicabile din exterior pentru controlul funcţionării circuitului SDRAM; exemple de diagrame de semnale care ilustrează funcţionarea memoriei la aplicarea comanzilor de: ACTIVARE (b), READ (c) şi WRITE (d).

iar după citirea a patru date consecutive (de la coloanele n, $n+1$, $n+2$, $n+3$) se termină rafala prin aplicarea comenzii PREÎNCĂRCARE; succesiunea de comenzi generată de controller este: T_0 - READ; T_1 - NOP; T_2 - NOP; T_3 - NOP; T_4 - PREÎNCĂRCARE; T_5 - NOP; T_6 - NOP. De notat că timpul de prescriere x pentru comanda PREÎNCĂRCARE apare ca fiind ascuns deoarece simultan se fac şi citirile de date succesive. După comanda PREÎNCĂRCARE o următoare comandă la aceeaşi bancă nu se poate aplica decât numai după un inteval prescris de timp τ_{RP}. De asemenea, trunchierea/terminarea unei rafale de o anumită lungime sau de o pagină se poate realiza şi cu o comandă TERMINARE RAFALĂ (cu condiţia să nu fi fost prescrisă autopreîncărcarea, $A_{10} = H$), care trebuie aplicată cu x tacturi înainte de ultimul tact care generează date valide.

- Comanda WRITE. Înregistrarea comanzii WRITE iniţiază un acces pentru o înscriere rafală la o coloană de adresă $A_7 \div A_0$ într-o bancă fixată prin $BA1$, $BA0$ în care a fost deja activată o linie de cuvânt; dacă $A_{10} = H$ se realizează o preîncărcare automată la terminatea rafalei. Primul cuvânt de date de intrare DQ_{int} se înscrie în coloana de adresă $A_7 \div A_0$ pe primul front pozitiv de ceas, când se înregistrează comanda WRITE, apoi, câte un cuvânt de intrare pe fiecare din fronturile pozitive de ceas următoare până la completarea lungimii rafalei sau paginii (pentru mod pagină); la terminare, dacă nu mai există altă comandă, intrarea DQ va trece în HZ şi va ignora oricare dată aplicată. O comandă WRITE poate fi urmată de o altă comandă WRITE pe oricare front pozitiv următor, în consecinţă se poate înscrie un flux continuu de date în aceeaşi bancă sau în bănci diferite.

Fluxul de date pentru o comandă WRITE poate fi trunchiat/(terminat) printr-o comandă READ sau PREÎNCĂRCARE. De exemplu, în Figura 3.97-d se realizează o înscriere rafală, cu lungimea 2, în banca 1 începând cu coloana n, apoi trei înscrieri consecutive: prima în banca 2, începând de la coloana p, a două în banca 3, începând cu coloana q iar a treia tot în banca 1, începând cu coloana $n+2$. În banca 2 nu se înscriu în rafală două coloane deoarece este trunchiată de comandă WRITE pentru banca 3, dar nici în banca 3 nu se înscriu două coloane în rafală deoarece este trunchiată de comanda WRITE pentru banca 1. Dar şi în banca 1 se înscrie tot numai o coloană (col $n+2$) deoarece este trunchiată prin aplicarea comenzii PREÎNCĂRCARE (dată pentru banca 1 sau pentru toate). În aplicarea comenzii PREÎNCĂRCARE trebuie respectat intervalul de timp τ_{WR} începând de la frontul pozitiv al ultimei date care se doreşte a fi înscrisă (în acest caz T_4); numărul de tacturi pentru acest interval rezultă în funcţie de frecvenţa de ceas cu care se comandă memoria (in general τ_{WR} se acoperă prin 1-2 tacturi de întârziere până la aplicarea comenzii PREÎNCĂRCARE). Semnalul DQM trebuie pus în H imediat după ultima dată înscrisă până la aplicarea comenzii PREÎNCĂRCARE pentru a masca în continuare intrarea datelor de înscriere. (În cazul când PREÎNCĂRCARE se aplică la teminarea lungimii rafalei, DQM nu se pune în H, dar intervalul τ_{WR} trebuie respectat şi în acest caz.) După comanda PREÎNCĂRCARE aplicarea comenzii ACTIVARE nu se poate aplica decât numai după intervalul τ_{RP}.

- Comanda PREÎNCĂRCARE este utilizată pentru dezactivarea unei linii activate într-o bancă sau a liniilor activate în toate băncile. Dacă în momentul înregistrării comenzii PREÎNCĂRCARE bitul A_{10}, din cuvântul de adresă, este H atunci toate băncile sunt preîncărcate, iar dacă A_{10} este în L atunci numai banca al cărui număr este dat prin cuvântul $BA1BA0$ va fi preîncărcată. Odată preîncărcată o bancă/băn-

cile este în continuare în stare inactivă, deci înainte de a aplica o comandă READ sau WRITE banca respectivă trebuie să fie activată (prin comanda ACTIVARE); după comanda PREÎNCĂRCARE următoarea comandă (ACTIVARE) nu poate fi aplicată înainte de τ_{RP}.

Preîncărcarea automată realizează, la fel, o preîncărcare a unei bănci dar nu este iniţiată prin aplicarea cuvântului de comandă PREÎNCĂRECARE. Preîncărcarea automată este prescrisă odată cu comenzile WRITE sau READ dacă bitul $A_{10} = H$, iar operaţia de preîncărcare se efectuează (automat) la terminarea lungimii de rafală, cu excepţia lungimii de rafală pagină când preîncărcarea nu se execută la terminarea parcurgerii paginii.

Exemplul 3.33 Să se proiecteze un controller pentru o memorie dinamică sincronă, SDRAM_CNTRL, având următoarele date impuse:

- Memorie SDRAM (Micron Technology),128Mb ($1Mb \times 32 \times 4$ bănci),MT48LC4M32B2.
- 1 client (μP): apelează scriere şi citire, operaţii sincronizate cu SDRAM_CNTRL ($4M \times$ 32 biţi).
- Gestiunea automată a reîmprospătării memoriei.
- Frecvenţa de lucru: 100 MHz ($T_{CLK} = 10ns$).
- Rafală secvenţială cu lungimea 4, latentă CAS = 2 tacte.
- Adresă multiplu de 4.

Port	Descriere semnal
CLK	CLK, semnal de ceas comun pentru: client, SDRAM_CNTRL, SDRAM.
RESET	RESET_L, semnal de resetare (repornire)
Interfaţarea cu clientul	
RQ	RQ, semnalul de acces (generat de client)
ACK	ACK, semnalul de confirmare (generat de SDRAM_CNTRL)
RW	R/W_L, semnalul pentru operaţiile de citire/scriere (generat de client)
ADR_CLIENT[21:0]	$A_{21} \div A_0$, cuvânt de adresă generat de client (bancă+linie+coloană)
DQO[31:0]	$DQ_{31} \div DQ_0$, cuvânt de date citite din SDRAM
DQ_I[31:0]	$DQ_{31} \div DQ_0$, cuvânt de date pentru înscrierea în SDRAM
Interfaţarea cu SDRAM	
CS	CS_L, semnal selectare SDRAM
CKE	CKE, semnal validare ceas (se consideră permanent în H)
BA[1:0]	$BA1BA0$, cuvânt selectare bănci
ADR[11:0]	$A_{11} \div A_0$, cuvânt adresă selectare linie; $A_7 \div A_0$, cuvânt adresă selectare coloană
RAS	RAS_L, semnal strobare adresă linie ($A_{11} \div A_0$)
CAS	CAS_L, cuvânt strobare adresă colonă ($A_7 \div A_0$)
WE	WE_L, semnal validare înscriere date
DQM[3:0]	$DQM_3 \div DQM_0$, cuvânt pentru mascare date (pe byte)
DQ[31:0]	$DQ_{31} \div DQ_0$, cuvânt de date bidirecţionale

Soluţie. Modul de interfaţare, cu specificarea semnalelor (direcţie, activare, lungime de cuvânt) între client (μP) şi SDRAM prin intermediul controllerului este prezentat sub formă de schemă bloc în Figura 3.98-a. De asemenea, este prezentată interdependenţa între semnalele cerere acces, RQ (de la clent) şi confirmare, ACK (de la controller), în realizarea

protocolului pentru operaţia de citire şi de înscriere, Figura 3.98-b. Cuvântul adresă pentru o locaţie în memorie, generat de client, este segmentat în subcuvintele pentru adresă bancă, linie şi coloane, de către controller şi transmis în această forma la SDRAM, Figura 3.98-c.

Controllerul va fi implementat sub forma unui automat (ASM). Organigrama ASM, Figura 3.99-a se compune din trei părţi: înitializarea memoriei, reîmprospătarea memoriei şi operaţiile de memorie (activare, citire, înscriere şi preîncărcare). La conectarea tensiunii sau activarea, RESET_L = 0, circuitul de memorie trebuie iniţializat şi aceasta constă din: o comandă de PREÎNCĂRCARE (PA), două comenzii de AUTOREFRESH ($AR1$, $AR2$) şi o comandă de programare (ÎNCĂRCARE REGISTRU MOD, LMR).

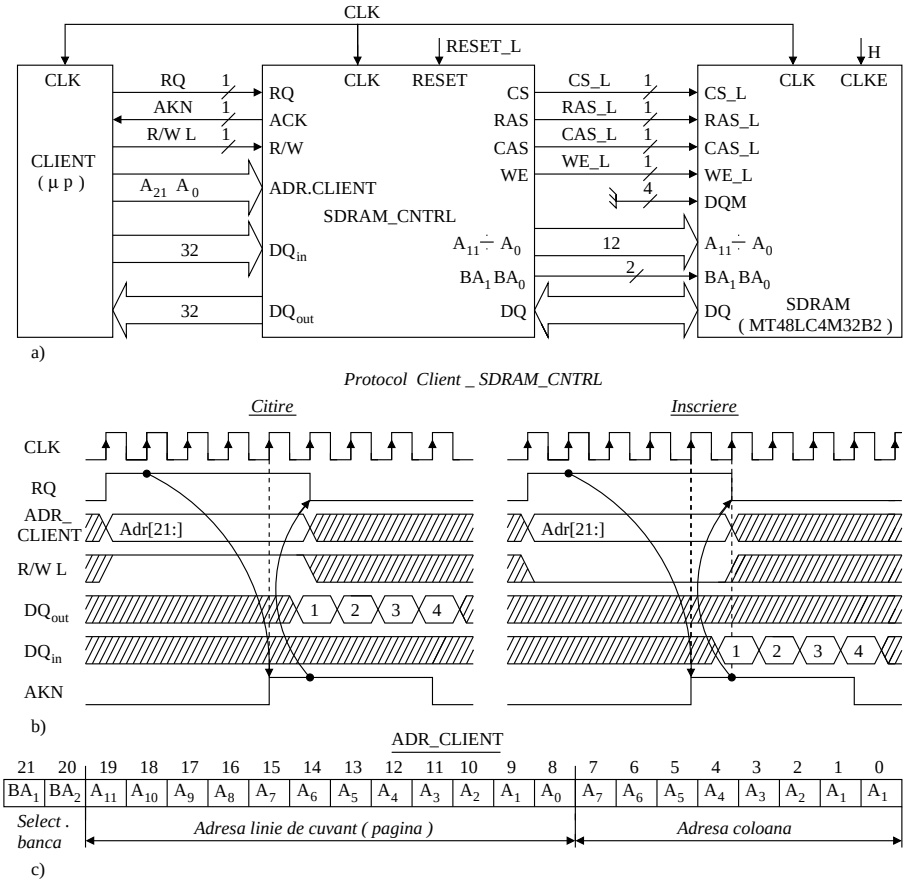


Figura 3.98 Explicativă pentru exemplul 3.33: a) schema bloc a interfaţării CLIENT-SDRAM_CNTRL şi SDRAM_CNTRL-SDRAM; b) semnalele de protocol CLIENT-SDRAM_CNTRL pentru operaţia de citire şi înscriere; c) subcâmpurile pentru adresă bancă, linie şi coloană în cuvântul de adresă generat de client.

În starea (0) de iniţializare, Init_st, automatul va staţiona 17 tacte de ceas pe durata cărora, cu anumite determinări de temporizare (prescrise de fabricant), se generează comenzile PA, AR1, AR2 şi LMR; pe baza acestor comenzi controllerul va calcula semnalele care se aplică la SDRAM. Automatul rămâne în starea (0) pe durata a 17 tacte de ceas când pe

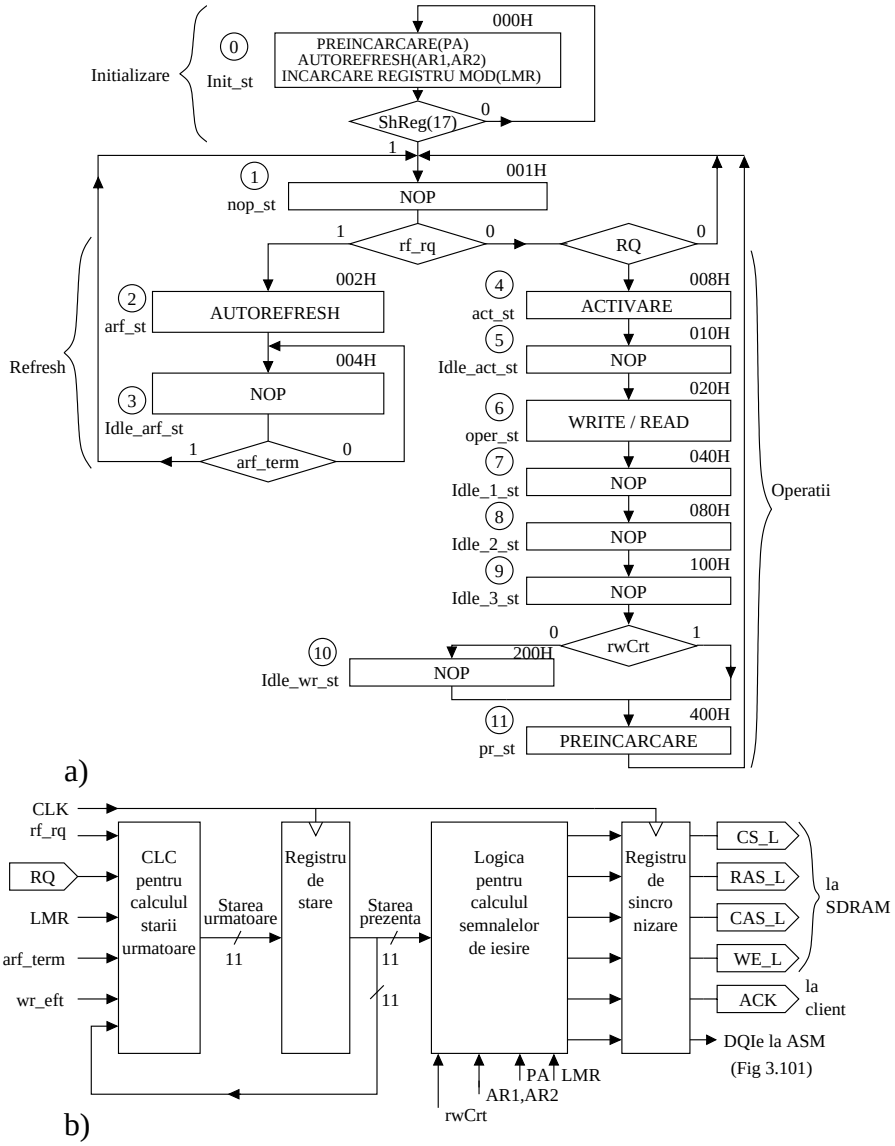


Figura 3.99 Explicativă pentru exemplul 3.33: a) organigrama ASM pentru funcţionarea SDRAM_CNTRL; b) structurarea contrlollerului ca un automat Moore cu întârziere.

baza unui registru de deplasare de iniţializare, initShReg, se generează cele patru comanzi. La RESET_L = 0 se înscrie 1 în prima celulă a registrului ShReg(0) = 1. Prin deplasarea valorii 1 din celulă în celulă rezultă: o comandă de PREINCĂRCARE, ShReg(1) = 1 = PA; două comenzi de AUTOREFRESH, ShReg(3) = 1 = $AR1$ şi $ShReg(10) = 1 = AR2$; o comandă pentru încărcarea registrului de mod ShReg(17) = 1 = LMR (când se trece la stare următoare).

Refresh-ul memoriei trebuie făcut la fiecare 64 ms, ceea ce înseamnă că la o repartizare uniformă a reîmprospătării celor 4096 linii ale matricei, intervalul în timp între reîmprospătarea a două linii este $\tau_{ref} = 64ms/4096 = 16\mu s$, iar raportat acest interval la frecvenţa de ceas de 100 MHz ($T_{CLK} = 10ns$) rezultă că după un număr de $16\mu s/10ns = 1600$ perioade de ceas trebuie dată o comandă de AUTOREFRESH (se consideră acoperitor 1500 perioade).

Semnalul cerere refresh, rf_rq, este produs de un numărător cu prescriere (rfCounter), cu numărare în sens invers, $CU/\overline{CD} = 0$, ca în Figura 3.100-c, al cărui semnal RCO comandă un latch SR (iar numărătorul începe un nou ciclu de 1500 de tacte). Ieşirea Q a latch-ului SR este semnalul rf_rq de cerere pentru aplicarea comenzii AUTOREFRESH; când rf_rq=1 se trece în starea (2) a automatului. La generarea comenzii AUTOREFRESH, în starea (2), arf_st, latchul SR este resetat şi, în acelaşi timp, se înscrie valoarea 1 în poziţia rdr(0) a registrului durată reîmprospătare, rdr, cu lungimea de 7 biţi. Se intră în bucla de aşteptare din starea (3), Idle_arf_st, pe durata procesului de reîmprospătare, până la activarea semnalului arf_term; durata procesului de reîmprospătare pentru o linie de cuvânt, fixată la şapte tacturi de ceas, este realizată prin parcurgerea registrului de deplasare stânga, rdr (registru durată refresh), până la celula a şaptea când rdr(6)=1=arf_term, Figura 3.100-d.

In partea de operaţii a diagramei ASM se intră când nu există o cerere de reîmprospătare, rf_rq = 0, şi este activată cererea de acces de la client, RQ = 1. Prima operaţie, care trebuie efectuată după o reîmprospătare sau după o preîncărcare, este cea de activare, ceea ce se realizează în starea (4), act_st, prin comanda ACTIVARE. La linia de cuvânt activată o următoare comandă READ/WRITE, generată în starea (6), oper_st, poate fi aplicată numai după intervalul de timp τ_{RCD}; starea (5), Idle_act_st, care generează comanda NOP este introdusă tocmai pentru acoperirea întârzierii τ_{RCD}. Pe tacturile corespunzătoare stărilor (6), (7), (8), (9) se realizează rafala cu lungimea de patru tacturi pentru operaţia de înscriere sau citire aplicată în starea (6).

O rafală cu lungime fixă, pentru citire sau înscriere, poate fi urmată, sau trunchiată/ întreruptă printr-o comandă PREÎNCĂRCARE (starea (11), pr_st), la aceeaşi bancă (dacă nu a fost prescrisă AUTO-PREÎNCĂRCARE, $A_{10} = 1$). Dar, dacă se efectuează o rafală de înscriere atunci comanda PREÎNCĂRCARE nu se poate genera decât numai după consumarea intervalului de timp τ_{WR}, vezi Figura 3.97-d, de la aplicarea pe intrarea de date a memoriei a ultimului cuvânt de date care se doreşte a fi înscris. În consecinţă, la terminarea rafalei de patru date se testează dacă operaţia curentă a fost de înscriere, rwCrt = 0, şi în caz afirmativ se intră în starea (10), Idle_wr_st care asigură acoperirea intervalului τ_{WR}. După aplicarea operaţiei de preîncărcare, starea (11), aplicarea unei noi comenzi ACTIVARE, pentru selectarea unei alte linii de cuvânt, se poate aplica numai după consumarea intevalului de timp τ_{RP} (vezi Figura 3.97-c) ceea ce se realizează prin comanda NOP din starea (1), nop_st. Variabila de testat rwCrt se generează ca ieşirea unui latch SR comandat prin conjuncţia $RQ \cdot R, W\_L$.

Tabelul de tranziţie al stărilor şi al ieşirilor (comenzilor), corespunzător organigramei ASM, este prezentat in Figura 3.100-a; rezultă că automatul este de tip Moore (comenzile

ShReg(17)	rf_rg	arf_tm	RQ	wr_eft	Starea prezenta $z_{10}z_9.....z_1z_0$ nr	Denumire	Cod	Starea urmatoare $w_{10}w_9.....w_1w_0$ nr	Denumire	Cod	Comenzi	CS_L	RAS_L	CAS_L	WE_L
0	–	–	–	–	⓪	Init_st	000 H	⓪	Init_st	000 H	PREINCARCARE(PA) AUTOREFRESH(AR1,AR2)	L L	L L	H L	L H
1								①	nop_st	001 H	LOAD MOD REG(LMR)	L	L	L	L
–	0	–	0		①	nop_st	001 H	⓪	nop_st	000 H	NOP	L	H	H	H
			1	–				④	act_st	008 H					
	1							②	arf_st	002 H					
					②	arf_st	002 H	③	Idle_arf_st	004 H	AUTO_REFRESH	L	L	L	H
–	–	0	–	–	③	Idle_arf_st	004 H	③	Idle_arf_st	004 H	NOP	L	H	H	H
		1						①	nop_st	001 H					
–	–	–	–	–	④	act_st	008 H	⑤	Idle_act_st	010 H	ACTIVARE	L	L	H	H
–	–	–	–	–	⑤	Idle_act_st	010 H	⑥	oper_st	020 H	NOP	L	H	H	H
–	–	–	–	–	⑥	oper_st	020 H	⑦	Idle_1_st	040 H	READ/WRITE	L	H	L	*
–	–	–	–	–	⑦	Idle_1_st	040 H	⑧	Idle_2_st	080 H	NOP	L	H	H	H
–	–	–	–	–	⑧	Idle_2_st	080 H	⑨	Idle_3_st	100 H	NOP	L	H	H	H
–	–	–	–	0	⑨	Idle_3_st	100 H	⑩	Idle_wr_st	200 H	NOP	L	H	H	H
				1				⑪	pr_st	400 H					
–	–	–	–	–	⑩	Idle_wr_st	200 H	⑪	pr_st	400 H	NOP	L	H	H	H
–	–	–	–	–	⑪	pr_st	400 H	①	nop_st	010 H	PREINCARCARE	L	L	H	L

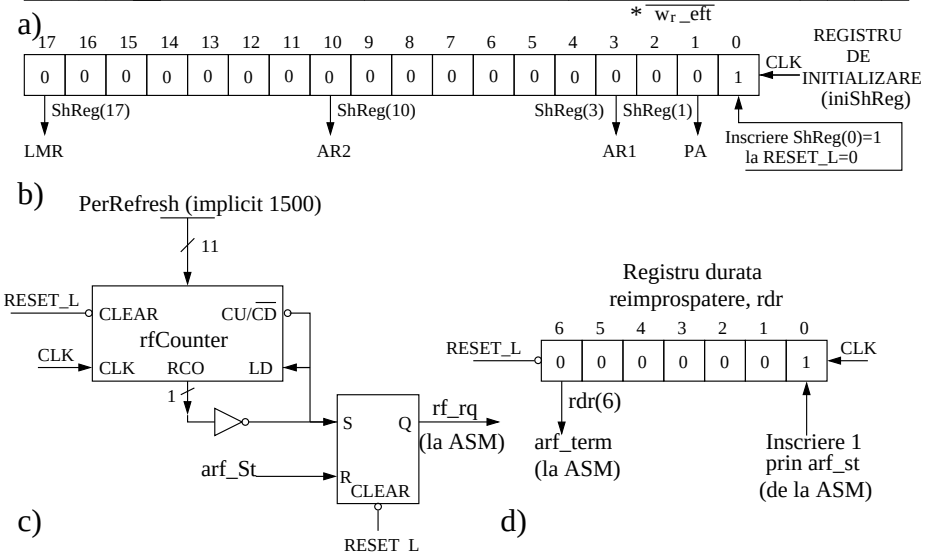


Figura 3.100 Explicativă pentru Exemplul 3.33: a) tabelul de tranziţie al stărilor şi tabelul ieşirilor (comenzilor) pentru SDRAM_CNTRL; b) generarea comenzilor în starea de iniţializare pe baza unui registru de deplasare, ShReg; c) structura generatorului pentru intervalele de timp de reîmprospătare la o linie de cuvânt; d) structura circuitului pentru generarea duratei procesului de reîmprospătare la o linie de cuvânt.

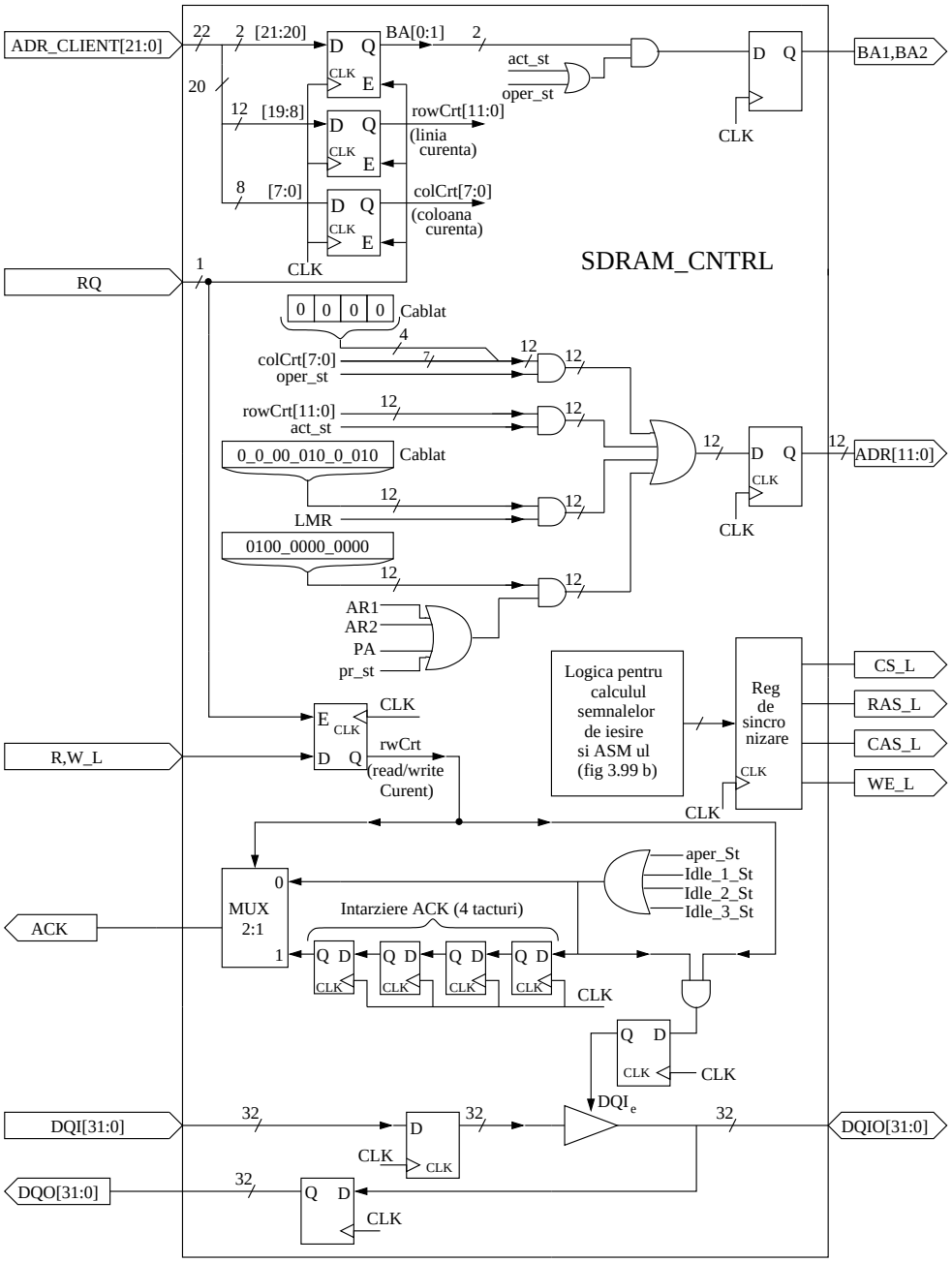


Figura 3.101 Explicativă pentru Exemplul 3.33. Circuistica controllerului pentru generarea semnalelor exterioare (nu s-a figurat pentru semnalele CS_L, RAS_L, CAS_L şi WE_L).

depind numai de stări), care se implementează cu întârziere (pe un registru de sincronizare) pentru ca să se obţină semnale sincronizate pentru SDRAM, Figura 3.99-b. Codificarea stărilor automatului se alege de tipul "one-hot" pentru a obţine o structură simplă de automat (11 bistabile D); cuvintele de cod pentru fiecare stare sunt notate în tabel şi pe organigramă în hexazecimal. Din tabelul de tranziţie al stărilor, pe baza celor cinci semnale de intrare (unul generat de client, RQ, iar celelalte generate de controller), a stărilor prezente, $z_{10}z_{10}\dots z_1z_0$, şi a stărilor următoare, $w_{11}w_{10}\dots w_1w_0$, se poate realiza sinteza circuitului combinaţional al semiautomatului pentru calculul stării următoare.

În tabelul ieşirilor pentru fiecare comandă, realizată într-o stare a automatului, pe baza informaţiilor de catalog din Figura 3.97-a, s-au completat valorile corespunzătoare ale semnalelor generate de automat: CS_L, RAS_L, CAS_L, WE_L, ACK, DQ_L. Sinteza expresiilor logice pentru aceste semnale în funcţie de cuvintele de stare nu ridică probleme, în starea (6) trebuie făcută sinteza şi în funcţie de wr_eft (pentru a distinge între comenzile WRITE şi READ) iar în starea (0) pentru cele trei tipuri de comenzi: PA, AR şi LMR; aceste semnale de condiţionare sunt introduse în blocul combinaţional de după registrul de stare, Figura 3.99-b.

Generarea semnalului de confirmare acceptare (ACK = 1) cerere client (RQ = 1) corespunde situaţiei când cererea respectivă pentru R,W_L este executată, adică automatul este în una din stările (6), (7), (8) sau (9) ceea de se realizează printr-o poartă OR4, Figura 3.101. Dar, semnalul ACK pentru cazul când operaţia cerută a fost READ trebuie întârziat cu patru tacturi, patru bistabile înseriate (2 tacturi pentru compensarea latenţei CAS, 1 tact pentru lateţa bistabilului D de pe ieşirea DQO şi un tact pentru compensarea latenţei semnalului RQ introdus printr-un bistabil D). Alegerea între ACK pentru READ sau WRITE se face cu un MUX2:1 a cărui selectare se obţine prin semnalul compus $RQ \cdot R, W\_L = 1$ (R, W_L se aplică pe intrarea de Enable)..

Cuvântul de 12 biţi, transmis pe portul ADR[11:0] al memoriei, vezi Figura 3.96-a, este calculat de controller în patru variante

1. ca un cuvânt de adresă linie, în starea act_st, şi este identic cu biţi ADR_CLIENT[19:8], vezi Figura 3.98-c;
2. ca un cuvânt de adresă coloană, în starea oper_st, şi este identic cu biţii ADR_CLIENT[7:0] la care se completează cei patru biţi superiori cu cuvântul cablat 0000;
3. ca un cuvânt de programare pentru registrul de mod, în starea Init_st, LMR=1, şi se obţine ca un cuvânt cablat, 0_0_00_010_0_010, vezi Figura 3.96-b;
4. ca un cuvânt pentru comanda PREÎNCĂRCARE, în starea pr_st sau Init_st, şi se obţine ca un cuvânt cablat, 0100_0000_0000, $A_{11} = 1$ preîncărcarea tuturor băncilor.

Datele de la client $DQI[31:0]$, separate de datele înspre client $DQO[31:0]$, sunt multiplexate pentru o singură magistrală bidirecţională între controller şi memorie. Multiplexarea se face în controller prin intermediul unui buffer TSL comandat în stare normală de funcţionare (înscrierea DQI în memorie) prin semnalul DQ_e dacă operaţia curentă este de înscriere (Modelul VERILOG al acestui controller este prezentat în capitolul 5 din volumul II al acestei lucrări).

3.6.3 Circuite actuale pentru memoriile de date

Memoria RAM de capacitate mare este utilizată în primul rând în scopul de a stoca date pentru procesare (funcţie aritmetică) şi nu atât pentru a implementa funcţii

logice. Dar la aplicaţii cu cantităţi mari de date se impune, implicit, şi o modalitate de citire şi scriere a unei cantităţi mari de date (/cuvinte) în unitatea de timp; o primă astfel de modalitate a fost citirea şi înscrierea sub formă de rafală. Există multe modalităţi de creştere a debitului de date pentru lucrul cu memoria, câteva din acestea, care sunt de bază, se vor prezenta în continuare.

Accesarea dublă pe perioada de ceas, DDR (**D**ouble **D**ata **R**ate). La memoria SDRAM accesarea, fie pentru READ, fie pentru WRITE, se realizează (eşantionează/ înregistrează) la un singur cuvânt de date pe o perioadă de ceas (pe frontul pozitiv); dar la memoriile de tip DDR SDRAM pe durata unei perioade de ceas sunt accesate două cuvinte (pe magistrală) care pot fi trimise (înscrise în memorie) sau primite (citite din memorie), adică un cuvânt pe frontul pozitiv şi un cuvânt pe forntul negativ al semnalului de ceas. De fapt, se transmit circuitului DDR SDRAM două semnale de ceas CLK şi negatul acestuia $\overline{\text{CLK}}$ (în opoziţie cu 180°), Figura 3.102; în exteriorul memoriei, pe magistrală, un cuvânt apare pentru CLK iar următorul pentru $\overline{\text{CLK}}$, deci dacă se comandă cu frecvenţle de ceas 100, 133, 166, 290 MHz rata de date este de 200, 266, 333, 400 ... Mcuvinte/s. Frontul pozitiv pentru DDR SDRAM se consideră când pentru CLK există tranziţie L-H, indicat în figură printr-o săgeată, iar pentru $\overline{\text{CLK}}$ când există tranziţia H-L.

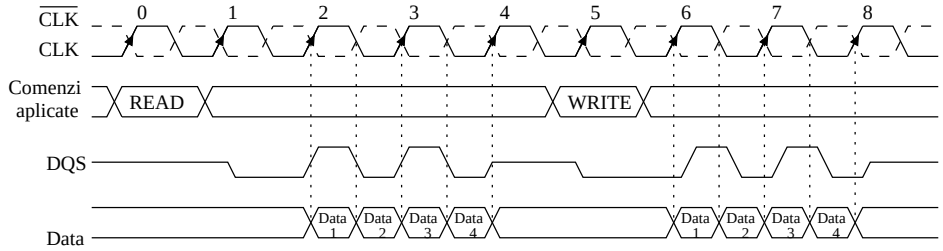


Figura 3.102 Semnalele specifice pentru memoria cu rata dublă de accesare DDR SDRAM. Sincronizarea datelor (strobarea) pentru înscriere sau citire se face cu semnalul DQS.

Pentru a se trece de la o structurare de memorie SDRAM, Figura 3.96, la o structurare de memorie DDR SDRAM se modifică doar partea de interfaţare I/O date (încadrată printr-o linie punctată în figură) cealaltă parte (nucleul memoriei) rămâne nemodificată.

La organizarea de tip DDR SDRAM, pentru o comandă READ/WRITE, pe o perioadă de ceas se accesează un cuvânt de $2n$ biţi la nucleul memoriei, dar înspre/de la pinii I/O sunt două cuvinte de n biţi (câte unul pe fiecare semiperioadă de ceas). Pentru un transfer sigur al datelor de la sursă la recepţie, la o rată dublă pe linia magistrală, la DDR SDRAM sunt introduse semnale (de strob) de sincronizare a datelor, DQS. Un semnal DQS este generat de controller (propagat către memorii) pentru a se realiza înscrierea corectă în memorie a datelor, DQ_{in}, iar un altul DQ_{out} (propagat către controller) este generat de către memorie pentru a asigura recepţia corectă (în timp) la controller a datelor citite din memorie. Strobarea datelor citite (la controller) se face pe frontul semnalului DQS, iar la înscriere (în memorie) se face pe mijlocul palierului semnalului DQS (când nu există transfer pe magistrală,

linia pentru semnalul DQS este în starea HZ, reprezentată la nivelul median între H şi L). Circuitele DDR în raport cu cele SDRAM, deoarece au mai multe funcţiuni care trebuie programate, prezintă un al doile registru de mod, EMR (Extended Mode Register) care se programează într-o modalitate similară cu primul registru de mod.

DDR2 SDRAM este o variantă îmbunătăţită a DDR în tendinţa de creştere a ratei de transfer pe magistrală la valori de 400, 533 cu posibilitate până la 667 sau chiar 800 M cuvinte/s. Această creştere a ratei fluxului de date se poate obţine doar printr-o adaptare a liniei de conexiune între controller şi cipul DDR2 SDRAM, în scopul îmbunătăţiri semnalelor de date, DQ, de strobare, DQS, de mascare, DQM atât pentru înscriere cât şi pentru citire. Adaptarea la intrarea pe circuitul DDR2, Figura 3.103-a, se face printr-un divizor $2Z_0$ la V_{DD} şi $2Z_0$ la V_{SS} (terminator Thevenin, vezi Figura 1.75-b), a cărui rezistenţă echivalentă este egală cu impedanţa caracteristică a liniei Z_0. Printr-un pin suplimentar ODT (On-Die Termination) pe cipul DDR2 se aplică un semnal de la controller care, pe baza (şi a programării) registrului de mod extins, EMR, poate realiza comanda, prin comutatoarele SW_1 şi SW_2, de conectarea sau deconectarea terminatorului Thevenin (sunt două divizoare cu două valori de rezistenţe, $2Z_{01}$, $2Z_{02}$ pentru a realiza adaptarea la două valori de impedanţe caracteristice Z_{01} şi Z_{02}).

Pentru operaţia de citire când datele se transmit de la memorie la controller, Figura 3.103-b, este prezentată adaptarea în controller atât pentru cazul când există un singur modul de memorie activ (din care se citeşte) cât şi pentru cazul când sunt două module de memorie conectate. Când sunt două module de memorie conectate, adaptarea se realizează şi la al doilea modul de memorie (din care nu se citeşte, este în aşteptare). Adaptarea pe ieşirea driverului pentru datele DQ_{out} de pe modulul de memorie (activ), constă în egalizarea împedanţei de ieşire în starea H cu cea din starea L la valoarea de $18 \pm 1.5\Omega$. În acest scop în interiorul controllerului, pe o rezistenţă etalon, se măsoară căderea de tensiune atât pentru ieşirea în starea H cât şi pentru ieşirea în starea L a driverului de ieşire din memorie. Pe baza acestor două tensiunii măsurate controllerul generează o succesiune de comenzi WRITE prin care se ajustează, pentru egalizare, la driverul de ieşire de la memorie, reţeaua de rezistenţe de ieşire din starea H cu reţeaua de rezistanţe de ieşire în starea L.

În Figura 3.103-c este prezentată adaptarea pentru înscrierea în memorie atât pentru un singur modul cât şi pentru două module de memorie conectate. Responsabilitatea generării succesiunii de comenzi pe pinul ODT de la DDR2 SDRAM revine controllerului.

QDR SRAM. Modalitatea QDR aplicabilă la memoriile statice RAM este o extensie a modalităţii DDR în sensul că, pe lângă accesarea cu o rată dublă pe perioada de ceas, de data aceasta memoria este accesată DDR la două porturi (memorie dublu port); deci în raport cu o memorie SRAM pe magistrală se obţine un flux de date cu o rată qvadruplă, de unde şi denumirea **QDR** (**Q**uad **D**ata **R**ate). Memoria QDR SRAM, a cărei structurare de principiu este prezentată în Figura 3.104-a, prezintă două porturi cu funcţionare independentă, unul de intrare (înscriere) şi unul de ieşire (citire), fiecare din acestea fiind accesate de două ori pe perioada de ceas (CLK, $\overline{\text{CLK}}$). Porturile fiind accesate simultan, cuvintele ADRESĂ pentru înscriere sau citire sunt multiplexate pe magistrala de adresare, memoria apare ca realizând un flux de date în acelaşi sens.

Memoria QDR SRAM a fost realizată pentru aplicaţiile unde succesiunea între

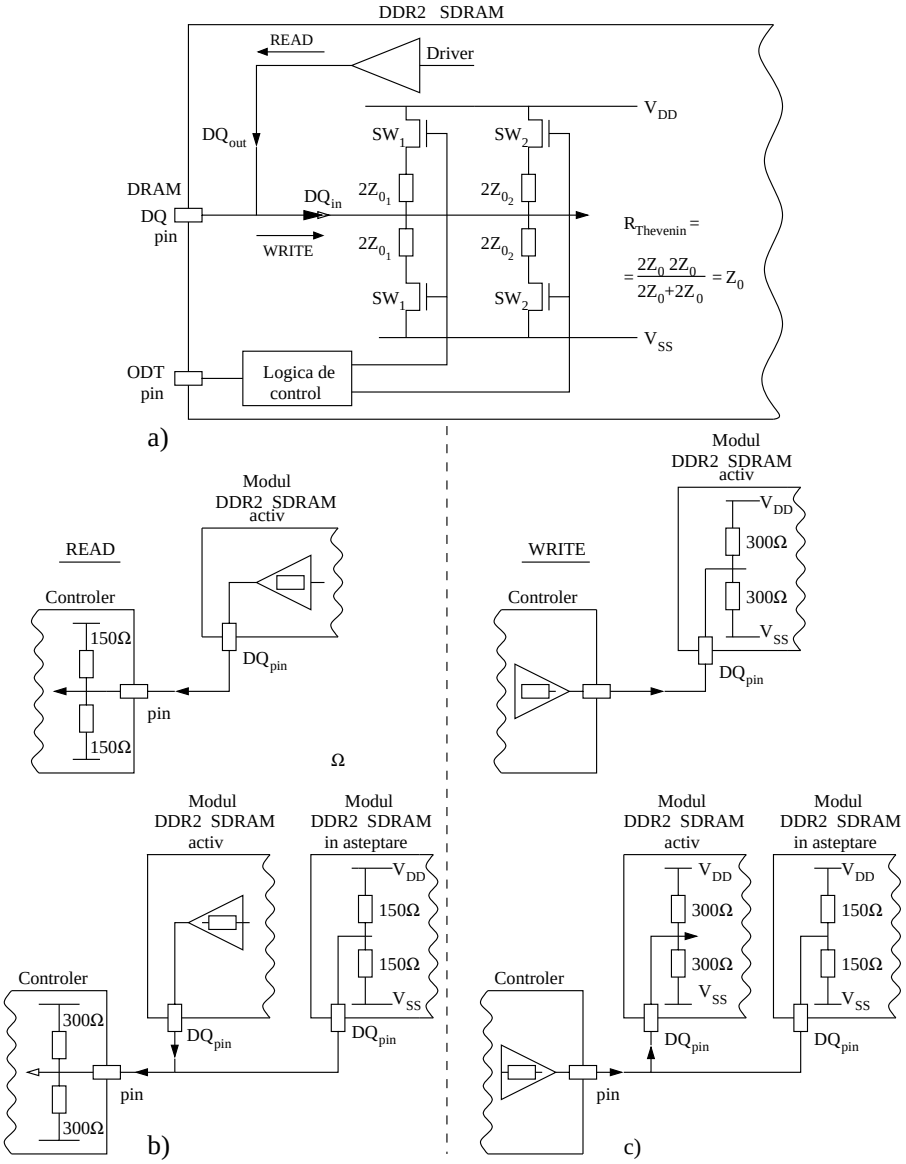


Figura 3.103 Memoria cu rată dublă de accesare DDR2 SDRAM: a) structura (doar a) circuitului pentru adaptarea intrărilor pe cipul de memorie; exemplificare de adaptare între controller şi memorie pentru operaţia de citire (b) şi pentru operaţia de înscriere (c)

operaţiile de citire, RD, şi cele de înscriere, WR, este foarte strânsă în timp, de exemplu alternanţele RD-WR-RD-WR-..., RD-RD-WR-RD-RD-WR-..., RD-WR-WR-RD-WR-WR-...; adică acele aplicaţii la care raportul mediu între numerele acestor două operaţii efectuate nu depăşeşte valoarea doi sau trei. Aplicaţiile unde valoarea acestui raport mediu este mai mare de trei, deci un şir continuu de date de înscriere sau de citire, sunt bine acoperite de către o memorie DDR. Memoria QDR SRAM, Figura 3.104-b, permite două tipuri de implementare: cu lungime de rafală 2 şi cu lungime de rafală 4, iar pentru fiecare din acestea, în funcţionare trebuie considerate două aspecte, accesarea adresei pe magistrala de adrese şi plasarea (captarea) datelor pentru înscriere.

Memoria QDR SRAM cu **lungimea de rafală 2** poate susţine indefinit o succesiune de comenzi externe READ (2 cuvinte DQ_{in}), WRITE (2 cuvinte DQ_{out}) în fiecare ciclu de ceas; intern, prima jumătate a perioadei de ceas realizează funcţia READ iar în a doua jumătate funcţia WRITE. Evident, pe magistrala de adrese se aplică succesiv adresa pentru citire urmată de cea da înscriere; frontul pozitiv al semnalului CLK înregistrează adresa de citire în portul de ieşire (PC) iar frontul pozitiv al semnalului $\overline{\text{CLK}}$ înregistrază adresa de înscriere în portul de intrare (PI).

Memoria QDR SRAM cu **lungimea de rafală 4** poate susţine indefinit numai pentru o alternanţă de comenzi externe READ (4 cuvinte DQ_{out}), WRITE (4 cuvinte DQ_{in}) pe durata de câte două cicluri de ceas, la fel ca şi memoria cu lungimea de rafală 2, dar intern utilizează ciclurile diferit faţă de organizarea cu lungime de rafală 2. Memoria SRAM utilizează un ciclu de ceas pentru realizarea internă a comenzii READ apoi pe următoarele două cicluri de ceas rezultă la ieşirea portului de citire patru cuvinte DQ_{out}. Pe următorul ciclu de ceas, după cel ocupat de realizarea comenzii READ, o comandă WRITE poate fi iniţiată care determină înscrierea în portul de intrare a patru cuvinte date, DQ_{in}. Rezultă că succesiunea celor două comenzi implică numai aplicarea unei adrese pe ciclu; frontul pozitiv al semanlului CLK înregistrează adresa de citire iar următorul front pozitiv, al semanlului CLK, este disponibil pentru înregistrarea adresei de înscriere. Dacă nu este aplicată o comandă WRITE pe următorul front pozitiv al semnalului de cesa CLK atunci pe acest front nu poate fi iniţiat nici un alt ciclu RD-WR; rafala cu lungimea 4 nu poate fi stopată mai devreme, aceasta trebuie să se termine.

Pentru sistemele care permit lucrul cu rafale de lungime 4 (şi nu numai lucrul cu rafale cu lungime 2) se recomandă primul tip de organizare de QDR (lungime 4) pentru că: 1 - adresele pentru comenzile de READ şi WRITE sunt prezentate de către controller doar câte una pe un ciclu de ceas; 2 - la aceeaşi viteză realizabilă pe siliciu se poate comanda cu o frecvenţă de ceas mai ridicată.

Referitor la plasarea datei de înscriere, DQ_{in}, există o mică diferenţa în funcţionarea memoriei QDR cu rafală de lungime 2 şi rafală de lungime 4, ceea ce se poate observa din diagrama de semnale din figură. Pentru rafala de lungime 2 captarea datelor de înscriere DQ_{in1} se realizează imediat ce semnalul de înscriere devine activ, WR_L=0, şi pe frontul pozitiv al semnalului CLK, iar pe următorul front pozitiv al semnalului $\overline{CLK}$ este captat cuvântul DQ_{in2} şi interior este executată operaţia de înscriere. Pentru rafala de lungime 4 captarea datelor de înscriere DQ_{in1} se realizează cu un ciclu de ceas întâziere după activarea semnalului de înscriere, WR_L=0; deci la următorul front pozitiv CLK este captat DQ_{in1}, la următorul front pozitiv $\overline{CLK}$ este captat DQ_{in2}, urmează apoi captarea pentru DQ_{in3}, DQ_{in4}, respectiv pe fronturile

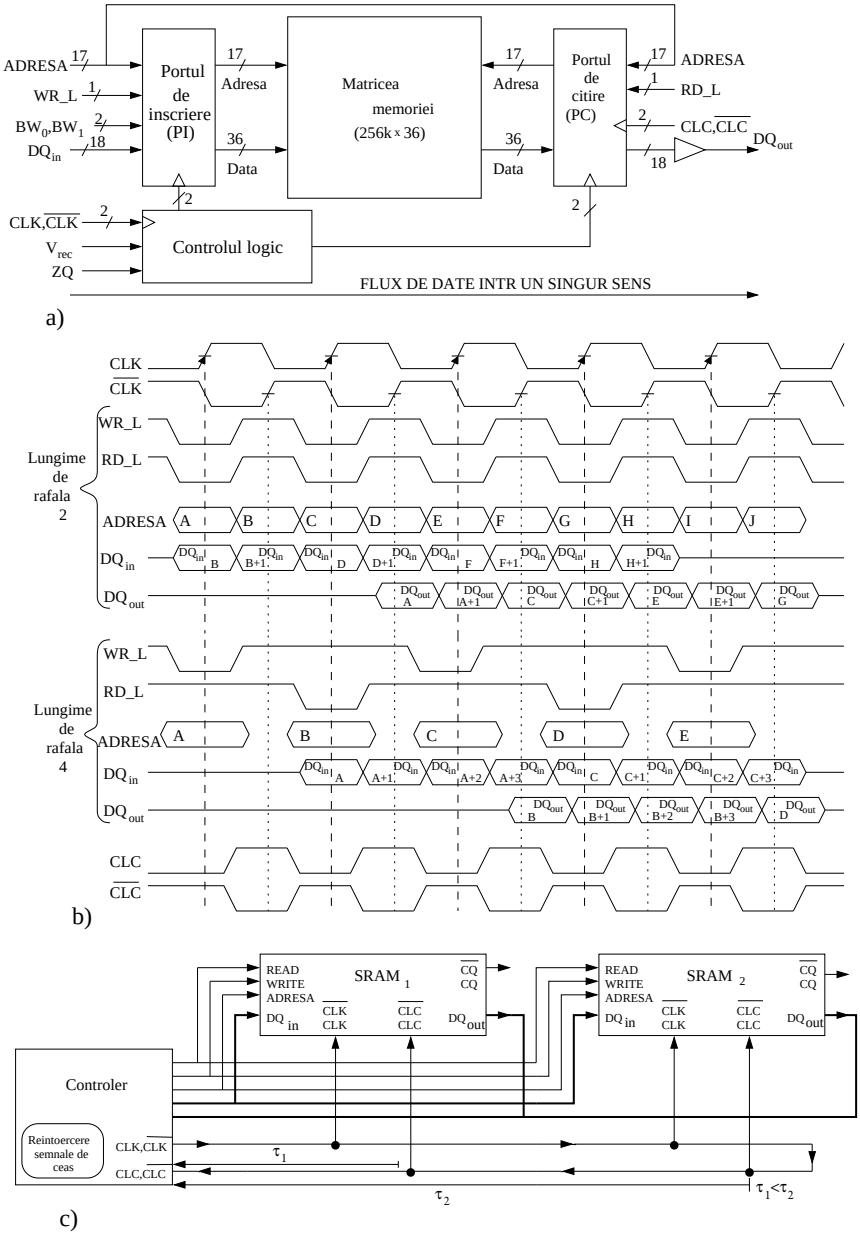


Figura 3.104 Memoria de tip QDR SRAM: a) structurarea de principiu a memoriei (prin caracteristica de dublu port se realizeaza un singur sens pentru fluxul de date); b) diagramele de semnale pentru comenzile READ şi WRITE pentru QDR cu lungimea de rafală 2 şi 4; c) organizarea aplicarii semnalelor de la/înspre controller pentru sincronizare când memoria este realizata din mai multe module.

pozitive ale semnalelor CLK şi $\overline{CLK}$.

Semnalele de ceas pentru memoriile QDS SRAM trebuie să asigure restricţiile de temporizare. Pentru înscriere controllerul, Figura 3.104-c, trebuie să genereze memoriei semnalele de ADERSĂ, WR_L, DQ_{in} cu cel puţin τ_{SU} înainte de aplicarea frontului pozitiv al semnalului CLK şi evident menţinerea validă a acestora pe un interval $\tau_{\rm H}$ după acest front. De asemenea, pentru citire cu aceleaşi restricţii de temporizare trebuie eşantionate datele DQ_{out}, ADRESĂ şi RD_L în portul de ieşire, mai mult aceste date DQ_{out} trebuie înscrise corect în controller după ce au fost transferate pe distanţa dintre memorie şi controller. Pentru memoriile QDS SRAM care sunt comandate la frecvenţe sub 133 MHz semnalele CLK şi $\overline{CLK}$ pot fi generate încât să realizeze restricţiile de temporizare, dar la memoriile QDR care lucrează la frecvenţe ridicate, peste 133 MHz, timpii de acces devin sensibil egali cu timpii de propagare a semnalelor pe distanţa de la controller la primul modul de memorie.

Pentru înscrierea datelor , când sunt mai multe module QDR, semnalele CLK, $\overline{CLK}$ se generează la controller cu o mică întârziere faţă de generarea semnalelor ADRESĂ, WR_L, DQ_{in} încât să se asigure $\tau_{\rm SU}$; iar $\tau_{\rm SU}$ este asigurat la fiecare modul deoarece se presupune că pentru fiecare modul toate aceste semnale generate de controller parcurg împreună aceeaşi distanţă (deci aceeaşi întârziere de propagare). Dar la citire pentru că datele DQ_{out} parcurg de la module până la controller distanţe diferite (de exemplu, în figură pentru cele două module întârzierile de propagare până la controller sunt τ_1 şi τ_2, $\tau_1 < \tau_2$) pentru ca să fie captate în controller cu acelaşi front de ceas aceste date trebuie să fi fost înregistrate în porturile de ieşire de la memorie anterior cu intervale de timp τ_1 şi τ_2. Rezultă că, pe lângă semnalele CLK şi $\overline{CLK}$ pentru inscriere, este necesară încă o pereche de semnale de ceas CLC şi $\overline{CLC}$ pentru citire. Aceste semnale CLC şi $\overline{CLC}$, defazate faţă de CLK şi $\overline{CLK}$ se obţin printr-o buclă a traseului semnalelor de ceas, acest traseu porneşte de la controller şi se întoarce tot la controller. Semnalele CLK şi $\overline{CLK}$ generate de controller sunt aplicate succesiv la fiecare modul QDR şi sunt folosite pentru sincronizarea comenzii WRITE, apoi la reîntoarcere, cu defazajele corespunzătoare diatanţelor parcurse, sunt aplicate succesiv dar în ordine inversă la fiecare modul ca semnale CLC $\overline{CLC}$ pentru înscrierea datelor DQ_{out} în porturile de citire; iar când ajung în controller vor sincroniza înscrierea datelor recepţionate. Deoarece prin acestă buclă semnalele de ceas pot avea o încărcare mare, mai nou, memoriile QDR generează ele însele o pereche de semnale ecou, CQ şi $\overline{CQ}$, care sunt utilizate pentru sincronizarea în controller a datelor citite DQ_{out}; aceste semnale de ceas ecou sunt produse din semnalele de ceas primite de memorie, CLK şi $\overline{CLK}$, prin introducerea unor defazaje. (În structura de principiu a memoriei QDR din Figura 3.104-a, semnalul ZQ are o funcţie similară cu a semnalului aplicat pe pinul ODT la o memorie DDR2, adică ajustarea impedanţei de ieşire a driverelor de ieşire pentru a se adapta la impedanţele de pe circuitul imprimat. BW_1, BW_0 sunt semnale prin care se poate selecta pentru înscriere un anume byte din cuvântul DQ_{in}.)

Pentru circuitele de memorie, utilizate în sistemele de de calcul, tendinţa este ca acestea să devină din ce în ce mai mult circuite programate (prin unul sau mai multe Load Mod Register) şi, în viitor, memoria să dobândească o anumită inteligenţă astfel încât (integrând şi controllerul) să poată realiza independent anumite funcţii.

3.6.4 Memoria adresabilă prin conţinut, CAM

La toate tipurile de memorie prezentate până acum procesul de obţinere/restablire a unei informaţii (cuvânt) se realizează în felul următor: prin cunoaşterea unei adrese (deci se impune memorarea, într-un fel, a adresei) care se aplică memoriei se extrage din locaţia, de la această adresă, informaţia căutată. Dar, se poate realiza acest proces şi în sens invers, adică se cunoaşte informaţia (cuvântul de date) şi se determină adresa (locaţiei) unde este stocat (acest cuvânt) în memorie. Practic, acest proces invers se realizează în felul următor: cuvântul data cunoscut, $D_{n-1}D_{n-2}\dots D_1D_0$, se compară, în paralel, cu cuvintele data din toate locaţiile memoriei şi se generează un semnal de găsire M (match) dacă există stocat acest cuvânt şi totodată se obţine şi adresa (sau adresele), AM_{m-1}, …, AM_k, … AM_0, locaţiei care conţin acest cuvânt, Figura 3.105-a. Deoarece la acest tip de memorie explorarea se realizează prin conţinut, şi nu prin adresă, referirea sa se face prin sintagma memorie adresabilă prin conţinut, **CAM** (**C**ontent **A**ddresable **M**emory).

Pentru structurarea de principiu a unei memorii CMOS CAM, Figura 3.105-b, se poate porni de la circuitul pentru generarea identităţii a două cuvinte, realizat cu tranzistoare de trecere, Figura 1.53-d. Câte un astfel de circuit de generare a identităţii este realizat pentru toate celulele, Figura 3.91-a, unei locaţii (adresă) a memoriei. Poarta XOR pentru celula i este relizată cu tranzistoarele de trecere T_1 şi T_2 (şi realizează $x_i\overline{D_i} + \overline{x_i}D_i$). Cele două valori x_i şi $\overline{x_i}$ se găsesc înscrise în celula i, ca ieşiri ale celor două inversoare CMOS (în celulă este înscris bitul x_i al cuvântului $X = x_{n-1}\dots x_i\dots x_1x_0$), iar cele două valori D_i şi D_i, ale bitului de rang i din cuvântul de comparat, $D = D_{n-1}\dots D_i\dots D_1D_0$, se aplică din exteriorul memoriei respectiv pe coloanele de bit C_1 şi C_2 la celulele i de pe toate liniile de cuvânt (adrese). (Celula de memorie CMOS se consideră în starea 0 ($x_i = 0$, $\overline{x_i} = 1$) când ieşirea inversorului din dreapta este H şi ieşirea inversorului din stânga este L iar în starea 1 când cel din stânga este în H şi cel din dreapta este în L.) Circuitul NOR al generatorului de identitate, între cuvântul D şi cuvântul X de pe o linie de cuvânt, este constituit (distribuit) din toate tranzistoarele T_3, câte unul pentru fiecare celulă de memorie, şi care sunt comandate de către ieşirile porţilor XOR. Această poartă NOR distribuită poate fi realizată dinamic (cu preîncărcarea la timpul potrivit, de exemplu când M=1) sau ca o pseudo poartă CMOS (ca în structura prezentată, τ_s este tranzistorul de sarcină) dacă nu este critică cerinţa de viteză.

Memoria CMOS CAM cu structurarea prezentată poate fi utilizată în modul normal pentru înscriere şi citire; operaţia de înscriere este necesară pentru a stoca cuvintele data X la oricare adresă, iar citirea numai pentru utilităţi de testare; în Figura 3.105-c este prezentată o astfel de memorie cu patru adrese pentru cuvinte cu lungimea de patru biţi. Pentru funcţionare ca memorie CAM cuvântul căutat D se aplică pe pinii de intrare, ca dată de intrare, astfel că pe coloanele de bit C_1 şi C_2 vor fi respectiv valorile D_i şi $\overline{D_i}$, dar nu se aplică memoriei nici un cuvânt de adresă. Dacă bitul D_i este identic cu bitul x_i din celula i atunci poarta XOR corespunzătoare va aplica 0 pe poarta tranzistorului T_3 (blocându-l) din poarta NOR a generatorului de identitate; toate tranzistoarele T_3 de pe o linie de cuvânt (de la o poartă NOR) vor fi blocate numai când $X \equiv D$, indicând la ieşire, prin acţionarea $AM_k = 1$, adresa (liniei) la care s-a găsit cuvântul căutat. Dacă cel puţin un bit din cele două cuvinte sunt diferite, tranzistorul T_3 corespunzător acelui bit va conduce, poarta NOR de pe linia respectivă produce 0 pe ieşire, $AM_k = 0$, indicând neidentitatea cuvin-

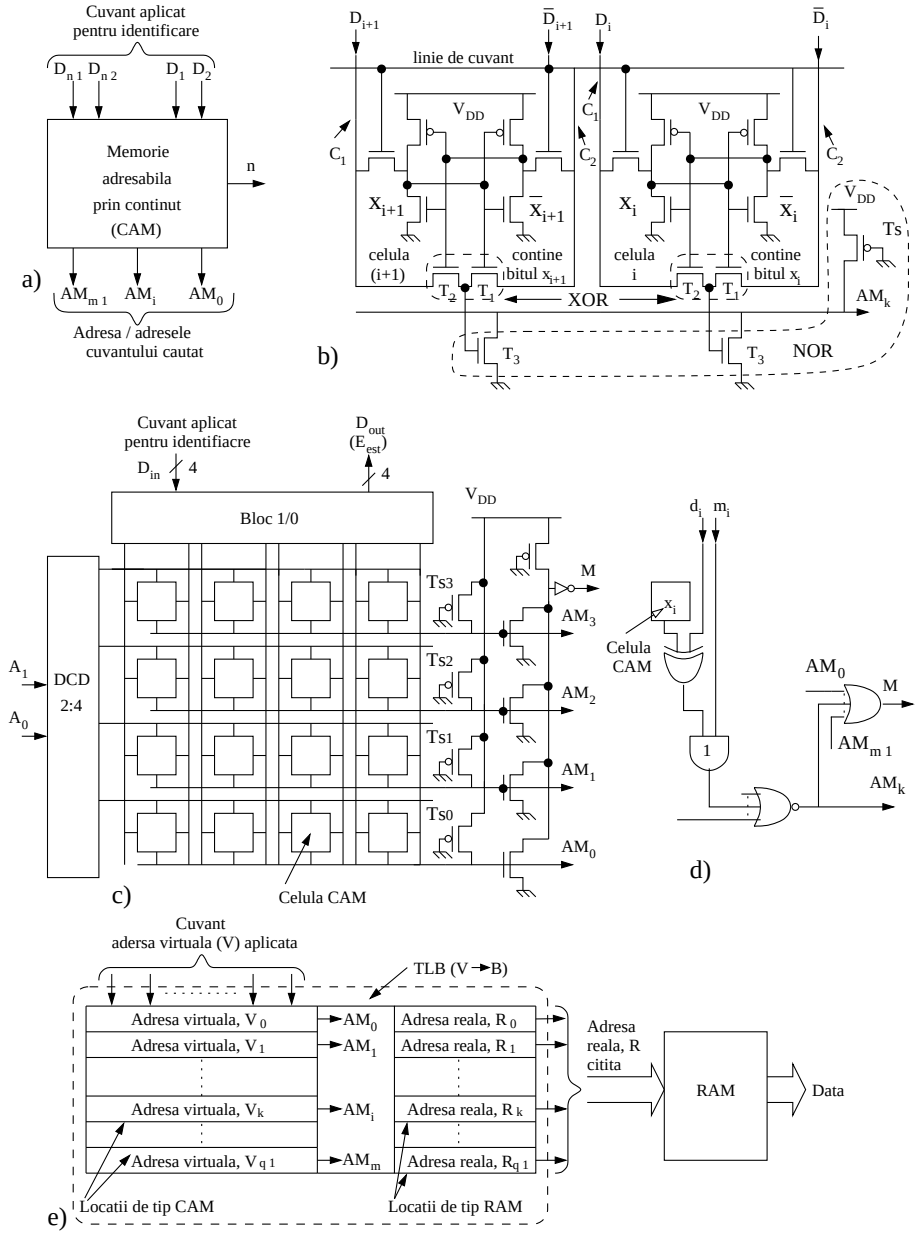


Figura 3.105 Memoria adresabilă prin conţinut, CAM: a) schemă bloc pentru o memorie CAM; b) structurarea unei celule de memorie CAM pe baza unei celule RAM-CMOS prin adăugarea componentei de generare a funcţiei de identitate (poarta XOR); c) organizarea unei memorii CAM de patru locaţii pentru cuvinte de patru biţi; d) structura logică a unei celule CAM pentru căutarea cuvintelor incomplet definite; e) structurare de principiu, pe bază de memorie CAM, a unui circuit pentru translatarea adreselor virtuale în adese reale (TLB).

telor, $X \not\equiv D$. În acest mod de structurare al memoriei CAM adresa la care s-a găsit cuvântul căutat corespunde cu numărul poziţiei în cuvântul $AM_{m-1} \dots AM_k \dots AM_0$, pentru care bitul are valoarea 1. În cazul în care există mai multe adrese la care s-a găsit cuvântul căutat se poate alege doar una din aceste adrese prin aplicarea cuvântului $AM_m \dots AM_k \dots AM_0$ la intrarea unui codificator prioritar $m : \lceil log_2 m \rceil$ (alegerea uneia din adresele găsite depinde de modul de alocare a priorităţilor, vezi 2.4.2). Pe lângă aflarea adresei locaţiei, unde există identitate, $AM_k = 1$, structura de memorie CAM trebuie înzestrată şi cu generarea unui semnal de găsire, $M = 1$, ceea ce se poate realiza cu o poartă NOR, ale cărei tranzistoare în paralel din reţeaua nMOS sunt comandate de ieşirile AM_k, $0 \leq k \leq m-1$; ieşirea negată din acestă poartă generează $M = 1$ când cuvântul căutat a fost găsit la cel puţin o adresă.

Uneori, este necesar a se căuta în memorie un cuvânt D_k care este incomplet cunoscut, adică valorile anumitor biţi ai săi nu se cunosc. La accesarea printr-un cuvânt incomplet definit structura celulei (pentru un bit) a generatorului de identitate este prezentată în Figura 3.105-d. Structura acestei celule rezultă din strucura celulei anterioare la care se adaugă poarta 1 care este comandată de bitul de mască m_i; dacă D_i este definit atunci m_i trebuie să fie 1 (poarta AND2 este deschisă), iar dacă D_i nu se cunoaşte atunci bitul de mască trebuie să producă un rezultat ca şi când biţii D_i şi x_i au valori identice, deci $m_i = 0$ (poarta AND2 este închisă). Cu această regulă rezultă configuraţia cuvântului mască, m, care are aceeaşi lungime ca şi a cuvântului căutat, cu maparea: pentru biţii D_i cunoscuţi $\rightarrow m_i = 1$, iar pentru biţii D_i nedefiniţi $\rightarrow m_i = 0$. Memoriei CAM i se aplică din exterior simultan două cuvinte, cuvântul dată căutat, D_k, şi cuvântul mască m. În exemplul următor, pentru un cuvânt incomplet definit D_k, se realizează cuvântul mască m şi se prezintă căutarea la două adrese din care la una cu succes (a) iar al cealaltă (b) fără succes.

$D_k =$	–	1	0	–	1	–	1	0	(Data căutată)
$X =$	0	1	0	0	1	1	1	0	(Data existentă în CAM)
$m =$	0	1	1	0	1	0	1	1	(cuvântul mască)
Rezultat *a)*	0	0	0	0	0	0	0	0	$\rightarrow AM_k = 1$

$D_k =$	–	1	0	–	1	–	1	0	(Data căutată)
$X =$	1	0	0	1	1	0	1	0	(Data existentă în CAM)
$m =$	0	1	1	0	1	0	1	1	(cuvântul mască)
Rezultat *b)*	0	1	0	0	0	0	0	0	$\rightarrow AM_k = 0$

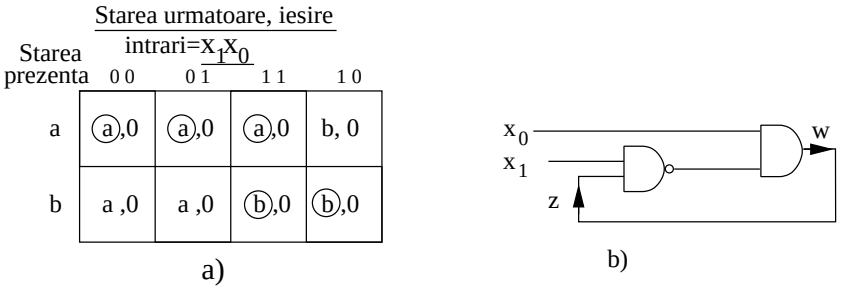
O utilizare foarte frecventă a memoriei CAM este în cadrul sistemelor pe bază de microprocesor, pentru translatarea adresei virtuale în adresă reală prin aşanumitul circuit **TLB** (Translation Lookaside Buffer). Într-un sistem de calcul spaţiul de adresare virtual, V, cu care lucrează programele este mult mai mare decât spaţiul de adresare real (/fizic), R, cu care lucrează procesorul, adică numărul de adrese (fizice) din modulele de memorie ale sistemului. (De exemplu, să considerăm că spaţiul virtual ar fi

de 2^{32} adrese (4G), adresare cu un cuvânt de 32 biţi, iar spaţiul real de 2^{20} adrese (1M), adresare cu un cuvânt de 20 biţi). Translatarea unei adrese virtuale intr-o adresă reală, V→R, se realizează cu un tabel, TBL, implementat cu o memorie CAM, Figura 3.105-e.

Memoria CAM cu un număr de q locaţii are pe fiecare linie stocate două cuvinte; în prima jumătate a liniei cu structură CAM este un cuvânt care reprezintă o adresă virtuală iar în a doua jumătate a liniei (cu strutură de memorie RAM) este un cuvânt care reprezintă o adresă reală (lungimea cuvântului de adresă virtuală este mai mare decât a celui de adresă reală). În tabel se înscriu în prima jumătate, în funcţie de numărul de locaţii q, doar acele adrese virtuale care se preconizează că vor fi referite prin rularea programului, iar pe fiecare linie din a doua jumătate se înscrie adresa (reală) locaţiei din memoria RAM a sistemului unde se află data referită de adresa virtuală (inscrisă în prima jumătate a aceleiaşi linii). La aplicarea unui cuvânt de adresă virtuală la jumătatea de tip CAM a tabelului, acest cuvânt este comparat cu toate adresele virtuale înscrise şi în cazul unei identităţi se generează pentru acea linie semnalul $AM_k = 1$. Semnalul AM_k activat va produce citirea adresei reale de pe a doua jumătate a liniei respective, iar această adresă reală este aplicată la memoria RAM a sistemului de unde se citeşte/înscrie data referită de adresa virtuală.

PROBLEME

P3.1 Să se deducă structura automatului asincron care are următorul tabel de evoluţie al stărilor (Figura a):



P3.2 Să se studieze condiţia de instabilitate pentru circuitul a cărui structură este prezentată mai sus (Figura b din textul problemei P3.1)

P3.3 Să se determine stările totale stabile ale circuitului asincron obţinut prin conectarea ieşirilor $\overline{F}_1$ şi $\overline{F}_2$ ale unei porţi DAR-NEGAT respectiv la intrările $\overline{A}_2$ şi $\overline{A}_1$. Structura porţii DAR este dată în problema P2.22.

P3.4 Un automat detectează dacă într-un şir de biţi aplicat pe intrarea x există consecutiv trei sau mai mulţi biţi cu valoarea 1, dacă da se generează ieşirea $y = 1$. Să se deseneze graful de tranziţie al stărilor/ieşirii precum şi tabelul de tranziţie al stărilor/ieşirii.

P3.5 Un automat detectează dacă într-un şir de biţi aplicat pe intrarea x exită în ultimi trei biţi aplicaţi exact doi biţi cu valoarea 1, dacă da se generează ieşirea

$y = 1$. De exemplu pentru secvenţa de intrare 011011100 se generează pe ieşire secvenţa 001111010. Să se deseneze graful de tranziţie al stărilor/ieşirilor precum şi tabelul de tranziţie al stărilor/ieşirii.

P3.6 Un automat calculează funcţia majoritar de trei variabile $F = \sum_0^7(3, 5, 6, 7)$ dar tripletul valorilor celor trei variabile este aplicat serial pe intrarea x a automatului. Automatul generează $y = 0$ până când a treia valoare din triplet este aplicată pe intrare iar atunci va genera $y = 0$ sau 1 daca numărul de biţi 0 respectiv de 1 a fost majoritar în triplet; de exemplu pentru şirul aplicat pe intrare 011100101 va genera la ieşire 001000001. Să se deseneze graful de tranziţie al stărilor/ieşirii precum şi tabelul de tranziţie al stărilor/ieşirii.

P3.7 Un automat, de tip Mealy, identifică dintr-un şir de biţi, aplicat la intrarea x, secvenţa 10110 şi generează, când această secvenţă este identificată, unu logic pe ieşirea y. În şirul aplicat peste secvenţa corectă se poate suprapune, parţial, şi o alta următoare, secvenţa corectă. Să se deseneze graful de tranziţie al stărilor/ieşirii precum şi tabelul de tranziţie al stărilor/ieşirii.

P3.8 Să se realizeze graful de tranziţie al stărilor/ieşirii pentru un automat Mealy care are funcţiunea de detector de secvenţe: dintr-un şir oarecare de biţi aplicat la intrarea x se detectează secvenţa 010110 şi se generează, cand această secvenţa este identificată, unu logic pe ieşirea y; peste secvenţa corectă se poate suprapune, parţial, şi o altă următoare secvenţa corectă.

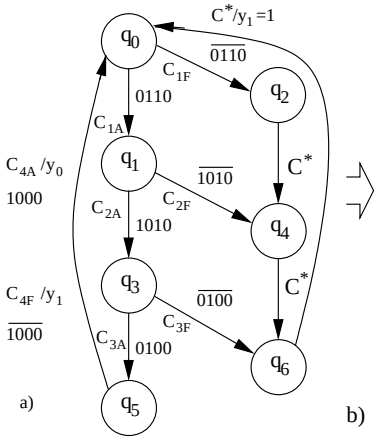
P3.9 Un detector de secvenţe, de tip automat Mealy, când detectează secvenţa 01101 dintr-un şir de biţi aplicat pe intrarea x generează 1 pe ieşirea y_1, iar când detectează secvenţa 01111 generează 1 pe ieşirea y_0. Unei secvenţe din şir i se poate suprapune o altă secvenţă de acelaşi tip sau din celălalt tip. (cuvântul de ieşire este ordonat y_1y_0).

P3.10 Un automat de tip Moore detectează dacă secvenţe de câte patru biţi aplicate pe intrarea x constituie un cod corect pentru o cifră exprimată în codul BCD (8-4-2-1). Dacă secvenţa este un cod BCD corect se generează $y = 1$, dacă nu se generează $y = 0$. Să se realizeze graful de tranziţie al stărilor.

P3.11 Un automat Mealy care detectează, într-un şir de biţi aplicat pe intrarea x, secvenţa 01010 are următoarea funcţionare. Când secvenţa 01010 este detectată în şir se generează ieşirea $y_0 = 0$. Dacă în şirul de intrare se detectează subsecvenţa 011 atunci se generează $y_1 = 1$ şi se iniţializează, din nou, căutarea secvenţei 01010. Peste o secvenţă 01010 se poate suprapune începerea unei următoare secvenţe 01010. Să se realizeze graful de tranziţie al stărilor/ieşirilor.

P3.12 Un automat Mealy cu o singură intrare x şi două ieşiri y_0 (pentru deschidere) şi y_1 (pentru alarmă), utilizat ca cheie electronică, are următoarea funcţionare. Pentru deschidere trebuie să se aplice pe intrarea x succesiv patru cuvinte de cod C_1, C_2, C_3 şi C_4 (fiecare de câte patru biţi), de exemplu $C_1 = 0110 = 6|_{10}$, $C_2 = 1010 = 10|_{10}$, $C_3 = 0100 = 4|_{10}$ şi $C_4 = 1000 = 8|_{10}$, după care se comandă deschiderea $y_0 = 1$. Dimpotrivă se comandă alarma, $y_1 = 1$, după patru cuvinte de cod dacă acestea sau printre acestea există coduri greşite. Un cod corect se notează cu C_{iA} iar un cod greşit cu $C_{iF}, i = 1, 2, 3, 4$ iar prin C^* oricare combinaţie de patru biţi care generează o tranziţie.

Să se construiască graful de tranziţie al stărilor/ieşirilor şi tabelul de tranziţie al stărilor/ieşirilor.

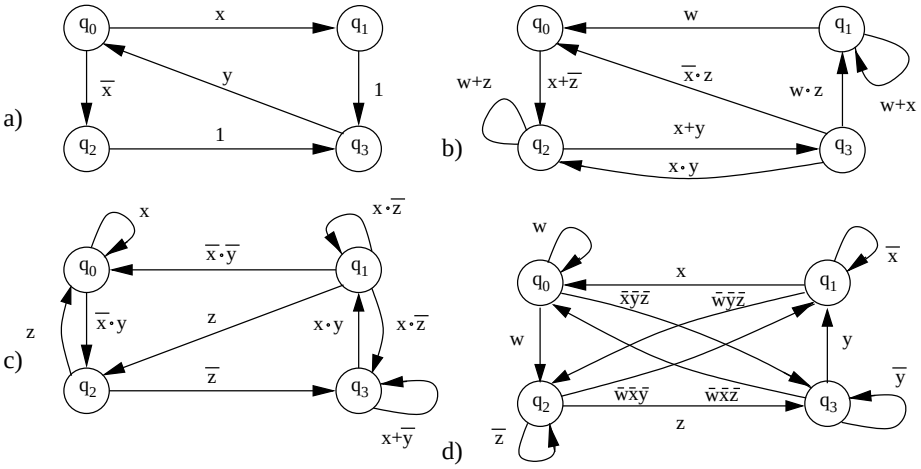


b)

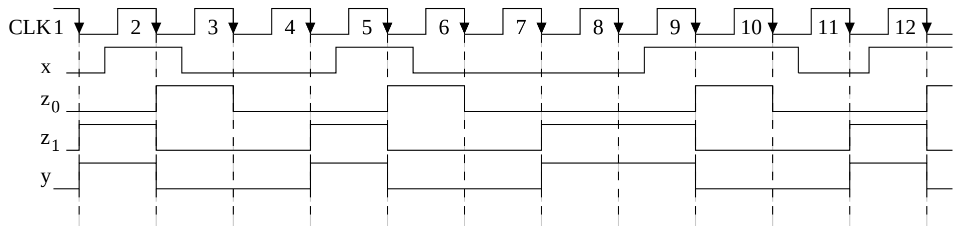
Starea prezenta	C_1		C_2		C_3		C_4	
	C_{1A}	C_{1F}	C_{2A}	C_{2F}	C_{3A}	C_{3F}	C_{4A}	C_{4F}
q_0	q_1	q_2	q_2	q_2	q_2	q_2	q_2	q_2
q_1	q_4	q_4	q_3	q_4	q_4	q_4	q_4	q_4
q_2	q_4	q_4	q_4	q_4	q_4	q_4	q_4	q_4
q_3	q_6	q_6	q_6	q_6	q_5	q_6	q_6	q_6
q_4	q_6	q_6	q_6	q_6	q_6	q_6	q_6	q_6
q_5	q_0/y_1	q_0/y_1	q_0/y_1	q_0/y_1	q_0/y_1	q_0/y_1	q_0/y_0	q_0/y_1
q_6	q_0/y_1	q_0/y_1	q_0/y_1	q_0/y_1	q_0/y_1	q_0/y_1	q_0/y_1	q_0/y_1

P3.13 Un automat Mealy primeşte serial pe intrarea x cuvinte de câte patru biţi care reprezintă un număr în cod EXCESS-3 şi generează serial pe ieşirea y_1 cuvinte de patru biţi care reprezintă codul BCD al numărului aplicat pe intrare, deci o conversie serială EXCESS-3 în BCD. Dacă pentru codul aplicat pe intrare nu există un număr $(0 \div 9)$ în BCD automatul generează 1 pe ieşirea y_0. Să se realizeze graful de tranziţie al stărilor/ieşiri (primul bit aplicat pe intrare din codul EXCESS-3 este cel mai puţin semnificativ.

P3.14 Să se determine dacă următoarele grafuri de tranziţie sunt definite ambiguu



P3.15 Pentru automatul cu diagrama de semnale din figură să se construiască graful de tranziţie al stărilor/ieşirii.



P3.16 Pentru grafurile de tranziţie ale stărilor/ieşirilor din figura de mai jos să se deducă tabelele de tranziţie ale stărilor/ieşirilor.

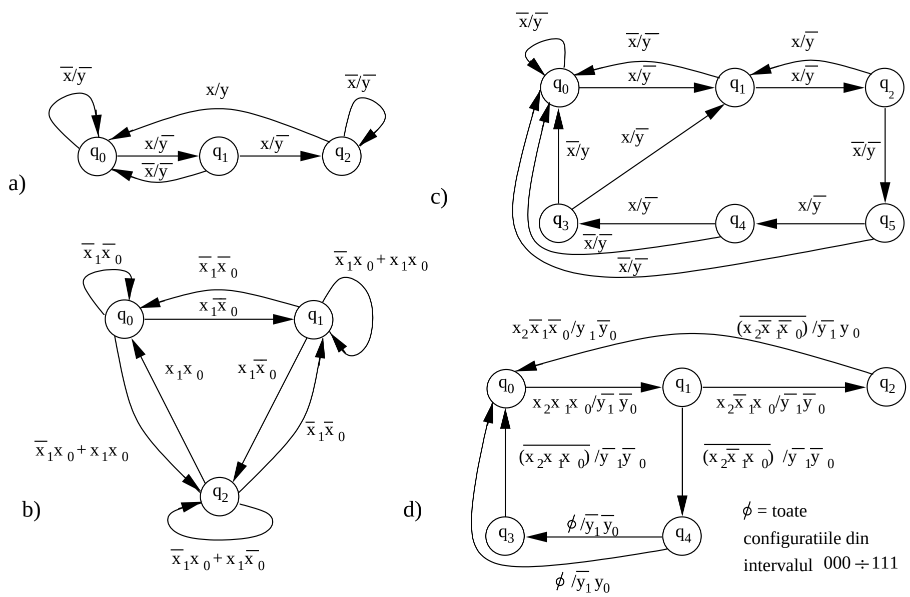


Figura 3.122 Grafurile de tranziţie pentru problema P3.16

P3.17 Pe cele două intrări x_1 şi x_0 ale unui automat sincron se aplică două şiruri de biţi. Automatul va genera o ieşire activă, $y = 1$, când pentru cinci tacte consecutive subşirurile aplicate la cele două intrări sunt identice. Să se realizeze graful de tranziţie al stărilor/ieşirilor şi apoi tabelul de tranziţie al stărilor/ieşirilor.

P3.18 Pentru un anumit experiment este necesar a se detecta dacă o bilă de un diametru D, situată într-un tub, se deplasează în sus sau în jos. Pentru a detecta deplasarea se fixează în lugimea tubului doi senzori S_1 şi S_2 la o distanţa $< D$. Când bila este în dreptul unui senzor acel senzor va indica valoarea logică 1, altfel va indica valoarea logica 0. Când bila se deplasează în sus o ieşire y_1 va avea valoarea logică 1 iar când bila se deplasează în jos ieşirea y_2 va avea valoarea logică 1. Să se conceapă organigrama ASM şi apoi să se construiască tabelul de tranziţie al stărilor/ieşirilor.

P3.19 Graful de tranziţie al stărilor/ieşirilor de la problema P3.7. Să se convertească

într-o diagrama ASM şi apoi să se construiască tabelul de tranziţie al stărilor.

P3.20 Graful de tranziţie al stărilor/ieşirilor din exemplul 3.3 Figura 3.11. Să se convertească într-o organigramă ASM.

P3.21 Să se conceapă organigrama ASM a unui automat ale cărui succesiuni de cuvinte generate la ieşire, $y_2y_1y_0$, la aplicarea impulsurilor de ceas, este o numărare în cod binar natural 8-4-2-1 sau în cod GRAY. Cu ajutorul intrării $\overline{B}/G = 1$ se comandă numărarea în binar natural, $\overline{B}/G = 0$ sau în cod Gray $\overline{B}/G = 1$.

P3.22 Pentru automatul Mealy cu tabelul de tranziţie al stărilor/ieşirilor de mai jos să se reducă stările redundante prin metoda mapei implicanţilor.

a)

Starea prezenta	Starea urmatoare / iesire x=0	x=1
0	0/00	1/00
1	4/00	2/00
2	7/00	1/00
3	2/01	6/10
4	6/10	5/00
5	5/01	4/11
6	1/01	6/10
7	3/10	8/00
8	8/01	7/11

b)

Starea prezenta	Starea urmatoare/iesire x=0	x=1
A	C/0	B/1
B	D/0	B/1
C	A/1	D/0
D	B/1	C/0

c)

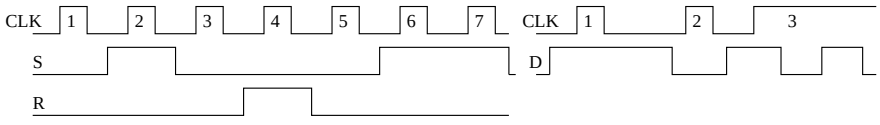
Starea prezenta	Starea urmatoare/iesire x=0	x=1
A	D/0	B/0
B	E/0	A/1
C	G/0	F/1
D	A/1	D/0
E	A/1	D/0
F	C/0	B/0
G	A/1	E/0

P3.23 Pentru cele două automate de tip Mealy cu tabelele de tranziţie ale stărilor/ieşirilor de mai sus (b şi c) să se deseneze grafurile de tranziţie ale stărilor/ieşirilor înainte şi după reducerea stărilor redundante.

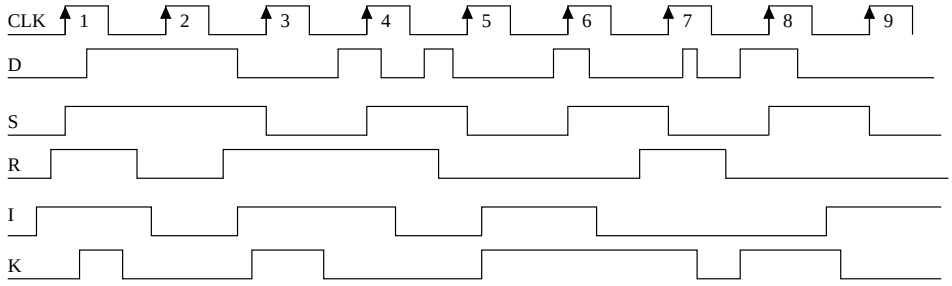
P3.24 Pentru automatul cu tabelul de tranziţie al stărilor/ieşirii din figura următoare să se deducă funcţiile de excitaţie pentru codificarea stărilor în două modalităţi: 1 - cod binar natural; 2 - cod Gray.

Starea prezenta	Starea urmatoare / iesire x=0	x=1
q_0	$q_1/1$	$q_0/1$
q_1	$q_2/0$	$q_3/0$
q_2	$q_2/0$	$q_3/0$
q_3	$q_1/1$	$q_0/0$

P3.25 Următoarele forme de variaţie de semnale se aplică la un latch SR cu ceas şi respectiv la unul D. Să se deseneze variaţia semnalelor Q şi Q_N.



P3.26 Următoarele forme de variaţie de semnale se aplică respectiv la bistabile D, SR şi JK. Să se deseneze variaţia semnalelor Q şi Q_N. Toate bistabilele au comutaţie pe frontul pozitiv de ceas.



P3.27 Formele de variaţie de semnale din figura de mai jos se obţin la ieşirea unui bistabil T respectiv D. Să se deseneze variaţia semnalelor pe intrările T respectiv D. Comutaţia bistabilelor este pe frontul negativ.

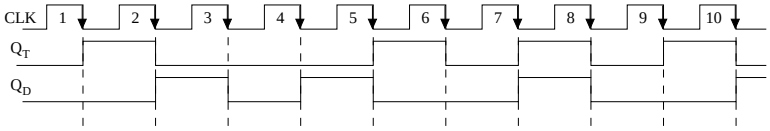


Figura 3.134 Problema P3.27

P3.28 Să se deseneze variaţia semnalului generat pe ieşirilor A şi B ale circuitelor din figura de mai jos:

Figura 3.136 Problema 3.28

P3.29 Să se deseneze variaţia semnalului generat pe ieşirile A şi B ale circuitelor din figura de mai jos. Iniţial bistabilele sunt în starea $Q = 0$.

P3.30 Pentru bistabilele din figura de mai jos se dau următoarele valori: $\tau_{\text{SU}} = 20ns$, $\tau_{\text{pLH(CQ)}} = \tau_{\text{pHL(CQ)}} = 50ns$. Să se determine frecvenţa maximă a semnalului de ceas şi variaţia semnalelor A, B şi C. Iniţial bistabilele sunt în starea $Q = 0$.

P3.31 În Figura 3.40-a este realizată o structura de latch D cu ceas pornind de la un latch SR pe bază de porţi NOR plus două porţi NAND şi un inversor. Să se structureze latch-uri D în felul următor:

a) latch SR pe bază de porţi NOR plus porţi NOR şi inversoare;

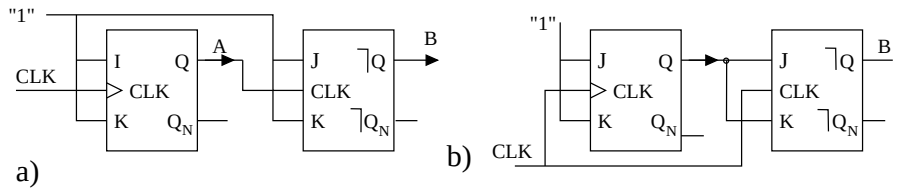


Figura 3.138 Problema 3.29

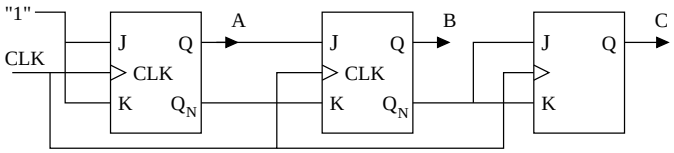
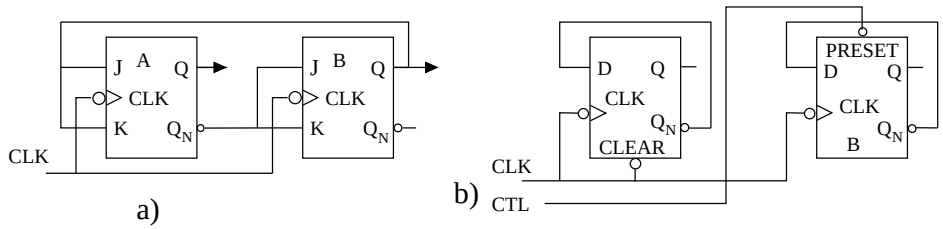


Figura 3.140 Problema 3.30

b) latch SR pe bază de porţi NAND plus porţi NAND şi inversor;
c) latch SR pe bază de porţi NAND plus numai porţi NAND.

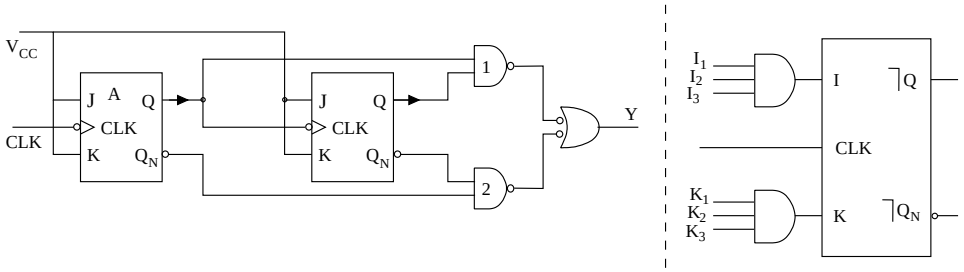
P3.32 Un bistabil PN are următoarea funcţionare:
1) pentru $PN = 00$ înscrie în $Q = 0$, $Q_N = 1$;
2) pentru $PN = 01$ fără modificare Q, Q_N;
3) pentru $PN = 10$ complementează $\overline{Q}$, $\overline{Q}_N$;
4) pentru $PN = 11$ înscrie $Q = 1$, $Q_N = 0$.
Să se realizeze: tabelul caracteristic, tabelul de execitaţie, ecuaţia de funcţionare şi structurarea pe un latch SR master slave. Cum poate bistabilul PN să fie transformat într-un bistabil D? De asemenea să se structureze un bistabil JK utilizând un bistabil D în un MUX 2:1.

P3.33 Pentru următoarele circuite să se deseneze variaţia semnalelor Q_A şi Q_B. Se consideră starea iniţială a bistabilelor $Q_A = 0$, $Q_B = 0$.



P3.34 Pentru următorul circuit (a) să se deseneze variaţia semnalelor Q_A, Q_B şi X când în circuit există următoarele defecţiuni:
a) intrarea J_A este în gol;
b) intrarea K_B este în gol;
c) ieşirea Q_B este în gol;
d) intrarea de ceas la bistabilul B este pusă la masă

e) poarta $NAND_2$ are ieşirea în gol.
Se consideră că circuitul este realizat cu componente TTL.



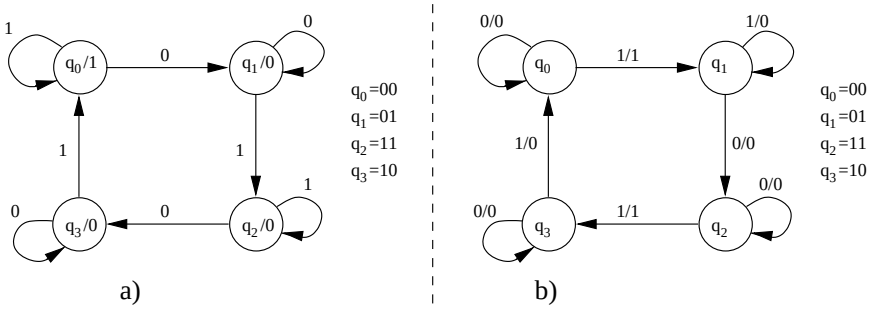
P3.35 Pentru bistabilul de mai sus (b) să se deseneze variaţia semnalului de ieşire Q când pe intrare se aplică următoarele şiruri de biţi: $J_1 = 1010011$, $J_2 = 0111010$, $J_3 = 1111000$, $K_1 = 0001110$, $K_3 = 1010101$. Pentru fiecare impuls de ceas se aplică, din şirurile date, câte un bit pe fiecare intrare; se incepe cu bitul cel mai din dreapta.

P3.36 Să se proiecteze un automat cu bistabile D care pentru intrarea $x = 1$ generează succesiv, şi ciclic, codurile Gray de doi biţi, iar pentru $x = 0$ rămâne în aceeaşi stare.

P3.37 Să se proiecteze un automat cu bistabile JK, cu două intrări E(nable) şi x care pentru $E = 0$ va rămâne în aceeaşi stare indiferent de valoarea lui x. Dar când $E = 1$ va genera la ieşire, ciclic, succesiunea $y_1y_0 = 00, 01, 10, 11, 00, 01, \cdots$ pentru $x = 1$, respectiv, ciclic, succesiunea $y_1y_0 = 00, 11, 10, 01, 00, 11 \ldots$ pentru $x = 0$.

P3.38 Să se proiecteze un automat care pentru un cuvânt binar de n biţi, $b_{n-1}, b_{n-2} \ldots b_1, b_0$, aplicat serial pe intrarea x, începând cu bitul b_0 - câte un bit pe fiecare tact - generează serial pe ieşirea y complementul faţa de doi al cuvântului aplicat pe intrare; în acelaşi tact al aplicării bitului pe intrare se obţine şi bitul corespunzător pe ieşire. Pentru a indica faptul că secvenţa aplicată pe intrare s-a terminat şi că circuitul se iniţializează să primească o altă secvenţă automatul mai are o intrare I care trebuie să primească valoarea 1 pentru un tact, altfel $I = 0$. Implementarea se va face cu bistabil D.

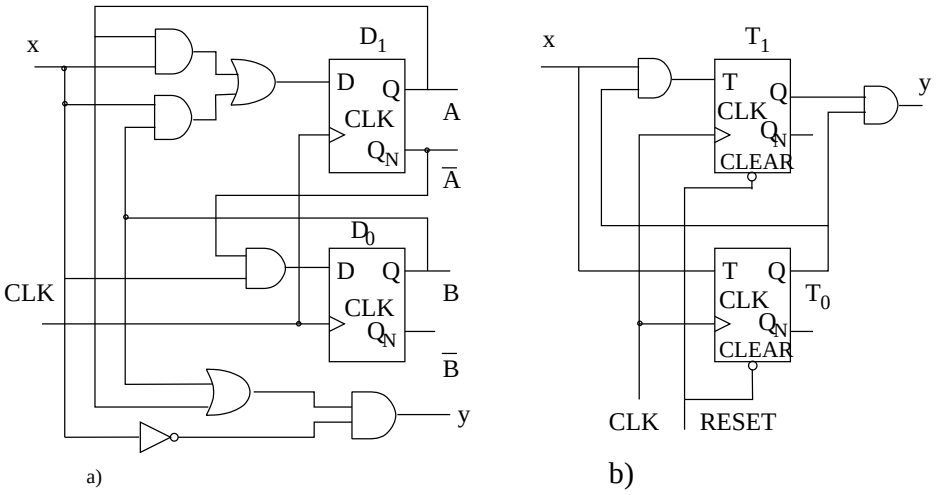
P3.39 Pornind de la graful de tranziţie al stărilor reprezentat în figura următoare (a) să se realizeze automatul întâi cu bistabile D apoi cu bistabile JK.



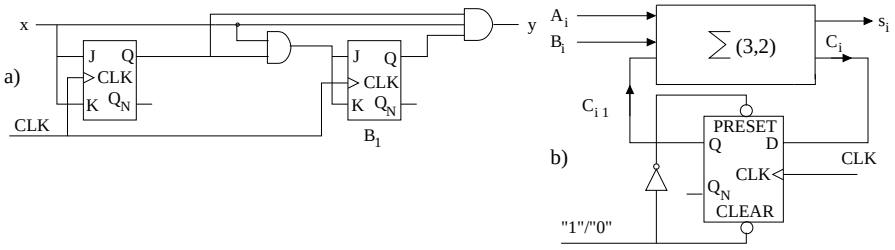
P3.40 Pornind de la graful de tranziţie al stărilor reprezentat în figura de mai sus (b) să se implementeze automatul în varianta cu bistabile JK şi bistabile T.

P3.41 Pornind de la grafurile de tranziţie al stărilor şi tabelele de tranziţie al stărilor deduse în problemele P 3.4, P 3.5 şi P 3.6 să se implementeze automatele pe bază de bistabile de tip D.

P3.42 Pentru automatele din figura a (cu bistabile D) şi din figura b (cu bistabile T) de mai jos să se deducă tabelele de tranziţie ale stărilor şi ieşirilor, apoi grafurile de tranziţie ale stărilor.



P3.43 Pentru automatul din figura a de mai jos să se construiască tabelul de tranziţie al stărilor/ieşirii şi graful de tranziţie al stărilor/ieşirii. Apoi pentru $x = 1$ să se deseneze diagrama de variaţie a semnalelor z_1, z_0, y pe un interval de 10 tacte de ceas, pornind din starea $z_1 z_2 = 00$



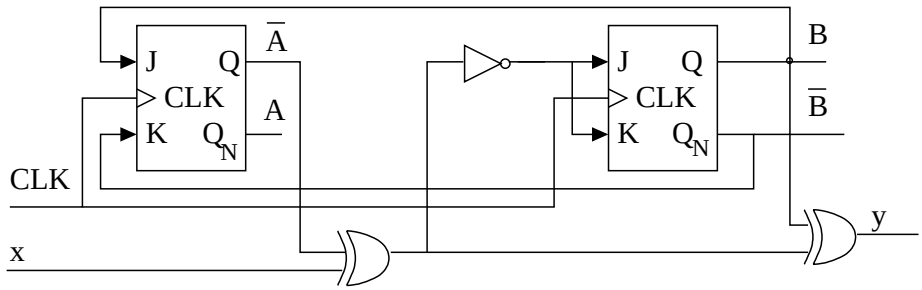
P3.44 Circuitul secvenţial din figura b, desenat la problema P3.43 are două intrări A_i, B_i şi o ieşire s_i. Structural, se compune dintr-un sumator complet, $\sum(3,2)$, şi un bistabil D. Să se deducă tabelul de tranziţie al stărilor/ieşirii şi graful de tranziţie.

P3.45 Un semiautomat are trei bistabile Q_2, Q_1 şi Q_0 de tip D şi o intrare x. Ecuaţiile care descriu funcţiile de excitaţie sunt:

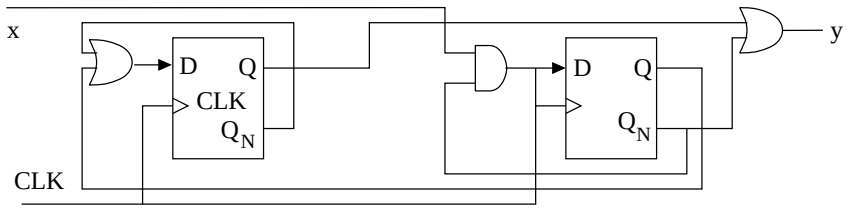
$$wD_2 = (z_1\overline{z}_0 + \overline{z}_1 z_0)x + (z_1 z_0 + \overline{z}_1\overline{z}_0)\overline{x}; \quad wD_1 = z_2, \quad wD_0 = z_1$$

a) Să se construiască tabelul de tranziţie al stărilor;
b) Să se realizeze două grafuri de tranziţie ale stărilor unul pentru $x = 0$ şi altul pentru $x = 1$.

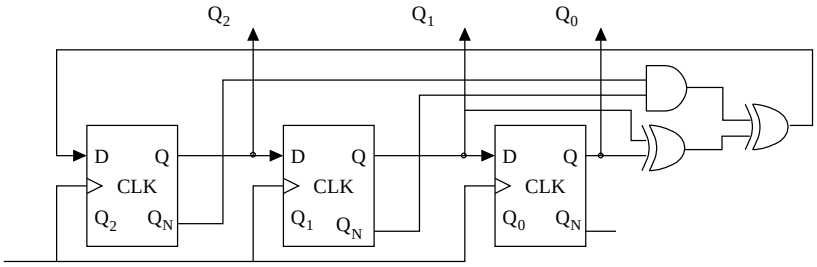
P3.46 Automatul din figura de mai jos este construit pe baza a două bistabile, Q_1 este un JK iar Q_2 este de tip T. Să se deducă tabelul de tranziţie al stărilor/ieşirii şi graful de tranziţie al stărilor/ieşirii.



P3.47 Pentru automatul din figura de mai jos să se scrie funcţiile de excitaţie/ieşire şi să se deducă tabelul de tranziţie al stărilor/ieşirilor iar apoi să se construiască graful de tranziţie al stărilor/ieşirilor.



P3.48 Să se analizeze automatul din figura de mai jos.

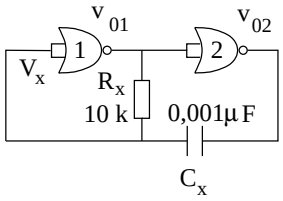


P3.49 Pentru automatul din figura a) de la problema P 3.42, considerând că se află în starea $z_1 z_0 = 00$, să se determine succesiunea stărilor şi succesiunea valorilor ieşirii când pe intrare se aplică şirul de valori: 010110111011110.

P3.50 Pentru circuitul din Figura 3.58-a cu valorile $V_{DD} = 5V$, $V_T = 2,5V$, $R_x = 10k\Omega$ şi $C_x = 0,001\mu F$.

a) Să se calculeze durata τ_w a impulsului generat la ieşire;
b) Dacă variaţia tensiunii de prag V_T este $\pm 20\%$ care este variaţia în durata impulsului generat.

P3.51 Pentru circuitul astabil din figura de mai jos să se deseneze diagrama de variaţie a semnalelor şi să se deducă formula pentru calculul frecvenţei.



P3.52 Pentru circuitul 555 să se determine: a) fiind conectat ca monostabil cu $R_x = 22k\Omega$ şi $C_x = 0,01\mu F$, care este durata τ_w a impulsului generat; b) fiind conectat ca astabil, să genereze un semnal dreptunghiular de frecvenţă 1 KHz şi factorul de umplere 75%.

P3.53 Cu un circuit 555 să se realizeze un receptor de linie compatibil la ieşire TTL.

P3.54 Utilizând un circuit 555 să se realizeze un releu cu întârziere la închidere.

P3.55 Pentru un numărător asincron modulo 8, realizat cu bistabile T, cu comutaţie pe frontul negativ, să se determine stările false introduse în secvenţa de numărare. Cum se poate obţine ieşirea $z_2z_1z_0$ fără stări false.

P3.56 Pentru un numărător asincron modulo 8, realizat cu bistabile cu comutaţie pe frontul pozitiv având timpul de propagare $\tau_{pCQ} = 8ns$ să se determine timpul cel mai lung de propagare şi între care stări se obţine.

P3.57 Să se modifice, utilizând o buclă de forţare netransparentă, un numărător asincron modulo 16 pentru o funcţionare de numărător modulo: 9, 11, 13, 14 şi 15.

P3.58 Să se realizeze un generator de faze utilizând un numărţor asincron modulo 8 (cu comutaţie pe frontul negativ al ceasului).

P3.59 Pentru un numărător asincron modulo 2^{10} cu $\tau_{pCQ} = 5ns$: a) când conţinutul este unul din numerele 10011000111, 00111111111, 11111111111 şi se aplică următorul impuls de ceas câte celule comută? b) care este frecvenţa maximă de ceas?

P3.60 Utilizând circuitul numărător 74xx163 (Figura 3.66-a) să se realizeze: a) două structuri de numărător modulo 11 în cod oarecare; b) un numărător modulo 11 în cod binar natural; c) un numărător modulo 11 în cod EXCESS3.

P3.61 Utilizând circuitul numărător 74xx161 (Figura 3.66-a) să se structureze numărătoare modulo 13 în cod binar.

P3.62 Pentru structurile din Figura 3.172 (a, b, c), pe baza circuitului numărător 74xx169 (vezi Figura 3.66-a) să se determine succesiunea cuvintelor de ieşire $z_3z_2z_1z_0$ atât pentru cazul când se consideră cuvântul iniţial $z_3z_2z_1z_0 = 0000$ cât şi pentru cazul când $z_3z_2z_1z_0 = 1111$.

P3.63 Să se structureze numărătoare modulo 2^{16}, în cod binar natural, pe baza circuitului numărător 74xx163.

P3.64 Să se structureze două numărătoare modulo 129 (unul în cod binar natural, altul în cod oarecare) pe baza circuitului 74xx163.

P3.65 Cu celule bistabil D să se realizeze sinteza următoarele numărătoare:

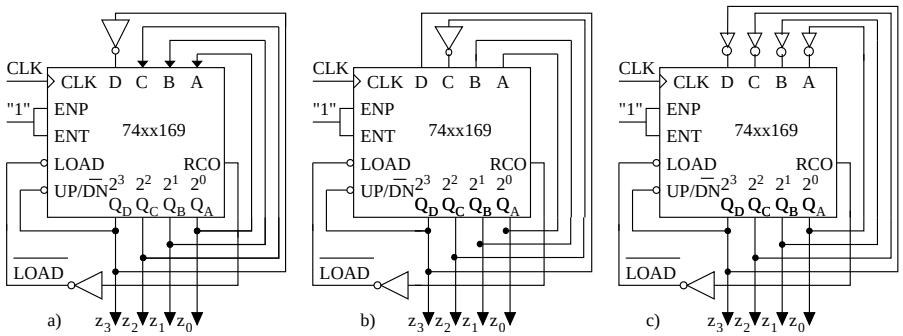


Figura 3.172 Problema 3.62

a) numărător modulo 16 în cod binar natural;
b) numărător modulo 6 care realizează următorul ciclu (notat în zecimal) 0-1-3-7-6-4-0-1-3...

P3.66 Cu celule bistabil T să se realizeze sinteza unui numărţor modulo 10 în cod BCD.

P3.67 Cu celule bistabil JK să se realizeze sinteza următoarelor numărătoare:
a) numărător modulo 3 care generează ciclic succesiunea 0-1-2-0-...;
b) numărător modulo 6 care generează ciclic succesiunea 0-1-3-2-4-6-0-1....

P3.68 Să se structureze un circuit care să multiplice frecvenţa de ceas cu coeficientul 0,375.

P3.69 Să se structureze un ceas care indică pe afişoare cu şapte segmente, timpul până la 12 ore. Pentru obţinerea semnalului de ceas se va utiliza frecvenţa reţelei electrice de 50 Hz.

P3.70 Pentru un spaţiu de parcare, cu capacitatea maximă $C_{max} = 100$ locuri, să se indice la intrare, în fiecare moment, printr-un semnal luminos dacă mai există locuri de parcare, iar dacă s-a ajuns la C_{max} să se închidă o barieră pe sensul de intrare.

P3.71 Pentru organigrama ASM din figura a) de mai jos să se implementeze semiautomatul.

P3.72 Pentru organigrama ASM din figura b) de mai sus să se implementeze semiautomatul.

P3.73 Utilizând un limbaj RTL (vezi 3.2.3.5) pentru un grup de patru registre $R3, R2, R1, R0$ cu lungimea de 32 biţi, să se specifice într-o diagramă ASM stările pentru efectuarea următoarelor transferuri: $R0 \leftarrow R1$, $R2 \leftarrow R3$; $SWAPR1, R2$ ($R1 \leftrightarrow R2$). Se vor structura două tipuri de conexiuni între registre: 1) fiecare cu fiecare (punct-la-punct); 2) prin intermediul unei singure magistrale.

P3.74 Pe baza unui circuit acumulator de patru biţi să se realizeze un numărător în cod BCD.

P3.75 Pe baza unui circuit acumulator de patru biţi să se realizeze un numărător modulo 12.

P3.76 La sumatorul din problema P 3.44: a) să se introducă registre de deplasare

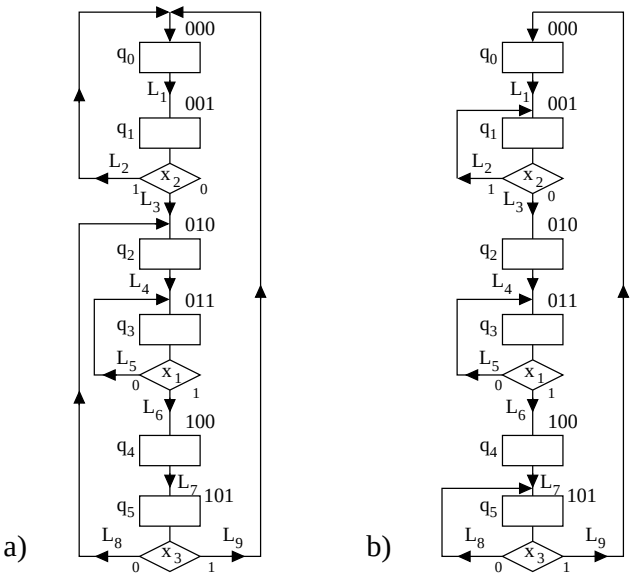


Figura 3.182 Problema 3.71

pentru operanzi şi suma; b) să se realizeze cu bistabil de tip JK

P3.77 În structura numărătorului Johnson, cu patru celule, să se modifice conexiunile astfel încât să genereze un număr impar (şapte) de stări.

P3.78 Pe baza unui numărător Johnson, cu cinci celule, să se realizeze:
a) un generator de 10 faze: $F_0, F_1, F_2, \ldots, F_8, F_9, F_0, F_1, \ldots$
b) un generator de 9 faze: $F_0, F_1, F_2, \ldots, F_7, F_8, F_0, F_1, \ldots$

P3.79 Utilizând circuitul registru universal 74xx194 (Figura P3.80-b şi P3.80-c) să se realizeze un circuit registru inel, fără autoamorsare: a) cu deplasare stânga, din starea iniţială $z_4z_3z_2z_1 = 0001$; b) cu deplasare dreapta, din starea iniţială $z_4z_3z_2z_1 = 1100$.

P3.80 Pe baza registrului 74xx194 să se realizeze un numărător în inel cu deplasare stânga cu autoamorsare şi autocorecţie cu ciclul $0001 \rightarrow 0010 \rightarrow 0100 \rightarrow 1000 \rightarrow 0001 \rightarrow \ldots$

P3.81 Pe baza circuitului 74xx194, Figura P3.80, registru universal cu patru celule, să se realizeze: a) un numărător Johnson de patru celule cu autoamorsare; b) un numărător Johnson cu autoamorsare şi autocorecţie.

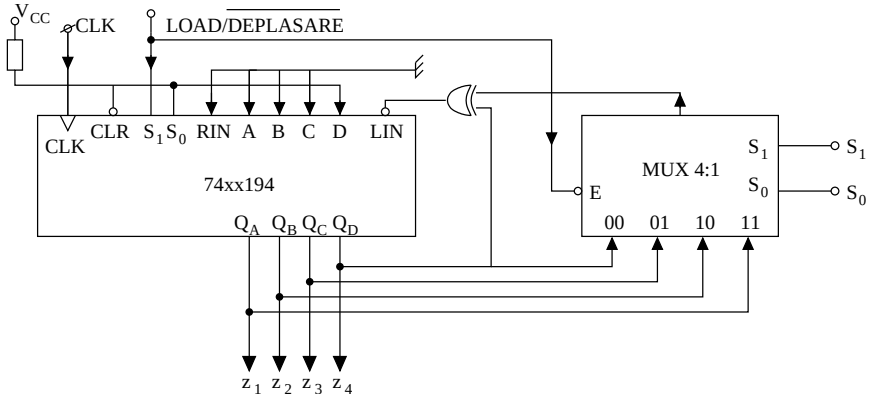
P3.82 Pentru polinomul caracteristic $x^4 \oplus x_3 \oplus I$ să se structureze circuitul secvenţial liniar cu reacţie cu sumator extern şi cu sumator inclus. Pentru fiecare din aceste structuri să se determine succesiunea stărilor generate când se consideră starea iniţială $z_4z_3z_2z_1 = 0001$. Apoi, să se introducă circuit de autocorecţie (şi autoamorsare) şi sa se deducă succesiunea stărilor.

P3.83 Să se realizeze structura de circuite secvenţiale liniare cu reacţie, cu sumator extern şi inclus, pentru polinomul caracteristic $x^4 \oplus x^3 \oplus x \oplus 1$.

P3.84 Utilizând trei celule ale registrului universal 74xx194 să se realizeze structuri de circuite secvenţiale liniare cu reacţie care să genereze secvenţe pseudo-aleatoare cu

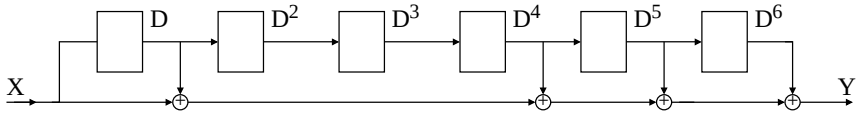
lungimea maximă; apoi să se structureze un numărător modulo 5 în cod arbitrar.

P3.85 Circuitul din figura reprezintă un generator de secvenţe pseudo-aleatoare selectabil pentru secvenţe de cuvinte de patru biţi $z_4z_3z_2z_1$, realizat pe baza circuitului registru universal 74xx194. Selectarea se realizează prin intermediul unui MUX4:1. Pentru cele patru cuvinte de selectare S_1S_0 ale multiplexorului să se determine secvenţele de cuvinte $z_4z_3z_2z_1$ precum şi şirul de biţi aplicat pe intrarea serie deplasare stânga, LIN. Se consideră iniţializarea generatorului în starea $z_4z_3z_2z_1 = 0001$ realizată prin acţionarea $LOAD/\overline{DEPLASARE} = 1$.



P3.86 Să se arate că numai un registru serie cu un număr n, par/impar, de celule şi o reţea de reacţie cu paritate respectiv impară/pară, conectate într-o structură de circuit secvenţial liniar cu reacţie, poate genera o secvenţă de lungime maximă, $2^n - 1$.

P3.87 Pentru circuitul secvenţial liniar fără reacţie din figura următoare să se determine răspunsul la o secvenţa impuls unitar, la o secvenţă treaptă unitară şi să se calculeze secvenţa de intrare nulă.



P3.88 Două circuite secvenţiale liniare fără reacţie cu funcţiile de transfer $H_1(D)$ şi $H_2(D)$ sunt comandate cu aceeaşi secvenţă de intrare $X = \ldots 0001011000\ldots$, obţinându-se secvenţele de ieşire: $Y_1 = \ldots 00011101101000\ldots$, $Y_2 = \ldots 00010110$ $011101000\ldots$ Să se deducă expresiile algebrice şi să se deseneze structurile pentru cele două circuite. Apoi să se înserieze cele două funcţii de transfer $H_3(D) = H_1(D) \cdot H_2(D)$ iar pentru circuitul rezultat să se deseneze structura şi să se determine răspunsul pentru secvenţa de intrare $X_3 = \ldots 000111000\ldots$

P3.89 Cu trei circuite secvenţiale liniare fără reacţie cu funcţiile de transfer $H_1(D)$, $H_2(D)$, $H_3(D)$, care pentru secvenţa $X = \ldots 000111000\ldots$ aplicată pe intrări generează ieşirile: $Y_1 = \ldots 0001011101000\ldots$, $Y_2 = \ldots 0001011101000\ldots$, $Y_3 = \ldots 0001101110101000\ldots$ să se realizeze circuitele cu funcţiile de transfer: $H_4(D) = H_1(D) \oplus H_2(D) \oplus H_3(D)$; $H_5(D) = H_1(D) \cdot H_2(D) \oplus H_3(D)$;

$H_6(D) = H_1(D) \cdot H_2(D) \cdot H_3(D)$.

P3.90 Utilizând un circuit RAM de capacitate 1M×4 biţi să se realizeze următoarele module de capacitate: 1M×4 biţi; 2M×4 biţi; 4M×4 biţi şi 2M×8 biţi.

Capitolul 4

SUPORTUL CIRCUISTIC ÎN PROIECTAREA APLICAŢIILOR

4.1 CONEXIUNI PROGRAMABILE

Succesiunea în desfăşurarea procesului de dezvoltare a unui sistem digital se reduce, mai mult sau mai puţin, la următoarele trei etape: DESCRIERE→SINTEZĂ → REALIZARE.

Descrierea funcţionalităţii se referă la acele specificaţii proprii aplicaţiei realizate de/cu sistemul digital care urmează a fi dezvoltat. Această descriere conduce la elaborarea unei **arhitecturi**, în general, compusă din mai multe blocuri între care există schimb informaţional, în timp, sub formă de semnale electrice. Pentru exprimarea funcţionării sistemului digital, la nivel de arhitectură, se poate utiliza fie o modalitate grafică (desene), fie o modalitate textuală (limbaje de descriere, HDL, vezi vol.II).

Sinteza constă în convertirea arhitecturii într-o structură logică, adică sinteza reţelei logice a sistemului.

Realizarea constă în detalierea reţelei logice la nivel de componentă (poartă, tranzistor) şi implementarea într-o anumită tehnologie.

Evident, această succesiune a etapelor poate fi cu iteraţii pe fiecare etapă sau chiar cu iteraţii de la ultima etapă la prima până când specificaţiile impuse aplicaţiei sunt realizate. În final, produsul/aplicaţia se prezintă sub forma unei implementări cu circuite integrate discrete conectate pe o placă de circuit imprimat sau sub forma unui singur circuit integrat, **SOC** (**S**ystem-**O**n-a-**C**hip). Între aceste două modalităţi extreme de implementare pot exista diferite variante în funcţie de cât de multe părţi ale implementării de pe placa de circuit imprimat pot fi transferate într-o implementare în siliciu. Aceste variante de implementare pe un circuit integrat sunt determinate de anumite criterii impuse aplicaţiei cum sunt: performanţa (în general viteză), consumul de putere, suprafaţa ocupată pe siliciu, timpul de dezvoltare, flexibilitatea pentru modificare (**reconfigurabilitatea**), siguranţa în funcţionare şi nu în ultimul rând costul.

Produsul în totalitate se constituie dintr-o componentă hardware şi o componentă software (compilator, program de încărcare, program de testare şi depanare, program de lucru interactiv cu sistemul elaborat, sistem de operare (executiv), etc). Pentru ponderarea între hard şi soft se analizează cum contribuie acestea, hard-ul prin performanţele de viteză iar soft-ul prin flexibilitate, la performanţele şi costul sistemului.

Costul unui circuit integrat poate fi aproximat pe baza următoarei relaţii:

$$\textbf{Cost/unitate} = \frac{\text{Costul dezvoltării}}{\text{Volumul producţiei}} + \text{Costul de fabricaţie/unitate} \qquad (4.1)$$

Costul de fabricaţie/unitate, când complexitatea procesului de fabricaţie este deja stăpânită, în general, este proporţional cu dimensiunea pe siliciu şi, uzual, pentru circuitele integrate comercial obtenabile se situează în ordinul unităţi sau zeci de EUR; **acest cost este repetabil** pentru fiecare circuit integrat.

Costul dezvoltării conţine cheltuielile pentru munca de proiectare, investiţia în instrumentele de proiectare (programe **CAD-C**omputer **A**ided **D**esign, calculatoare etc) plus cheltuielile auxiliare (regia). Costul dezvoltării poate fi foarte mare, în general de la zeci de mii de EUR în sus. Această componentă a costului/unitate poate fi redusă fie printr-o micşorare a costului procesului de proiectare, fie printr-o mărire a volumului producţiei sau prin ambele simultan; **acest cost nu este repetabil**, **NRE** (**N**onrecurring **E**ngineering **C**ost), se distribuie pe întregul lot fabricat deci nu favorizează seriile mici.

Proiectarea unui circuit electronic, ideal, ar trebui efectuată pentru obţinerea de performanţe maxime şi într-un timp cât mai scurt ceea ce este contradictoriu, în consecinţă se alege una din aceste două abordări: proiectare pentru performanţe maxime sau proiectarea într-un timp scurt.

Proiectarea pentru performanţe maxime implică un cost al dezvoltării foarte ridicat de ordinul de la sute de mii de EUR în sus şi care nu se amortizează dacă volumul producţiei nu este foarte mare (recomandată o astfel de producţie pentru produse de larg consum, industria auto, jucării, etc); în general o astfel de proiectare se realizează cu o echipă de zeci de ingineri $\times$ an. Acest tip de proiectare este referită ca proiectare complet realizată de către utilizator - **full custom design**.

Abordarea dezvoltării printr-o **proiectare în timp scurt**, evident, duce la apariţia rapidă a produsului pe piaţă. În unele cazuri apariţia pe piaţă a unui nou produs cu un an mai devreme decât al competitorului poate duce la un profit dublu faţă de competitor. Acest avantaj se explică prin faptul că primul în piaţă captează pe toţi clienţii şi la un preţ ridicat pe când la următorii veniţi în piaţă rămân restul de clienţi şi la un preţ scăzut; cu alte cuvinte profitul obţinut printr-o proiectare rapidă, foarte frecvent, depăşeşte profitul obţinut printr-o proiectare pentru performanţă (iar la a doua proiectare, care apare pe piaţă odată cu cea a competitorului, se realizează pentru produs performanţe foarte ridicate!) Dar, în general, o proiectare rapidă se face pentru o producţie în volum redus ceea ce nu ar duce la valori mici pe unitate pentru componenta costului de dezvoltare, relaţia 4.1. Soluţia, în acest caz, este să se utilizeze "prefabricate" adică circuite deja proiectate şi fabricate într-un stadiu avansat, cu mult peste 50% din efortul total de realizare, de către fabricantul de circuite integrate (turnătoria de siliciu, în general) şi care sunt preluate şi continuate în procesul de realizare de către beneficiar/client; ceea ce înseamnă o "personalizare"

a circuitului pentru aplicaţia sa cu un efort de dezvoltare (timp şi cost) destul de redus şi acceptabil pentru producţie de serie mică spre medie. În general, această "personalizare" se reduce la realizarea (programarea) unor conexiuni pe un suport integrat existent ("prefabricatul") conform unui proiect realizat de către client pentru aplicaţia sa specifică; o astfel de abordare a dezvoltării unui sistem digital la care participă şi clientul este referită prin **semicustom design**, mai nou referită şi prin abreviaţia **ASIC design** (**A**pplication **S**pecific **I**ntegrated **C**ircuit design).

Pentru realizarea personalizării circuitului de către utilizator la aplicaţia sa circuitul integrat "prefabricat" trebuie să asigure un anumit suport în efectuarea conexiunilor dorite. În acest sens pe circuitul "prefabricat" există posibilitatea de a se realiza anumite trasee noi pentru interconectare sau de a se realiza joncţionarea anumitor segmente de trase deja existente pentru o anumită conectare. În oricare din aceste variante o deosebită atenţie trebuie acordată întârzierii introduse, la propagarea semnalului, de aceste trasee de interconectare, întârziere care este proporţională cu rezistenţa şi capacitatea totală a traseului. Traseele noi pentru interconectare pot fi realizate pe suprafaţa de siliciu a circuitului integrat sau pe straturile metalice situate deasupra suprafeţei de siliciu. Realizarea acestor noi trasee necesită continuarea procesării, la turnătoria de siliciu, cu realizarea a $1 \div 3$ măşti pe baza informaţiei elaborate de către utilizator; această modalitate este referită ca **programarea (circuitului) prin mascare**. La circuitele semicustom cu programare prin mascare se obţin cele mai mici întârzieri pe traseele noi de interconectare în raport cu alte variante de realizare a conexiunilor programate.

Pentru realizarea traseelor de interconectare, prin joncţionarea anumitor segmente de linii metalizate deja implementate, trebuie să existe între capetele segmentelor respective câte un "comutator" ce poate fi închis sau deschis pentru conectarea sau separarea segmentelor respective. De fapt, acest "comutator" se realizează prin plasarea în punctele de joncţionare a unui fuzibil, antifuzibil, tranzistor cu poartă flotantă (de tip EPROM sau EEPROM), tranzistor de trecere sau poartă de transmisie. Realizarea unei conexiuni – o singură dată programabilă, **OTP** – prin intermediul unui fuzibil sau prin intermediul unui antifuzibil au fost prezentate în secţiunea 1.2, Figura 1.12, şi în secţiunea 2.4.6.1; ambele modalităţi pentru realizarea conexiunii prezintă avantajul unui consum redus de suprafaţă (cu puţin mai mare decât dublul lăţimii unei trase metalizate). Din păcate, deoarece pentru realizarea acestor conexiuni trebuie să se aplice din exterior tensiuni de programare în gama $10 \div 20V$, este necesară o circuistica auxiliară (tranzistoarele de programare cu dimensiuni mărite, tranzistoarele de izolare care trebuie să protejeze, de tensiunile ridicate în timpul programării, tranzistoarele ce operează la tensiuni reduse) ceea ce diminuează avantajul amintit anterior. Se preferă antifuzibilul, în raport cu fuzibilul, datorită avantajului de a nu prezenta posibilitatea de revenire la starea anterioară programării, în plus se pot obţine pe conexiune capacităţi şi rezistenţe mai reduse (unele variante de antifuzibil pe bază de siliciu amorf pot realiza rezistenţe în gama $50 \div 100\Omega$) în comparaţie cu modalităţile de realizare a conexiunilor prezentate în continuare.

Conexiunile de tip EPROM şi EEPROM, secţiunea 2.4.6.1, bazate pe tranzistorul cu poartă flotantă prezintă avantajul nevolabilităţii la dispariţia tensiunii de alimentare dar are şi dezavantajele: rezistenţă ridicată în conducţie a tranzistorului ($2 \div 4K\Omega$), consum ridicat de putere (în regim static).

Conexiuni pe bază de de celule SRAM. O celulă de memorie statică prin bitul

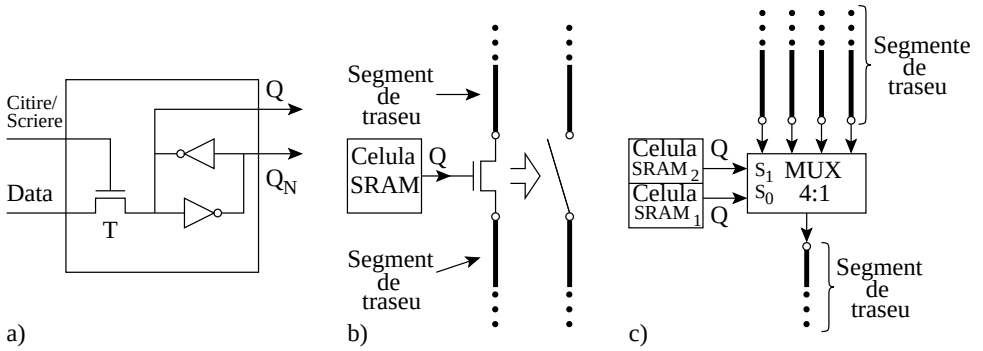


Figura 4.1 Conexiunea programabilă pe bază de celule SRAM: a) structura unei celule SRAM pentru controlul unei conexiuni; b) conexiune programată prin comanda unui tranzistor de trecere; c) conexiuni programate prin comanda unui multiplexor cu celule SRAM (în acest caz două celule SRAM comandă un MUX $2^n : 1$).

stocat 1 ($Q = 1$, $Q_N = 0$) sau 0 ($Q = 0$, $Q_N = 1$) poate comanda în conducţie sau în blocare un tranzistor de trecere, tranzistor care realizează joncţiunea între două capete de segmente de trasee de interconectare. Tranzistorul de trecere realizează funcţia unui comutator, între cele două capete, care este închis când celula SRAM este înscrisă în 1 şi este deschis când celula este înscrisă în 0, Figura 4.1-b; rezistenţa tranzistorului în conducţie, deci a punţii între cele două segmente de traseu, este în gama $0,5 \div 2K\Omega$. În mod similar, pe bază de celule SRAM, se poate realiza selectarea dintr-un grup de 2^n trasee doar a unuia şi conectarea acestuia la un alt segment de traseu prin intermediul unui MUX $2^n : 1$; biţii cuvântului de selectare aplicaţi multiplexorului sunt înscrişi în n celule SRAM, în Figura 4.1-c este prezentat cazul pentru $n = 2$.

Structura celulei SRAM se compune dintr-un latch cu inversoare plus un tranzistor de trecere T, prin care se comandă operaţia de înscriere/citire date la celulă, Figura 4.1-a. Celula este înscrisă (tranzistorul T în conducţie) în procesul de programare a circuitului şi este citită în procesul de depanare; în condiţii normale de funcţionare tranzistorul de trecere T din celulă este blocat iar ieşirea Q a celulei, prin valoarea sa, controlează conducţia sau blocarea unui tranzistor de trecere plasat între două capete de segmente de traseu de interconectare. Într-o astfel de utilizare celula SRAM are o frecvenţă a operaţiilor de citire/înscriere mult mai mică decât a unei celule dintr-o matrice de memorie convenţională; în consecinţă, o astfel de celulă SRAM se dimensionează în primul rând pentru a asigura o stabilitate şi densitate ridicată şi în al doilea rând pentru viteză. Aceste celule de memorie nu formează o matrice compactă, ele sunt dispersate pe întreaga suprafaţă a circuitului, sunt plasate în apropierea tranzistoarelor sau multiplexoarelor pe care trebuie să le controleze. Suprafaţa necesară pentru realizarea unei conexiuni este relativ destul de mare, necesită cel puţin cinci tranzistoare din structura celulei SRAM, plus tranzistorul de trecere comandat. De asemenea, conexiunea programată este volatilă, valoarea logică înscrisă în celulă este anulată la dispariţia tensiunii de alimentare; deci programarea conexiunilor trebuie realizată la fiecare punere sub tensiune a circuitului. În consecinţă, lângă circuitul

programat trebuie să existe un mecanism de memorare a informaţiei de programare (de configurare) a conexiunilor materializat printr-un PROM, EPROM, EEPROM sau date pe un hard(disk). Cu toate acestea, în raport cu celelalte modalităţi de realizare a conexiunilor programate, cea bazată pe celule SRAM, s-a impus datorită următoarelor avantaje:

- reprogramare rapidă (similar cu comutarea pe calculator la alt program);
- reprogramare dinamică (în circuit);
- proces standard de fabricaţie pentru comutatoarele programabile.

4.2 PROIECTAREA DE TIP FULL-CUSTOM

Proiectarea de tip full-custom este o proiectare pentru performanţă, consumatoare de investiţie iniţială, şi necesită o echipă de proiectanţi cu specializare ridicată, de ordinul zeci de ingineri $\times$ an, ceea ce determină un cost de dezvoltare foarte ridicat. În plus, pentru că durata de dezvoltare este lungă, de peste un an, pentru a obţine performanţe ridicate produsul trebuie proiectat a fi implementat într-o tehnologie care se experimentează în aceeaşi perioadă cu proiectarea circuitului. Procesul de dezvoltare are în succesiune etapele de: arhitectură–sinteză–realizare, dar fiecare etapă este realizată minuţios pentru a se obţine performanţe maxime. După parcurgerea etapelor în sensul anterior (în jos, **top-down**) se reia, prin iteraţii, şi parcurgerea în sens invers (în sus) pentru a se verifica îndeplinirea specificaţiilor, deci se porneşte de la nivel de layout.

Proiectarea geometriei tranzistoarelor în siliciu este o operaţie obligatorie pentru un **circuit full-custom**. Apoi, mai multe tranzistoare sunt integrate în celule care la fel sunt transpuse în layout şi testate. Acest proces de proiectare are o ascendenţă graduală în sensul că prin compunerea acestor celule se generează un layout pentru o celulă mai mare care la rândul său poate intra în geometria altei celule de dimensiune şi funcţionalitate mai ridicată ş.a.m.d până la circuitul final. Layout-ul fiecărui nivel de celulă este minuţios proiectat, verificat, simulat şi testat şi nu se trece la nivelul de celulă superioară până când specificaţiile nu sunt îndeplinite (o eroare strecurată la un nivel şi sesizată mai târziu poate fi remediată cu costuri foarte mari – timp şi bani).

Pentru generarea layout-ului nu totdeauna există programe software de proiectare suficient de performante deci, adeseori, se recurge la o proiectare manuală sau se intermixează proiectarea manuală cu cea automată. Obţinerea unor performanţe ridicate de viteză impune analiza întârzierilor atât pe liniile de conexiune cât şi pe nivelul unei porţi logice; restricţionarea numărului maxim pentru fan-out şi pentru fan-in este foarte importantă (vezi secţiunea 1.5.6). Proiectarea de tip full-custom produce un layout compact, de suprafaţă redusă şi de viteză ridicată; uzual pentru o implementare în aceeaşi tehnologie un circuit full-custom este, uzual, de $(3 \div 8)$ ori mai rapid decât acelaşi circuit realizat semicustom [Chinnery ′02]. Să nu uităm, că oricare circuit integrat, la origine, a fost full-custom!

4.3 PROIECTAREA CU ARII DE PORŢI LOGICE

Aria de porţi logice este un circuit integrat ce conţine o mulţime de porţi logice ce sunt plasate sub o formă matriceală dar fără a fi conectate între ele, Figura 4.2-a. O astfel de structură, datorită unei organizări matriceale, este referită adesea şi ca **matrice de porţi** (neconectate). Pe baza acestor porţi, prin realizarea conexiunilor între acestea, conform unor relaţii logice, se obţine circuitul logic corespunzător, Figura 4.2-b.

Dar, în prezent, circuitele arie de porţi nu se realizează ca o matrice de porţi logice ci ca o matrice structurată pe baza unor componente (tranzistoare) neconectate care constituie celula elementară a circuitului integrat. Cu o astfel de celulă, prin conectarea elementelor componente, se pot obţine mai multe tipuri de porţi logice care apoi, printr-o conectare corespunzătoare, pot forma circuitul logic dorit.

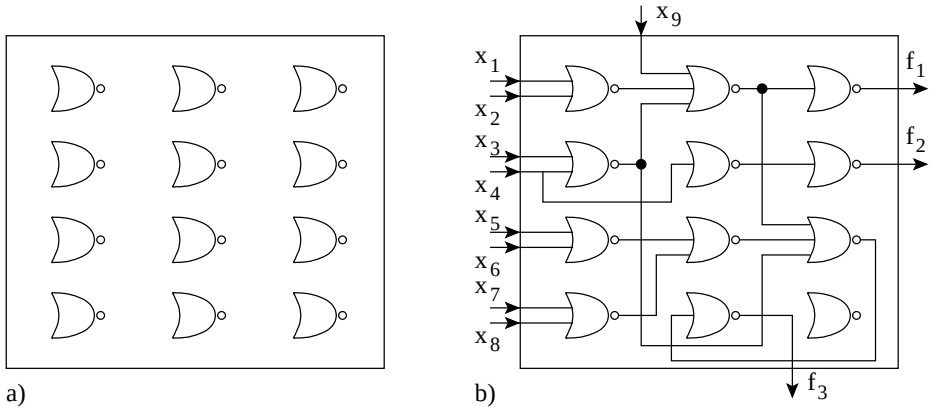


Figura 4.2 Circuitul arie de porţi logice: a) structură de principiu pentru o matrice cu porţi NOR; b) exemplu, posibil, de conectare a porţilor NOR pentru realizarea a trei funcţii logice.

Implementarea unei aplicaţii pe baza unui circuit arie de porţi logice constă în proiectarea interconexiunilor necesare între porţi/celule şi apoi la turnătoria de siliciu se realizează aceste interconexiuni, care necesită încă un număr de 2-3 măşti suplimentare. Costul proiectării măştilor pentru etapele de procesare ale circuitului (costul de tip NRE), până la nivelul de la care poate fi utilizat de către proiectantul aplicaţiei, se distribuie pe toate unităţile produse de turnătoria de siliciu, ca o investiţie iniţială. Oricum, acest număr de măşti suplimentare, cerut de aplicaţie, este mult mai mic decât numărul de măşti necesar aplicaţiei (peste 20) dacă aceasta ar fi realizată ca un produs full-custom. Tradiţional, proiectarea interconexiunilor se realizează pe coloanele şi rândurile dintre suprafeţele ocupate de celule plasate matriceal, denumite **canale de rutare**, operaţie care se realizează automat cu programe CAD adecvate.

În prezent circuitele arie de porţi logice, mai ales cele cu număr mare de porţi (zeci şi sute de mii), care pot conţine şi zone de memorie, nu mai prezintă canale de rutare, sunt referite ca **mare-de-porţi**. La un circuit arie de porţi logice de tip

mare-de-porţi interconexiunile sunt realizate pe straturile metalizate (3–4 straturi) situate deasupra ariei/matricei de porţi logice. **Selectarea porţilor** care vor fi utilizate pentru aplicaţie (**plasarea porţilor**) şi rutarea interconexiunilor se efectuează automat sau, foarte frecvent, în mod interactiv deoarece este necesar a se mări procentajul de porţi folosite şi de micşorare a interconexiunlor trasate manual (chiar şi un procent redus de porţi neutilizate şi de trasee duse manual reprezintă numere absolute destul de mari deoarece numărul total de porţi pe circuit poate fi de ordinul 10^4 sau 10^5); se poate ajunge uneori la un procentaj de porţi nefolosite de peste 50% din numărul total de porţi conţinute în circuit. Datorită procentului ridicat de porţi neutilizate, în circuitul final, şi a suprafeţelor mari consumate de pad-urile de intrare/ieşire suprafaţa unui circuit arie de porţi logice poate fi destul de mare; se pot consuma suprafeţe de $4 \div 5$ ori mai mari decât la realizarea aceleaşi aplicaţii sub formă de circuit full-custom. O proiectare de calitate se reflectă în circuitul rezultat prin neapariţia efectului de hazard static şi aceasta se obţine prin: egalizarea întârzierilor pe porţi (printr-o uniformizare a sarcinilor pe intrare şi a încărcărilor pe ieşire), egalizarea timpilor de propagare pe interconexiuni (printr-o lungime egală a traseelor).

Proiectarea aplicaţiilor pe arii de porţi se recomandă pentru serii care se realizează în volum de sute sau mii de unităţi şi este caracterizată de:

- timp de dezvoltare redus;
- automatizare ridicată a procesului de dezvoltare;
- performanţe (de viteză, de suprafaţă) relativ scăzute (în raport cu full-custom).

Exemplul 4.1 În Figura 4.3 este prezentată o structură (posibilă) de celulă pentru un circuit arie/matrice de porţi logice. Structura acestei celule se compune din două linii de difuzie, una n şi alta p, în care există respectiv câte patru tranzistoare nMOS şi pMOS neconectate, cu care se pot realiza patru dispozitive CMOS. Această structură mai conţine: patru bare din polisiliciu care sunt porţile comune pentru câte o pereche de tranzistoare complementare, traseele metalizate pentru alimentare V_{DD}, V_{SS}, precum şi ferestrele metalizate de acces (câte două) la fiecare tranzistor şi ferestrele metalizate din fiecare bară (poartă) de polisiliciu. În Figura 4.3-c sunt prezentate interconexiunile realizate pe celulă pentru implementarea unei porţi NOR4, Figura 4.3-b.

4.4 PROIECTAREA CU CELULE STANDARD

Proiectarea cu celule standard poate fi privită ca o extensie a proiectării pe bază de arii de porţi logice dar cu celule care nu mai sunt la nivel de tranzistor ci cu celule de nivel mai înalt, care deja realizează funcţii logice. Realizarea aplicaţiei pornind de la nivelul de tranzistor este consumatoare de timp de proiectare. Se poate reduce acest timp dacă se porneşte de la celule care pot fi porţi logice sau circuite care conţin mai multe porţi logice (multiplexoare, bistabile, sumatoare, comparatoare, etc). Aceste celule, denumite celule standard sau **policelule**, existente într-o bibliotecă de celule, sunt selectate de către proiectant în funcţie de aplicaţia de realizat, plasate şi interconectate între ele obţinându-se layout-ul viitorului circuit integrat. Elaborarea

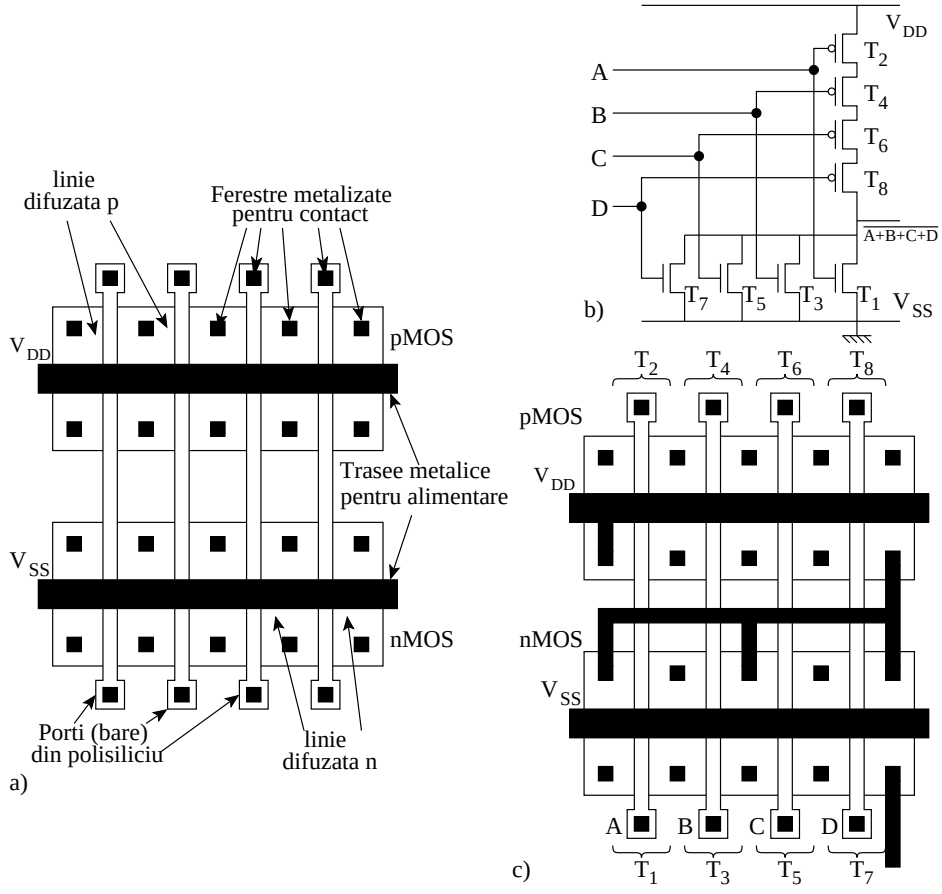


Figura 4.3 Explicativă pentru Exemplul 4.1: a) structură posibilă de celulă compusă din patru perechi de tranzistoare (pMOS şi nMOS) neconectate; c) conexiunile metalice necesare pe celulă pentru realizarea circuitului (NOR4) din figura b.

unei biblioteci de celule standard pentru o anumită tehnologie este o proiectare de tip full-custom şi necesită o activitate de ordinul zeci ingineri × luni (fiecare turnătorie, pentru o anumită tehnologie, pune la dispoziţia utilizatorilor o bibliotecă de celule standard).

În scopul uşurării operaţiei de rutare a conexiunilor se adoptă, în general, pentru fiecare celulă o aceeaşi înălţime (pentru a putea realiza rânduri de aceeaşi înălţime), lăţimea putând fi oricare în funcţie de dimensiunea circuitului celulei, iar intrările şi ieşirile se fixează numai pe laturile celulei spre care există acces din canalele de rutare. Canalele de rutare sunt pe suprafeţele dintre rânduri; la circuitul din Figura 4.4 celulele sunt plasate pe trei rânduri. Trecerile între canalele de rutare se face prin suprafeţele libere dintre celulele de pe aceaşi rând; uneori aceste treceri pot fi realizate şi prin suprafaţa celulei (orizontal sau vertical) dacă programul CAD de rutare are această facilitate. Comparativ cu proiectarea de tip full-custom la proiectarea cu celule standard suprafaţa consumată este de câteva ori mai mare dar este mai mică decât cea necesară pentru proiectarea de tip cu arie de porţi.

Spre deosebire de dezvoltarea pe bază de arie de porţi, unde sunt necesare doar 2–3 măşti pentru definitivarea aplicaţiei, o aplicaţie pe bază de celule standard necesită toate măştile, ca şi pentru full-custom; această asemănare face ca, uneori, acest tip de abordare de dezvoltare să fie referită ca **pseudo full-custom**. Totuşi, o reducere a costului iniţial se obţine faţă de full-custom, deoarece măştile pentru fiecare celulă standard există deja proiectate la turnătoria de siliciu. Dezvoltarea de tip celule standard se situează între cea cu arii de porţi şi cea full-custom; este recomandată la un volum de producţie de ordinul zeci de mii de unităţi.

Există şi variante de biblioteci de celule în care celulele pot fi de diferite înălţimi şi diferite forme; prin utilizarea unor astfel de celule se pot realiza layout-uri mult mai compacte, aproape de cele obţinute prin full-custom. Mai mult, cu astfel de celule, fiecare celulă poate fi considerată ca o componentă într-un bloc mai mare iar apoi aceste blocuri sunt tratate ca şi componente într-un bloc mai mare. De exemplu, în Figura 4.4-b, celulele A,B,C şi D, de dimensiuni diferite, sunt asamblate într-un bloc notat cu R (delimitat printr-un dreptunghi cu linie întreruptă), apoi acest bloc împreună cu blocurile R,S,T şi U (de dimensiune şi complexitate aproximativ de acelaşi nivel) sunt asamblate în blocul W. Rutarea pentru această **abordare ierarhizată**, în realizarea aplicaţiei, necesită un timp mai lung decât cea cu celule standard precum şi programe CAD mai performante, dar se poate obţine o compactare care nu depăşteşte cu mai mult de 20% suprafaţa unei proiectări full-custom.

În tehnologia de integrare un număr tot mai mare de etape, bazate pe mascare, sunt necesare pentru procesele de realizare a interconexiunilor în raport cu numărul proceselor de mascare pentru modificările de conductivitate din substrat. Interconexiunile sunt realizate pe baza unor straturi metalizate, ce pot fi mai mult de opt, referite ca metal 1, metal 2, etc., plasate succesiv deasupra substratului. De exemplu, în Figura 4.5 este prezentată o secţiune prin "stiva" de şase straturi metalizate pentru interconexiuni ale unui circuit integrat. Deasupra substratului pe un strat de dielectric se depune prin vaporizare chimică (Chemical Vapor Deposition) sau electrochimic (Electrochemical Deposition) primul strat metalic (Aluminiu, Cupru, Titanium, Tungstem sau diferite aliaje). Pe primul strat, metal 1, pe baza unui proces de mascare, se gravează reţeaua de trasee necesare pentru interconexiuni; peste acest strat metalic gravat se realizează un alt strat dielectric pe care se depune un al

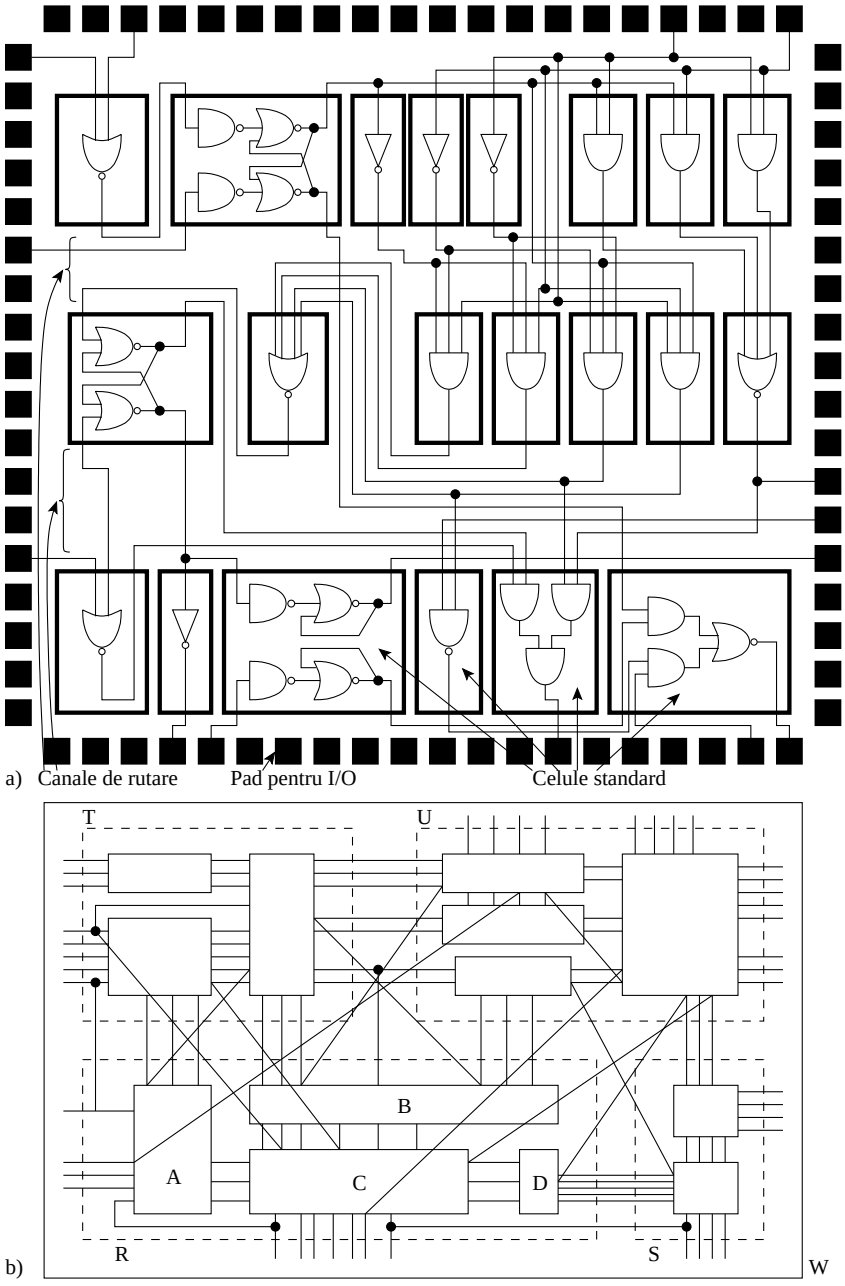


Figura 4.4 Proiectarea aplicaţiei pe bază de celule standard: a) exemplificarea plasării (pe trei rânduri de înălţimi egale) şi conectării unor celule standard; b) abordarea proiectării printr-o asamblare ierarhizată pe baza unor celule cu forme şi dimensiuni diferite.

doilea strat metalic, metal 2, în care se gravează interconexiunile pe care trebuie să le realizeze, se continuă în acest mod, alternanţă strat metalic–strat dielectric, până la ultimul strat metalic.

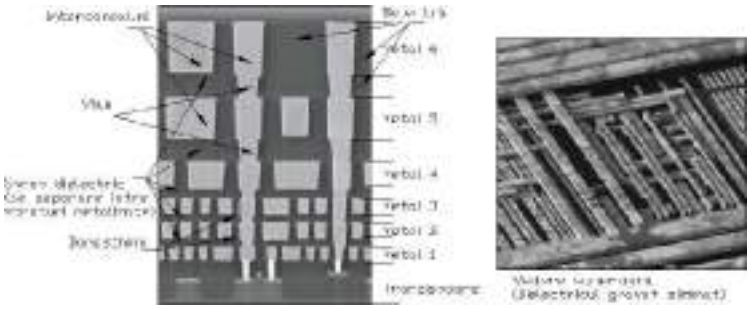


Figura 4.5 Interconexiunile metalice deasupra substratului unui circuit integrat: a) secţiune verticală prin straturile metalizate în care sunt interconexiunile; b)vedere superioară a interconexiunilor metalice din diferite straturi (după eliminarea, prin gravare, a masei de oxid de izolare dintre straturile metalice şi dintre trasee).

Pentru conectarea unor trasee, din două straturi metalice vecine, se realizează în stratul de dielectric dintre acestea, orificii de pătrundere, aceste pătrunderi sunt referite prin termenul de **via**. Via-surile dintr-un strat de dielectric sunt metalizate în acelaşi timp cu depunerea stratului metalic de deasupra stratului de dielectric. Peste zona de via, coaxial cu aceasta, trebuie să fie o zonă de interconectare, această pereche, via–zonă metalizată suprapusă este referită prin termenul de damaschină (amintind de procedeul vechi din giuvaergerie ori de inserarea de fire preţioase în ţesături). În secţiunea din figură, prin cele şase straturi metalizate, se observă două stive de damaschine care realizează o penetrare de la ultimul strat, metal 6, până la zonele de substrat (tranzistoare). Se observă că secţiunile traselor pentru interconexiuni cresc pe măsură ce sunt realizate într-un strat metalizat mai ridicat. Obişnuit, ultimele 3–4 straturi metalizate sunt utilizate pentru interconexiunile de alimentare (V_{DD}, V_{SS}) şi pentru conexiunile de lungime mare (globale, care străbat întreaga suprafaţă a cipului); straturile metalizate înspre substrat sunt utilizate pentru conexiuni scurte.

Performanţele de viteză, putere consumată şi de imunitate la diafonie şi la zgomot pentru dispozitivele programabile sunt puternic determinate de interconexiunile realizate. Metrica principală a unei interconexiuni este produsul RC — constanta de timp — unde R şi C sunt rezistenţa şi capacitatea interconexiunii respective. Performanţa de viteză a circuitului integrat este limitată de întârzierea RC a interconexiunilor; pe când timpii de propagare pe tranzistori se reduc progresiv odată cu scalarea, în timp ce reducerea lăţimii traselor de interconectare (R creşte) duce la creşterea întârzierea pe conexiuni. La o caracteristică de proces de $100nm$, timpul de propagare (la o comutaţie) pentru un tranzistor MOS este $5ps$ în timp ce întârzierea pe o interconexiune cu lungimea de $1mm$ este de $30ps$ (raportul întârzierilor este de $30ps/5ps = 6$); iar la o caracteristică de proces de $35nm$ raportul întârzierilor creşte la 100. În mod asemănător, raportul dintre energia consumată la comutaţia interconexiunii supra energia disipată la comutarea tranzistorului MOS este de 5 pentru caracteristica de

proces de $100nm$ şi creşte la valoarea de 30 pentru caracteristica de proces de $35nm$ [Lerouge '04]. Creşterea performanţelor circuitului pe baza îmbunătăţirii interconexiunilor se poate obţine prin micşorarea rezistenţei şi a capacităţii traseelor.

Reducerea rezistentţei interconexiunilor se poate obţine prin realizarea straturilor metalizate în cupru, înlocuirea straturilor metalizate din aluminiu cu cele de cupru micşorează rezistivitarea de la $\approx 3,3\mu\Omega \cdot cm$ la $\approx 1,8\mu\Omega \cdot cm$. De asemenea cuprul prezintă o mai bună comportare pentru efectul de electromigrare (a metalului) în raport cu aluminiul. Pentru tehnologiile cu caracteristica de proces începând cu $150nm$ sau $130nm$ se impune realizarea conexiunilor numai din cupru.

Reducerea capacităţii unei trase, atât valoarea totală cât şi valoarea capacităţii laterale/între liniile adiacente (care influenţează fenomenul de diafonie), se poate obţine prin micşorarea permitivităţii electrice, vezi relaţia 1.2, a dielectricului care izolează straturile metalice şi trasele metalice între ele. Permitivitatea electrică a obişnuitului izolator, bioxidul de siliciu florurat, este $\varepsilon \approx 3,6$ ($\varepsilon_0 = 8,85 \cdot 10^{-14} F/cm$ — permitivitatea electrică a vidului). Multe materiale pot avea o constantă electrică relativă între 1 şi 3,6 dar nu prezintă valori acceptabile care se impun pentru unele proprietăţi unui astfel de izolator: rezistenţă mecanică, absorbţie de umiditate, interacţiune chimică (spălare/gravare în procesul de fotolitografie, adeziune pentru depunere de metal), conductivitate termică şi tensiune electrică de străpungere. Cerinţele enumerate pot fi realizate de materiale numite CDO (Carbon Doped Oxide) şi care sunt potrivite pentru caracteristici de proces până la $90nm$. Pentru caracteristici de proces sub această valoare se impun materiale cu o permitivitate electrică relativă sub 2,2; o astfel de valoare poate fi realizată numai de materiale izolatoarea cu structură poroasă.

4.5 PROIECTAREA CU DISPOZITIVE LOGICE PROGRAMABILE COMPLEXE, CPLD

Dispozitivele logice programabile simple, **SPLD**, prezentate în secţiunea 2.4.7, sub forma unui circuit PLA, prezintă flexibilitate maximă deoarece pot fi programate pe matricea AND cât şi pe matricea OR. Deşi acestea posedă o astfel de flexibilitate, în practică, sunt mai frecvente circuitele de tip PAL care au numai nivelul AND programabil iar nivelul OR este restricţionat la un număr fix de intrări şi fără posibilitatea de a fi programat. În scopul "ameliorării" acestei restricţionări fiecare poartă OR a fost inclusă într-un circuit, referit în general prin termenul de **macrocelulă**, Figura 2.59, obţinându-se circuitul de tip GAL, care prezintă unele facilităţi pe intrare/ieşire printre care şi faptul că un pin al circuitului poate fi programat fie ca pin de intrare fie ca pin de ieşire. Astfel, au devenit clasice circuitele GAL16V8 şi GAL20V8 care sunt încapsulate cu pini pe două rânduri, capsulă **DIP** (**D**ual **I**nline **P**in) având numai respectiv 20 de pini (posibile 16 intrări şi 8 ieşiri) şi 24 pini (posibile 20 de intrări şi 8 ieşiri).

Pentru implementarea unui sistem de dimensiune mare un circuit PLD de tipul $16V8$ ori $20V8$, uneori, nu este satisfăcător din punct de vedere al resurselor (intrări, ieşiri, termeni AND, macrocelule), iar rezolvarea ar impune: 1- utilizarea mai multor circuite SPLD, ca şi componente discrete, conectate pe o placă de circuit imprimat;

2- realizarea unui circuit PLD integrat care să aibă resursele multiplicate de n ori. Prima soluţie poate fi practică cu anumite inconveniente (dimensiune, putere disipată, viteză) pe când a doua nu este o soluţie viabilă. De exemplu, circuitul de tip $16V8$ multiplicat de $n = 16$ ar fi un circuit $256V128$; acesta presupune 128 de intrări şi 128 intrări/ieşiri (macrocelule), dar aceasta ar avea următoarele inconveniente [Wakerley ′00]. În primul rând timpii de propagare ar creşte de cel puţin 8 ori (o poartă AND care are 512 intrări, 256 intrări negate şi 256 intrări nenegate, nu poate fi realizată pe un singur nivel de propagare). În al doilea rând, suprafaţa pe siliciu ar fi de 256 ori (n^2) mai mare în raport cu cea a unui circuit $16V8$.

Inconvenientele anterioare, prezentate de cele două soluţii, pot fi eliminate prin mixarea celor două soluţii: realizarea formei de pe placa de circuit imprimat dar sub formă integrată. Această soluţie constă în realizarea pe aceeaşi plachetă de siliciu a n circuite simple PLD, iar pe liniile şi coloanele dintre acestea există segmente de trasee care, prin programare, pot fi interconectate pentru realizarea aplicaţiei, Figura 4.6, obţinându-se astfel performanţe de viteză, dimensiune şi putere mai bune decât la cele două soluţii anterioare. Un astfel de circuit logic programabil care prezintă resurse pentru implmentarea sistemelor de dimensiune foarte mare este referit prin abreviaţia **CPLD** (**C**omplex **PLD**); un circuit CPLD poate implementa sisteme care ar necesita un număr de porţi de ordinul mii-zeci de mii.

Ideea arhitecturală, prezentată în Figura 4.6, pentru un CPLD este realizată sub diferite abordări de către fiecare firmă de componente, existând în prezent zeci de tipuri de circuite CPLD. Diferenţa între aceste tipuri constă în modalităţile specifice

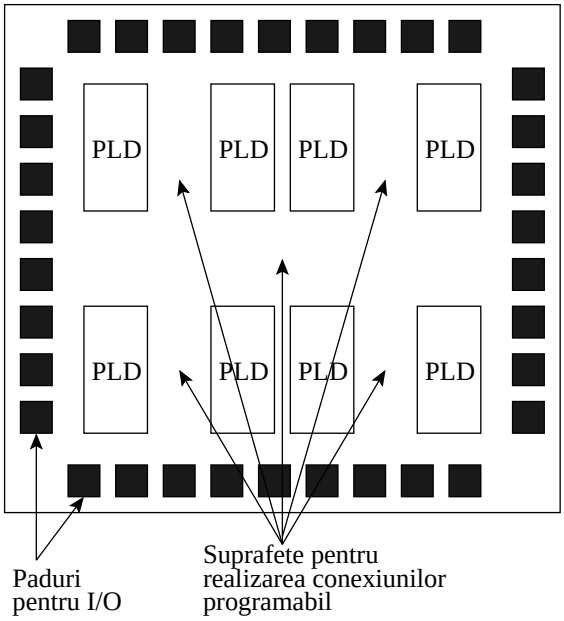


Figura 4.6 Organizarea de principiu pentru un circuit logic programabil complex, CPLD

prin care se realizează matricea programabilă AND, macrocelula, interconexiunile programabile, blocul de intrare/ieşire şi tensiunile de interfaţare cu exteriorul. Pentru exemplificarea organizării şi a resurselor disponibile la un CPLD, dar din punctul de vedere al proiectantului de aplicaţie, în continuare se va prezenta circuitul EPM7512B din seria MAX7000B (Multiple Array matriX) a firmei ALTERA. Seria MAX7000B prin scalare (pe baza aceloraşi componente, cu celule de programare a interconexiunilor de tip CMOS EEPROM, cu tensiunea de alimentare internă $V_{CCINT} = 2,5V$) realizează cele cinci circuite CPLD din Tabelul 4.1.

Tabelul 4.1 Seria de circuite CPLD, MAX7000B, ale firmei ALTERA

Caracteristica	Tipul de circuit				
	EPM7032B	EPM7064B	EPM7128B	EPM7256B	EPM7512B
Numărul de porţi utilizabile	600	1250	2500	5000	10000
Numărul total de macrocelule	32	64	128	256	512
Numărul de blocuri logice	2	4	8	16	32
Numărul de pini I/O	36	68	100	164	212
f_{CNT} [MHz]*	303,0	303,0	243,9	188,7	163,9

*frecvenţa de ceas internă globală de valoare maximă

Un circuit poate suporta reprogramarea celulelor CMOS EEPROM până la o sută de ori.

Circuitele EPM7xxx au o structurare de principiu prezentată în Figura 4.7-a care este detaliată în Figura 4.7-b. Scalarea fiecărui circuit din serie se bazează pe utilizarea următoarelor cinci elemente componente:

1. Blocul matriceal logic, **LAB** (**L**ogic **A**rray **B**lock);

2. Macrocelula;

3. Expandorul de termeni produs (cu alocare distribuită, cu alocare paralelă);

4. Matricea de interconexiuni programabile, **PIA** (**P**rogrammable **I**nterconnect **A**rray);

5. Blocul de intrare/ieşire (I/O Control Block) la care prin intermediul a patru intrări speciale se poate aplica oricărei macrocelule sau oricărui pin I/O fie patru semnale de intrare de utilitate generală, INT, fie semnale globale pentru control ce pot fi: semnale globale de ceas $GCLK1$, $GCLK2$; semnalul global de ştergere, $GCLR_n$; semnale globale de validare, $OE1$, $OE2$.

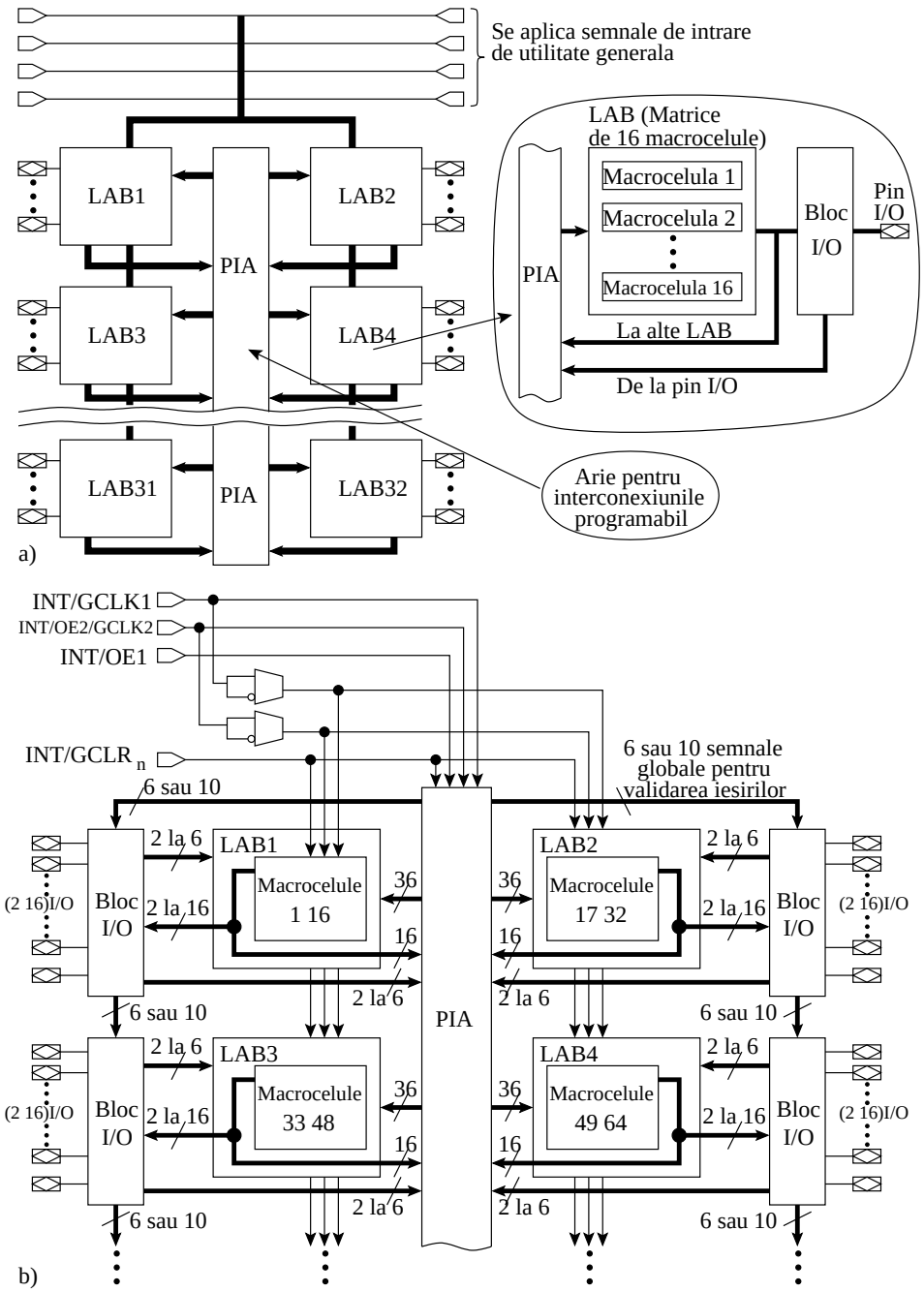


Figura 4.7 Seria MAX7000B (ALTERA): a) structurare de principiu a circuitelor CPLD din cadrul seriei cu detalierea pentru un bloc matriceal logic, LAB (în medalion); b) detaliere de structurare doar a primelor patru LAB-uri cu evidenţierea semnalelor corespunzătoare.

Blocul matriceal logic, LAB este o matrice de 16 macrocelule (vezi Figura 4.7-a medalion); în seria MAX7000 un circuit poate conţine 2, 4, 8, 16 sau 32 de LAB-uri. Fiecărui LAB i se aplică următoarele semnale:

- 36 de semnale de la PIA, care sunt variabilele de intrare în matricea AND;
- 3 semnale globale de control (utilizabile pentru operaţiile la registru din componenţa macrocelulelor);
- 2 până la 6 semnale direct de la pinii I/O (aplicabile direct la registrul din componenţa macrocelulelor pentru cazurile când se elimină intrările prin intermediul PIA).

Macrocelula, Figura 4.8-a, este compusă dintr-o matrice programabilă AND, o matrice pentru selectarea termenilor produs şi un registru programabil (bistabil cu facilităţi extinse); fiecare macrocelulă dintr-un LAB poate fi configurată individual pentru o operaţie logică combinaţională sau secvenţială.(Aici noţiunea de macrocelă este mai cuprinzătoare decât cea care a fost introdusă la circuitul GAL, Figura 2.57) Pot fi generaţi cinci termeni produs, de maximum 32 de variabile, care prin intermediul matricei de selectare sunt alocaţi la intrarea unei porţi OR, pentru a obţine o sumă de produse (ori o sumă de produse negată la ieşirea porţii XOR) sau cei termeni produs sunt aplicaţi la registru ca semnale de: PRESET, CLEAR, CE (validare ceas). Funcţia logică (sumă de produse) poate fi aplicată la ieşire (prin intermediul blocului I/O), fie după ce a fost stocată într-un registru (sincronizată) sau fie fără stocare în registru (nesincronizată).

Registrul macrocelulei suportă programare individuală pentru a realiza următoarele funcţionări de bistabil: D, T, JK şi SR. De asemenea se poate programa (prin intermediul unui multiplexor) ca semnal de ceas (de sincronizare) al bistabilului să fie unul din următoarele semnale:

- semnalul global de ceas $GCLK1$ sau $GCLK2$ (negate sau nenegate, vezi Figura 4.7-b);
- semnalul global de ceas $GCLK1$ sau $GCLK2$ dar ca semnale de validat (pe intrarea CE, de un termen produs în cadrul macrocelulei);
- de un termen produs în cadrul macrocelulei.

Semnalul de înscriere asincronă, $PRESET$, este unul din cei cinci termeni produs calculaţi în macrocelulă. La fel, şi semnalul de ştergere asincronă $CLEAR$ este unul din cei cinci termeni produs dar, în plus, ştergerea poate fi realizată şi prin semnalul general de ştergere $GCLR_n$.

Expandorul de termeni produs. Pentru cazul când cei cinci termeni produs, calculaţi în macrocelulă, nu sunt suficienţi pentru numărul necesar de termeni produs într-o sumă logică, se poate extinde calculul sumei de produse prin "împrumutarea" de termeni produs calculaţi şi în alte macrocelule; această extindere se poate realiza în două modalităţi. În primul rând, printr-o expandare cu alocare prin distribuţie: al cincelea termen calculat pe fiecare macrocelulă este negat şi aplicat înapoi ca o coloană în matricea programabilă AND (astfel apar încă 16 termeni produs în matricea AND, câte unul provenit de la fiecare macrocelulă). Termenul aplicat înapoi, când este

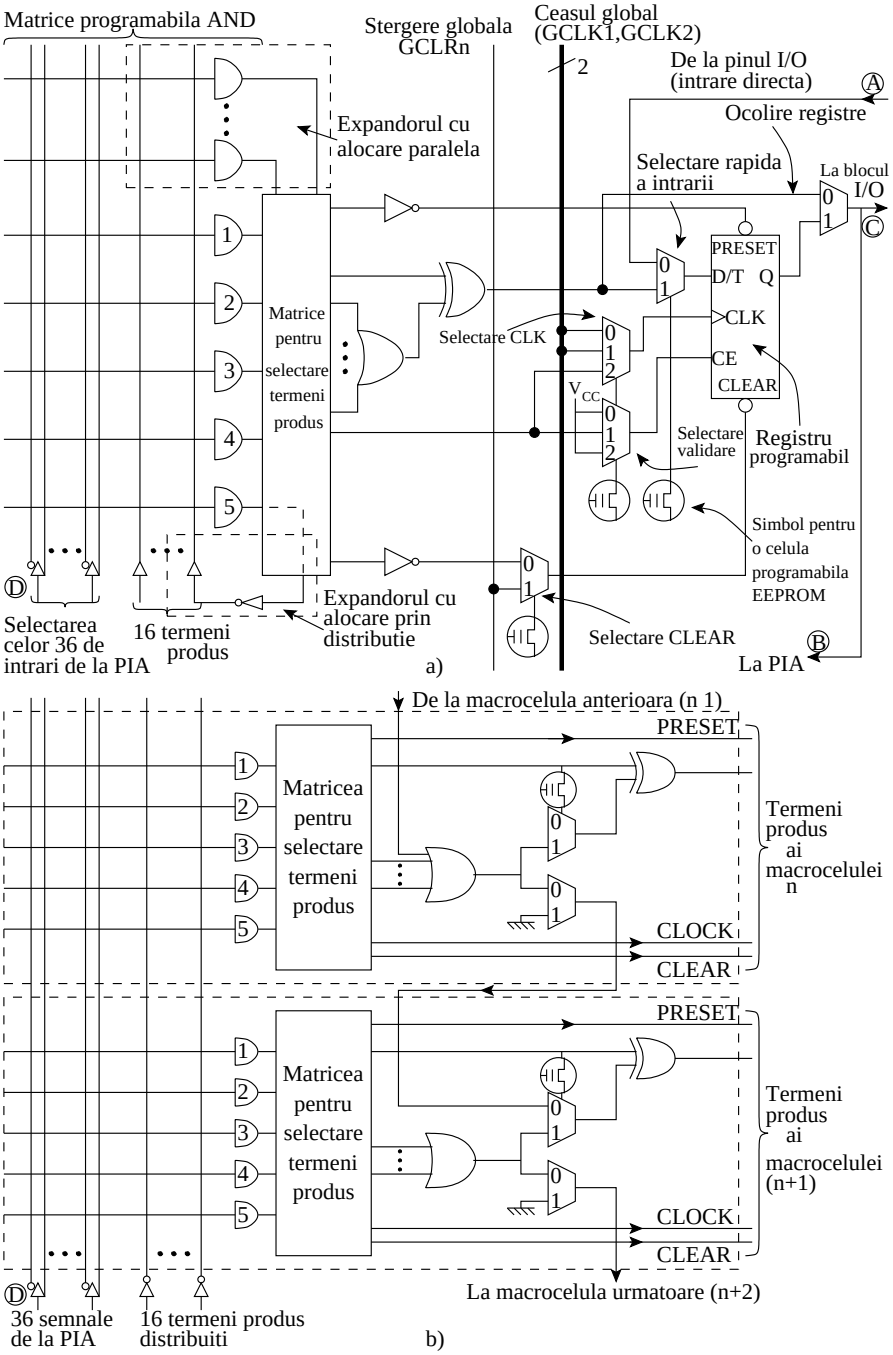


Figura 4.8 Seria MAX7000B - Macrocelula: a) structura unei macrocelule; b) modul de expandare cu alocare paralelă a termenilor produs între două macrocelule vecine.

selectat într-o sumă de produse, va introduce, evident, o întârziere suplimentară în calculul funcţiei respective. În al doilea rând, printr-o expandare cu alocare paralelă când funcţiile calculate de la maximum trei celule vecine anterioare sunt însumate cu funcţia de la macrocelula curentă, Figura 4.8-b; se pot obţine astfel sume de maximum 20 de termeni produs. De exemplu, pentru a realiza o sumă de produse de 19 termeni produs pe macrocelula 5 se procedează în felul următor: se colectează pe macrocelula 2 patru termeni produs, aceştia se colectează pe poarta OR din macrocelula 3 cu cinci termeni produs, apoi se colectează pe poarta OR din macrocelula 4 cu încă cinci termeni produs şi în final prin colectarea şi a celor cinci termeni produs pe poarta OR din macrocelula 5 se obţine funcţia dorită $(4+5+5+5=19)$. Evident, prin această alocare se introduce în calculul funcţiei o întârziere suplimentară egală cu întârzierea de propagare prin 3 porţi OR.

Blocul de intrare/ieşire, I/O, Figura 4.9-a. Cu acest bloc se poate configura individual ca fiecare pin I/O să poată fi utilizat ca un pin de intrare, ca un pin de ieşire sau ca un pin cu transfer bidirecţional. Pinul I/O este un terminal de intrare când driverul de ieşire, de tip TSL, este în starea HZ (semnalul de validare ieşire driver este în starea L) şi este un terminal de ieşire când driverul are o funcţionare normală (semnalul de validare ieşire driver este în starea H). Semnalul de validare ieşire driver poate fi selectat dintre: V_{CC}, masă şi un set de 6 sau 10 semnale globale de validare. Setul de semnale globale de validare se obţin (negate sau nenegate) de la PIA şi care au fost aplicate ca semnale de intrare pe diferite intrări ale circuitului sau ca semnale de la ieşirile unor alte macrocelule. Alimentarea driverului de ieşire se face la valori ale tensiunii de intrare/ieşire $V_{CCO} = 3,3V$; $2,2V$; $1,8V$ care sunt tensiuni ale standardului sub ale cărui specificaţii se face comunicaţia între pinul I/O al circuitului CPLD şi circuitele exterioare plasate pe aceeaşi placă de circuit imprimat, vezi Exemplul 4.4 (circuitul MAX7000B în interior şi bufferul de intrare sunt alimentate la tensiunea $V_{CCINT} = 2,5V$, care poate fi diferită de tensiunea V_{CCO}). Când pinul I/O operează ca pin de intrare există toleranţă pentru tensiunile exterioare aplicate pe intrare de $3,3V$ $2,5V$ şi $1,8V$. Dar când alimentarea driverului se face la tensiunea de intrare/ieşire de $V_{CCO} = 3,3V$ atunci nivelul H al driverului de ieşire este de $3,3V$, prin urmare există compatibilitate la conectarea în exterior şi cu circuite care sunt alimentate la standardul de $5V$.

Driverul de ieşire poate fi programat pentru o funcţionare de tipul cu drenul în gol (util pentru legarea pinului într-o conexiune SI cablat (vezi secţiunea 1.4.3)). De asemenea, driverul poate fi programat pentru modificarea valorii pantei fronturilor de creştere şi descreştere (slew-rate) ale semnalelor generate la ieşire. O pantă de valoare ridicată a semnalelor de ieşire creşte performanţele de viteză dar creşte de asemenea puterea disipată şi poate introduce zgomote, pe când o pantă redusă duce la o întârziere mai mare în circuit , o putere consumată micşorată şi eventual evitarea zgomotelor.

În regim de pin de intrare, semnalul de la ieşirea bufferului de intrare, se poate direcţiona spre o macrocelulă, fie prin aplicarea directă la bistabilul macrocelulei (calea notată cu (A)), fie prin intermediul PIA (calea notată cu (B)). Pe calea de aplicare directă la registru se poate selecta introducerea unei întârzieri programabile. Prin această întârziere variabilă se poate ajusta timpul de prestabilire, τ_{SU}, la bistabil; se poate adapta momentul aplicării semnalului pe intrarea D/T a bistabilului în raport cu aplicarea semnalului global de ceas (de sincronizare) care prezintă o anumită

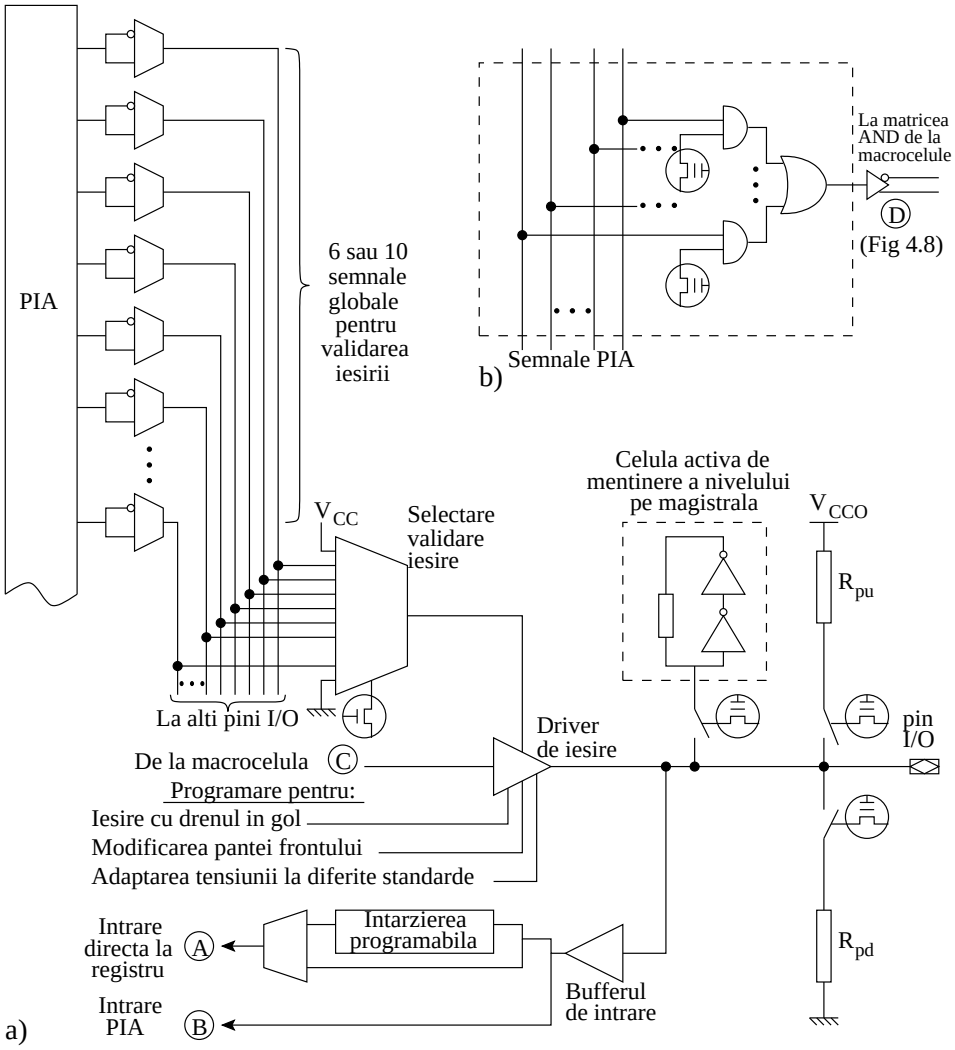


Figura 4.9 Seria MAX7000B: a) structura unui bloc de intrare/ieşire I/O; b) modul de programare, pe bază de celule EEPROM, a conexiunilor de la PIA către LAB-uri.

întârziere introdusă de bufferele prin care se propagă.

Pentru posibilitatea de conectare a pinului I/O la magistrale de tip TSL în blocul I/O sunt prevăzute rezistenţele R_{pv} şi R_{pd} de conectare respectiv la V_{CCO} şi masă, conectarea acestora poate fi programată; opţional poate exista o celulă activă de menţinere a nivelului (vezi Figura 1.46-e), care poate substitui cele două rezistenţe.

Matricea de interconexiuni programabile, PIA. Această matrice de interconexiuni programabile poate fi privită ca o magistrală la care sunt conectate toate intrările dedicate, toţi pinii I/O ai circuitului şi ieşirile de la macrocelule şi prin care se poate realiza, prin programare, conectarea oricărui semnal sursă la oricare destinaţie. În Figura 4.9-b este prezentat modul de programare pentru rutarea semnalelor din PIA la LAB (Macrocelule).

Dezvoltarea unei aplicaţii, pe baza unui circuit CPLD, are în succesiune aceleaşi etape indiferent de producătorul circuitului, diferenţele care apar sunt datorate suportului de dezvoltare şi performanţele acestui suport. Un sistem de dezvoltare pentru aplicaţii se realizează în jurul unui PC sau a unei staţii de lucru dotate cu un mediu software ce conţine module specifice pentru fiecare etapă a elaborării. Se porneşte de la o idee (izvorâtă dintr-o cerinţă sau dictată de piaţă) care este translatată într-o arhitectură şi anumite specificaţii electrice. Această arhitectură, eventual împărţită în blocuri, este introdusă în calculator fie **schematic** (sub formă de desen), fie **textual** (exprimare într-un limbaj de descriere) şi compilată corespunzător. Apoi urmează etapele de sinteză logică, de plasare şi rutare (pe circuitul CPLD utilizat) şi de verificare (simulare şi analiza timpilor obţinuţi); pe fiecare din aceste etape sau între aceste etape existând iteraţii şi, în final, generarea programului pentru configurarea circuitului CPLD, adică programarea/(înscrierea) celulelor de tip EEPROM. După înscrierea în CPLD a celulelor EEPROM (utilizând un suport hardware de programare–**programator**) cu ajutorul programului de configurare, se va citi starea rezultată a tuturor celulelor EEPROM pentru a se verifica, prin comparaţie, dacă configuraţia rezultată este identică cu cea din programul de configurare. Înscrierea/(încărcarea) CPLD-ului se poate face fie când acesta nu este încă inclus în sistem, fie când este deja introdus în sisteme, deci inclus pe placa de circuit imprimat (ultima variantă se utilizează curent pentru modificarea sau actualizarea unui sistem deja în funcţiune). Deoarece un CPLD este un circuit cu sute de pini, iar alocarea funcţiunilor pe fiecare pin este flexibilă (adică un pin poate fi alocat pentru un anumit semnal al circuitului MAX7000B), este necesar şi un program pentru proiectarea traseelor pe placa de circuit imprimat.

4.6 PROIECTAREA CU MATRICE LOGICE PROGRAMABILE LA UTILIZATOR, FPGA

Matricea logică programabilă de utilizator, **FPGA** (**F**ield **P**rogrammable **G**ate **A**rray) poate fi privită ca o sinteză (de succes) între circuitul arie de porţi logice şi circuitul CPLD. De la CPLD s-a "împrumutat" blocul logic dar nu de aşa mare dimensiune, în schimb, aceste blocuri logice de dimensiuni mai reduse, care constituie un suport programabil pentru implementare de funcţii logice, sunt în număr mult mai mare decât cele cuprinse într-un CPLD. Şi, similar, ca la circuitul arie de porţi,

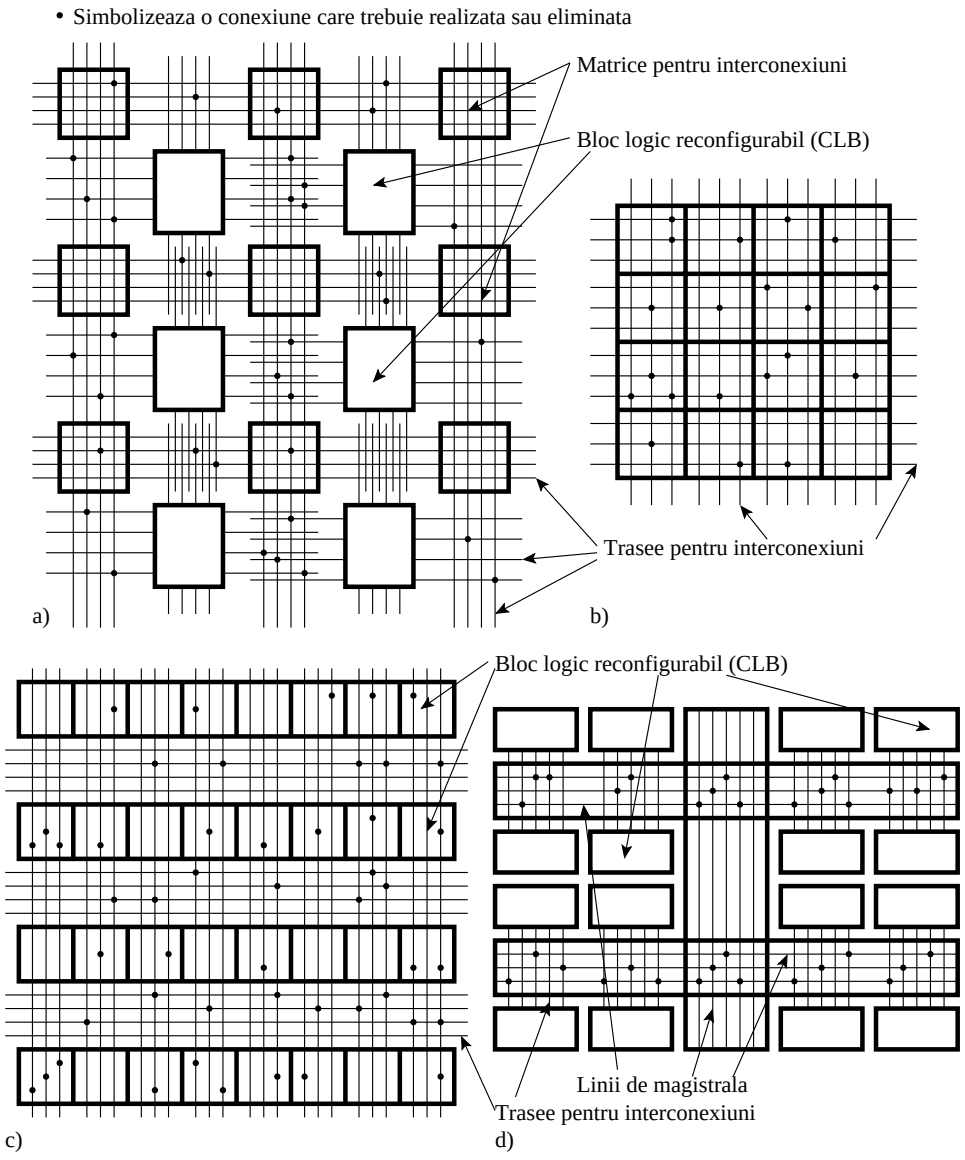


Figura 4.10 Modalităţi de structurare pentru circuitele FPGA: a) structură matriceală (cu canale de rutare); b) structură tip mare de CLB-uri; c) structurare cu CLB-uri în linie; d) structurare pe bază de magistrală (pentru interconectări).

blocurile logice ale unui circuit FPGA sunt plasate în diferite topologii pe suprafaţa cipului, care vor fi interconectate de către proiectantul aplicaţiei. Dar, spre deosebire de aria de porţi logice, la FPGA există deja prefabricate, împreună cu blocurile logice, anumite de segmente de trasee dintre care unele vor fi interconectate conform proiectării aplicaţiei. Şi ca durata implementării să fie scurtată, eliminând timpul de realizare al măştilor la turnătoria de siliciu impusă de o implementare pe circuitul arie de porţi logice, care poate fi de ordinul săptămânilor, pentru realizarea interconexiunilor la FPGA se evită programarea prin mascare. Suportul fizic al programării interconexiunilor poate fi: fuzibil sau antifuzibil (care sunt de tip o singură dată programabil – OTP), (E)EPROM (care are un număr de înscrieri de maximum $\times 10^3$), SRAM, Figura 4.1 (care necesită înscriere la fiecare punere sub tensiune a circuitului); se poate obţine astfel un timp de realizare a aplicaţiei de ordinul orelor sau zilelor. Evident, pentru conectarea cu exteriorul, circuitul FPGA mai trebuie să curpindă blocuri de intrare/ieşire, pentru fiecare pin I/O, la fel ca la CPLD.

Pentru **blocurile configurabile**, **CLB** (**C**onfigurable **L**ogic **B**lock), împreună cu segmentele (liniile) de trasee pentru interconectare, în cadrul circuitelor FPGA se pot identifica, mai frecvent, următoarele patru structurări:

1. Structură matriceală cu canale de rutare, Figura 4.10-a; similar ca la aria de porţi logice, Figura 4.2, porţile logice fiind substituite cu CLB (segmentele de interconectare fiind realizate pe liniile şi coloanele dintre CLB, iar la intersecţia acestora există matrice de interconectare pogramabile).

2. Structură pe bază de mare-de-celule CLB, Figura 4.10-b, similară ariei cu mare de porţi logice neconectate.

3. Structură pe bază de linii/coloane de celule CLB, Figura 4.10-c, similară structurilor cu celule standard, Figura 4.4.

4. Structură pe bază de magistrală, Figura 4.10-c, când CLB-urile sunt conectate între ele prin intermediul unei magistrale, similar ca la unele circuite CPLD, vezi PIA în Figura 4.7-a.

4.6.1 Blocul Logic Configurabil

Un bloc logic configurabil, CLB, poate fi realizat pe bază de tranzistoare, pe bază de porţi logice mai simple sau mai complexe, multiplexoare sau tabele de căutare, **LUT** (**L**ook-**U**p-**T**able, vezi secţiunea 2.4.6); s-au impus realizările pe bază de multiplexoare şi mai ales cele pe bază de LUT. Un CLB conţine două părţi un generator de funcţii (partea combinaţională) şi partea secvenţială (un registru care poate avea funcţiunile normale ale tipurilor de bistabile plus alte facilităţi suplimentare).

CLB pe bază de MUX. Un generator de funcţii de n variabile se poate realiza cu un circuit MUX $2^n : 1$, secţiunea 2.4.4.1; variabilele funcţiei se aplică pe cele n intrări de selectare iar coeficienţii funcţiei pe cele 2^n intrări de date, pentru fiecare configuraţie i a cuvântului format din coeficienţii funcţiei se obţine una din cele 2^{2^n} funcţii, f_i^n, $0 \leq i \leq 2^{2^n} - 1$. Structuri de generatoare, pe bază de multiplexoare, utilizate de firma ACTEL, sunt prezentate în Figura 4.11. Generatorul de funcţii din Figura 4.11-a, care are opt intrări şi o ieşire, f, poate realiza, utilizând unele intrări şi sub forma negată, următoarele funcţii: cele patru funcţii logice curente (NAND,

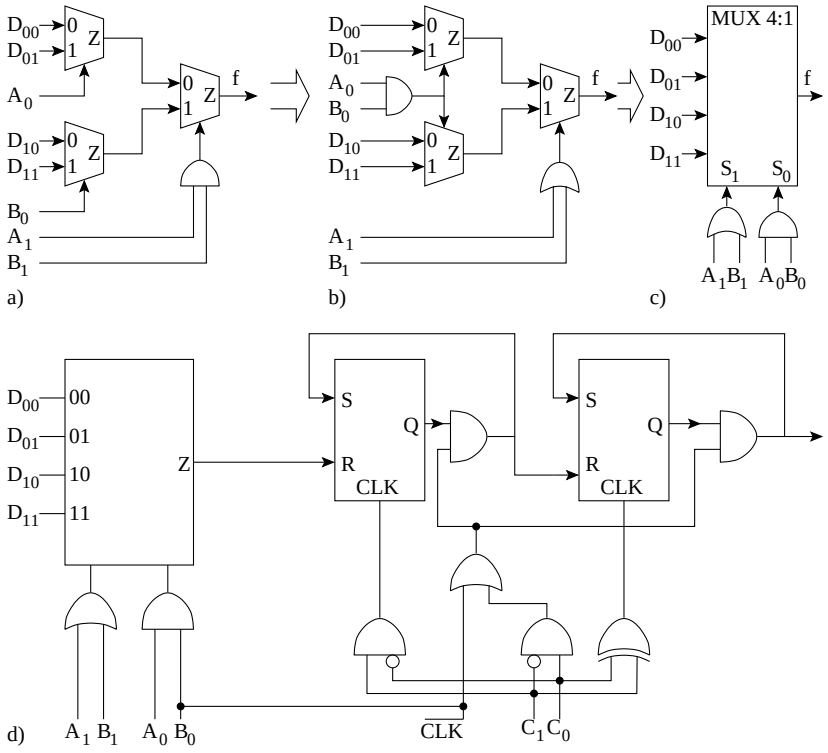


Figura 4.11 CLB pe bază de multiplexoare: a,b,c) generatoare de funcţii pe bază de multiplexoare; d) structură de CLB cu generator de funcţii pe bază de multiplexor iar partea secvenţială realizabilă în jurul a două latch-uri

NOR, AND, OR cu 2, 3 şi 4 intrări), XOR, XOR-OR-OR, XOR-OR-AND, OR-XOR, AND-XOR-OR, AND-OR, OR-AND etc. O variantă îmbunătăţită a generatorului de funcţii este cea din Figura 4.11-b care este echivalentă unui MUX4:1 din Figura 4.11-c pe care se pot implementa un număr de 766 funcţii distincte. Expresia logică a funcţiei f este:

$$f = \left[D_{00}(\overline{A_0B_0}) + D_{01}(A_0B_0)\right](\overline{A_1 + B_1}) + \left[\overline{D_{10}}(\overline{A_0B_0}) + D_{11}(A_0B_0)\right](A_1 + B_1)$$

Ataşând acestui generator de funcţii o parte secvenţială, compusă din două latch-uri şi câteva porţi logice, se obţine o structură pentru un CBL pe bază de multiplexor, Figura 4.11-d.

Partea secvenţială poate fi utilizată ca un bistabil D cu comutaţie pe frontul pozitiv sau negativ sau ca latch transparent pe palierul H sau pe palierul L dacă la intrările C_1 şi C_0 se aplică semnalul de ceas, CLK, "1", "0" în diferite combinaţii logice. De exemplu, pentru conexiunea C_0 ="0" şi $C_1 = CLK$ funcţionarea este de bistabil D cu comutaţie pe frontul pozitiv. Bistabil de tip JK sau SR se poate obţine prin utilizarea mai multor CLB şi cu conexiuni externe pentru reacţie. Partea secvenţială prin reconfigurări pe baza unor porţi existente în structura CLB, sau

prin utilizarea generatorului de funcţii, poate genera facilităţi care nu sunt comune bistabilelor curente. CLB-urile bazate pe multiplexoare produc o flexibilitate mare în generarea funcţiilor pentru un număr de tranzistoare relativ redus, în schimb din cauza unui număr mare de intrări apare o mare cerere asupra resurselor de rutare pentru realizarea interconexiunilor.

Exemplul 4.2

Pe baza unui bistabil de tip D se vor prezenta câteva structuri cu facilităţi neîntâlnite la bistabilul SR sau la bistabilul D. Ecuaţia de funcţionare a bistabilului SR este $Q(t+1) = S + \overline{R}Q(t)$, Tabelul 3.4, iar a bistabilului D este $Q(t+1) = D(t)$, relaţia 3.20.

- Figura 4.12-a, **S–dominant**; pentru ambele intrări de date activate, $SR = 11$, bistabilul SR se va înscrie în $Q(t+1) = 1$ (la bistabilul obişnuit această combinaţie produce nedeterminare).
- Figura 4.12-b, **R–dominant**; pentru ambele intrări de date activate, $SR = 11$, bistabilul SR se va înscrie în $Q(t+1) = 0$.
- Figura 4.12-c, **fără modificări**; pentru ambele intrări de date activate, $SR = 11$, bistabilul SR nu îşi modifică starea $Q(t+1) = Q(t)$.
- Figura 4.12-d, **basculează**; pentru ambele intrări de date activate, $SR = 11$, bistabilul SR basculează în starea opusă, similar bistabilului JK, adică o funcţionare de basculă (bistabil T).
- Figura 4.12-e, **bistabilul D cu MUX 2:1 pe intrare**; ieşirea $Q(t+1)$ va avea valoarea uneia din valorile de intrare, D_0 sau D_1, în funcţie de valoarea 0 sau 1 a variabilei de control C.
- Figura 4.12-f, **bistabil D cu validare a semnalului de ceas**, Figura 3.72-c, se va înscrie în bistabil intrarea D_{in}, $Q(t+1) = D_{in}(t)$, numai când semnalul de ceas este validat de $CE = 1$, altfel $Q(t+1) = Q(t)$ (vezi Figura 3.45-e).

CLB pe bază de LUT. Blocurile logice configurabile pe bază de LUT utilizează o memorie SRAM pentru stocarea valorilor coeficienţilor funcţiei, valori care se înscriu odată cu configurarea aplicaţiei în circuitul FPGA. O memorie SRAM de capacitate 2^n adrese $\times$ 1 bit poate stoca valorile funcţiei f_j^n, $0 \leq j \leq 2^{2^n} - 1$, (vezi secţiunea 1.1.3 şi 2.1); în fiecare locaţie, de adresă determinată de configuraţia cuvântului format din valorile variabilei funcţiei $x_{n-1}x_{n-2}\ldots x_1x_0$, se află stocat coeficientul d_{jm} al funcţiei ($d_{jm}\epsilon\{0,1\}$, $0 \leq m \leq 2^n - 1$). De exemplu, în LUT-ul din Figura 4.13-a, pentru valorile variabilelor de intrare $x_3x_2x_1x_0 = 1010 = 11|_{10}$, la locaţia de adresă 11 se află valoarea $f0$; în LUT este înscrisă funcţia de patru variabile f_{4636}^4. În general, pentru FPGA, numărul de variabile pentru funcţiile implementabile este între 4 şi 6; toate funcţiile au acelaşi timp de propagare care este timpul de citire al memoriei SRAM.

O structură simplificată de CLB, care conţine trei celule LUT, este prezentată în Figura 4.13-b (obţinută prin simplificarea structurii de CLB din seria XC4000 a firmei Xilinx). Două generatoare de funcţii G' şi F', realizate respectiv pe LUT_1 şi LUT_2, pot implementa oricare funcţie de patru variabile independente $G_1 \div G_4$ sau $F_1 \div F_4$. Al treilea generator H', realizat pe LUT_3, poate implementa oricare funcţie de trei variabile respectiv: F', G' (fiecare poate fi o funcţie de maxim patru variabile) şi variabila H_1 care este aplicată din exterior. Semnalele de la ieşirile generatoarelor de funcţii pot fi obţinute la două ieşiri ale CLB: F' şi H' pot fi conectate la ieşirea X prin intermediul MUX_4, iar G', H' pot fi conectate la ieşirea Y prin intermediul MUX_3. Se pot obţine pe ieşirile X şi Y: două funcţii de cel mult patru variabile independente, o singură funcţie de cinci variablile independente sau alte funcţii de

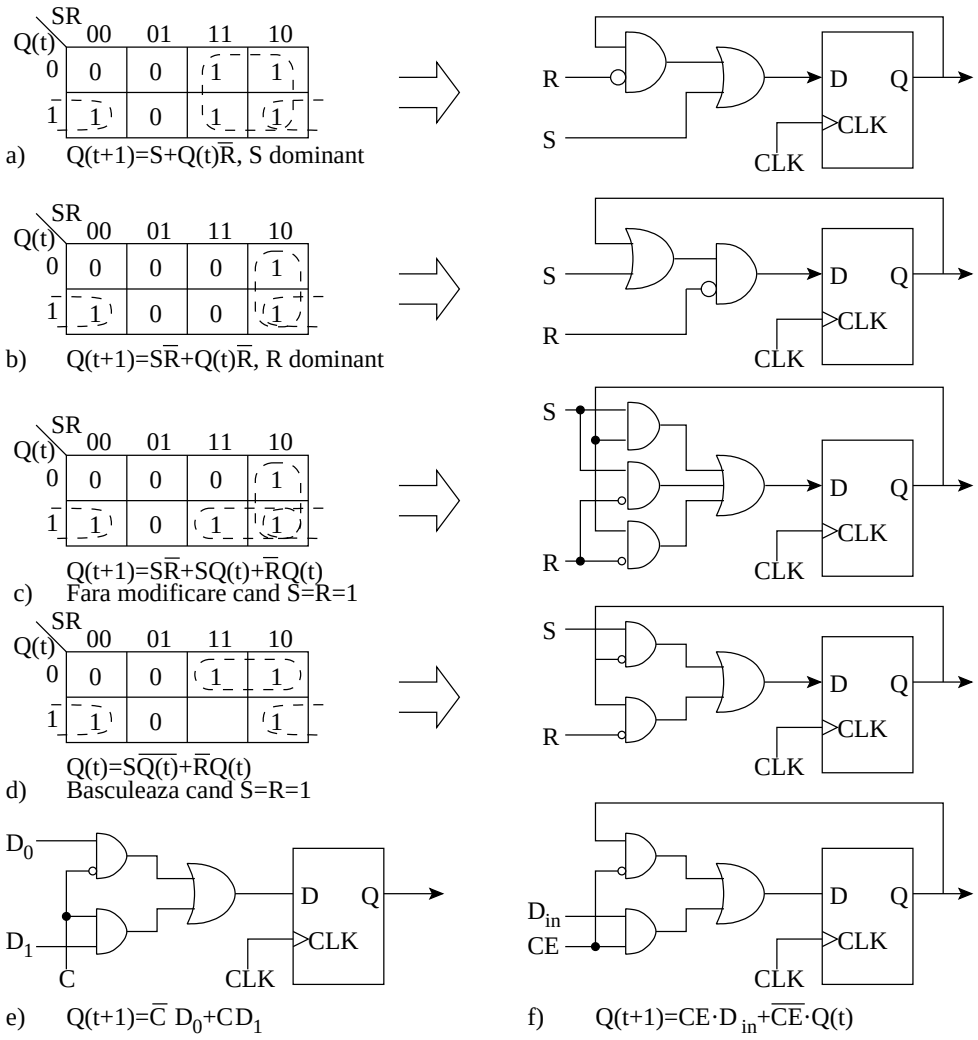


Figura 4.12 Posibile configuraţii pentru un bistabil D dintr-un CLB prin care se pot obţine următoarele facilităţi în funcţionare: a)pentru $SR = 11$, S este dominant; b) pentru $SR = 11$, R este dominant; c) pentru $SR = 11$, fără modificări; d) pentru $SR = 11$, basculează (bistabil T); e) bistabil D cu MUX 2:1 pe intrare; f) bistabil D cu validare a semnalului de ceas prin $CE = 1$.

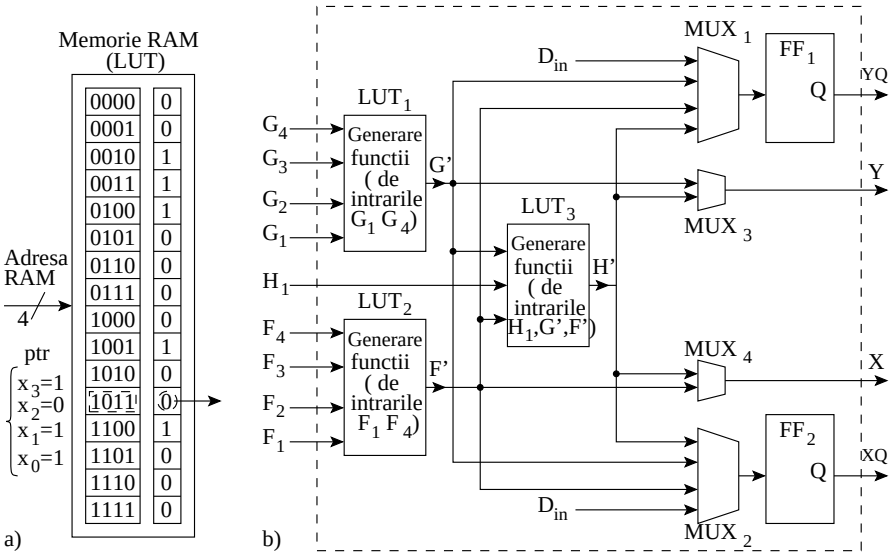


Figura 4.13 CLB pe bază de LUT: a) exemplu de generator de funcţie de 4 variabile implementat pe un SRAM de capacitate 2^4 adrese $\times$ 1 bit; b) structură de CLB pe baza a 3 celule LUT şi două bistabile, cu două ieşiri directe (X, Y) şi două ieşiri sincronizate (XQ, YQ).

până la nouă variabile.

Partea secvenţială se realizează în jurul a două bistabile FF_1 şi FF_2 care, în general, pot fi programate independent pentru: înscriere sincronă sau asincronă (pe intrări de tip $\overline{PRESET}$ şi $\overline{CLEAR}$); sincronizare prin diferite semnale de ceas; posibilitate de activare a semnalelor de ceas prin semnal de validare, CE. În structura din figură, intrările de date ale bistabilelor pot fi selectate, prin MUX_1 şi MUX_2, dintre funcţiile F', G', H' sau D_{in}, un semnal direct de la o intrare (specială) în CLB. Ieşirile YQ şi XQ sunt ieşiri sincronizate din CLB.

Exemplul 4.3 Automatul cu tabelul de tranziţie al starilor dat în Figura 4.14-a să se implementeze pe CLB-uri care au o structură ca cea prezentată în Figura 4.14-b.

Soluţie: Adoptând codificarea stărilor $A \to 001$, $B \to 101$, $C \to 010$, $D \to 110$, $E \to 011$ se obţine tabelul stărilor pentru funcţiile de execitaţie w_2, w_1, w_0 şi pentru ieşirea Z (de tip Mealy):

$$\begin{aligned} w_2 &= \overline{x}_1\overline{x}_0 z_1 z_0 + \overline{x}_1 x_0 \overline{z}_1 z_0 + x_1 x_0 z_1 \overline{z}_0 \\ w_1 &= \overline{x}_1\overline{x}_0 + \overline{x}_1 x_0 \overline{z}_2 z_1 + x_1 x_0 + x_1 \overline{x}_0 \overline{z}_2 z_1 \\ w_0 &= \overline{x}_1\overline{x}_0 \overline{z}_2 z_0 + \overline{x}_1 x_0 + x_1 \overline{x}_0 \\ Z &= \overline{z}_2 z_1 + x_0 \end{aligned}$$

Cu aceste expresii se calculează valorile funcţiilor w_2, w_1, w_0 şi Z (respectiv în tabelele din Figura 4.14-c,d,e,f) care se încarcă în LUT. Sunt necesare trei celule LUT cu patru intrări şi una cu trei intrări, deci se utilizează două CLB-uri, Figura 4.14-g. Ieşirea Z se poate obţine atât sincronizată (Mealy cu întârziere) cât şi nesincronizată (Mealy imediat).

a)

Starea prez.	Starea urmatoare/iesire, Intrari x_1x_0			
	00	01	11	10
A	E/0	B/1	C/1	A/0
B	C/0	B/1	C/1	A/0
C	C/1	E/1	D/1	E/1
D	C/0	A/1	D/1	A/0
E	E/1	E/1	C/1	E/1

b)

Starea prezenta			Starea urmatoare/iesire, Intrari x_1x_0			
z_2	z_1	z_0	00	01	11	10
0	0	1	011/0	101/1	010/1	001/0
1	0	1	010/0	101/1	010/1	001/0
0	1	0	010/1	011/1	110/1	011/1
1	1	0	010/0	001/1	110/1	001/0
0	1	1	011/1	011/1	010/1	011/1

c) CLB_1/LUT_1

x_1	x_0	z_1	z_0	w_2
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

d) CLB_1/LUT_2

x_1	x_0	z_2	z_1	w_1
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
0	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

e) CLB_2/LUT_1

x_1	x_0	z_2	z_0	w_0
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

f) CLB_2/LUT_2

x_0	z_2	z_1	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

g)

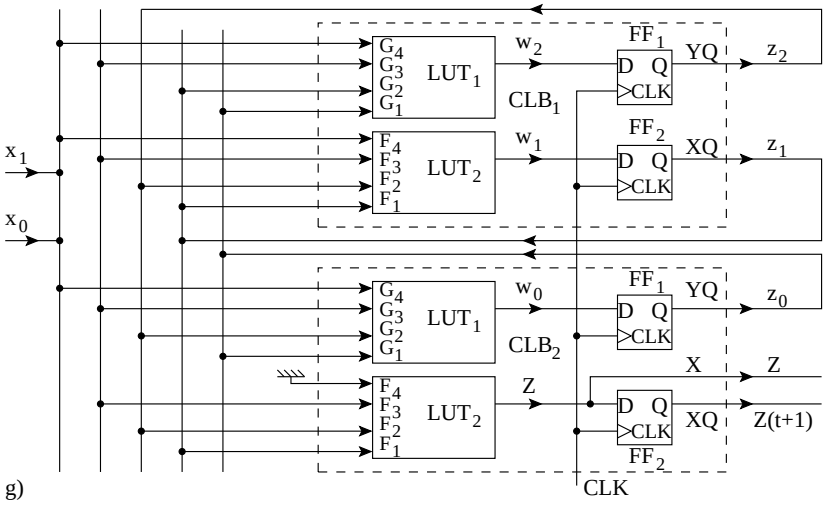


Figura 4.14 Explicativă pentru Exemplul 4.3

Blocul de intrare/ieşire, I/O. Blocul I/O constituie o interfaţă între pinii exteriori şi CLB. Aceste blocuri sunt situate în exteriorul suprafeţei matricei formate din CLB-uri, Figura 4.15-a. Structura si funcţiile unui bloc I/O al unui circuit FPGA sunt aceleşi cu cele ale unui bloc I/O al unui CPLD, care a fost prezentat în secţiunea 4.5, Figura 4.9-a. Mai nou, blocului I/O i se adaugă sî următoarele două facilităţi: 1-controlul digital al impedanţei driverilor de ieşire şi adaptarea impedanţelorla intrare; 2-controlul fluxului de date pe intrare sau pe ieşire la rata dubla, DDR, secţiunea 3.6.3. Structura şi funcţionarea unui astfel de driver I/O va fi prezentat în Exemplul 4.4.

4.6.2 Resursele de interconectare

Pentru o organizare de tip matriceal, Figura 4.10-a, a unui circuit FPGA topologia plasării componentelor pe cip este prezentată în Figura 4.15-a. Pe întresuprafeţele (orizontale şi verticale) dintre CLB-uri sunt plasate segmente de linii metalizate (de diferite lungimi) precum şi elemente de conectare programabile. Suprafaţa matricei de CLB-uri este înconjurată de blocuri I/O, fiecare dintre aceste blocuri putând fi conectat la un pin I/O (la un pad). În jurul blocurilor I/O (înspre paduri) există segmente de linii metalizate care, prin interconectări programabile, permit conectarea unui bloc I/O la oricare pin al circuitului; aceasta permite o proiectare a aplicaţiei fără restricţionări impuse de poziţia pinilor (deci, un circuit FPGA poate fi programat pentru un cablaj imprimat deja existent).

Resursa de interconectare a unui FPGA este constitită din segmentele de linii metalizate care, prin intermediul punctelor de interconectare programabile, pot fi constituite în trasee de transfer pentru semnale între intrările, ieşirile CLB-urilor şi blocurile I/O. Circuitele FPGA din punct de vedere al suportului de interconectare pot fi cu o arhitectură de rutare segmentată sau nesegmentată. Arhitectura nesegmentată oferă lungimi fixe pentru segmentele metalice de rutare, care străbat întreaga lungime a cipului, ceea ce poate constitui o uşurare în realizarea softului de plasare şi rutare. În schimb, arhitectura segmentată conţine linii metalizate de diferite lungimi putându-se astfel să se realizeze o rutare optimă de la logic la logic; testele au arătat că o arhitectură segmentată oferă posibilitatea de a împacheta mai multă logică şi la performanţe mai ridicate decât cea nesegmentată. Performanţa unui traseu este determinat de timpul de propagare care este o întârziere de tip RC, întârziere care este puternic influenţată de punctele de interconectare programabile (evident, circuitele cu trasee realizate prin mascare au performanţă de viteză superioare în raport cu cele cu interconexiuni programabile). Actual, tehnologiile de rutare cu conexiuni programabile se realizează cu aşa numitele **interconexiuni active** (bufferate), adică pe segmentele metalizate la fiecare punt de interconectare se introduce un buffer, prin acesta se elimină întârzierea de rutare variabilă, dependentă de factorul de încărcare (fan-out) al semnalului.

Pentru concretizarea resurselor de interconectare se va prezenta în continuare suportul oferit de circuitele din seria XC4000 (Xilinx). În Figura 4.15-a (medalion) sunt prezentate toate segmentele de linii metalizate de interconectare prin care, potenţial, un CLB poate fi conectat sau poate să se conecteze la alte CLB-uri sau blocuri I/O, detalierea la nivel de puncte de interconectare programabile este dată în Figura 4.15-b

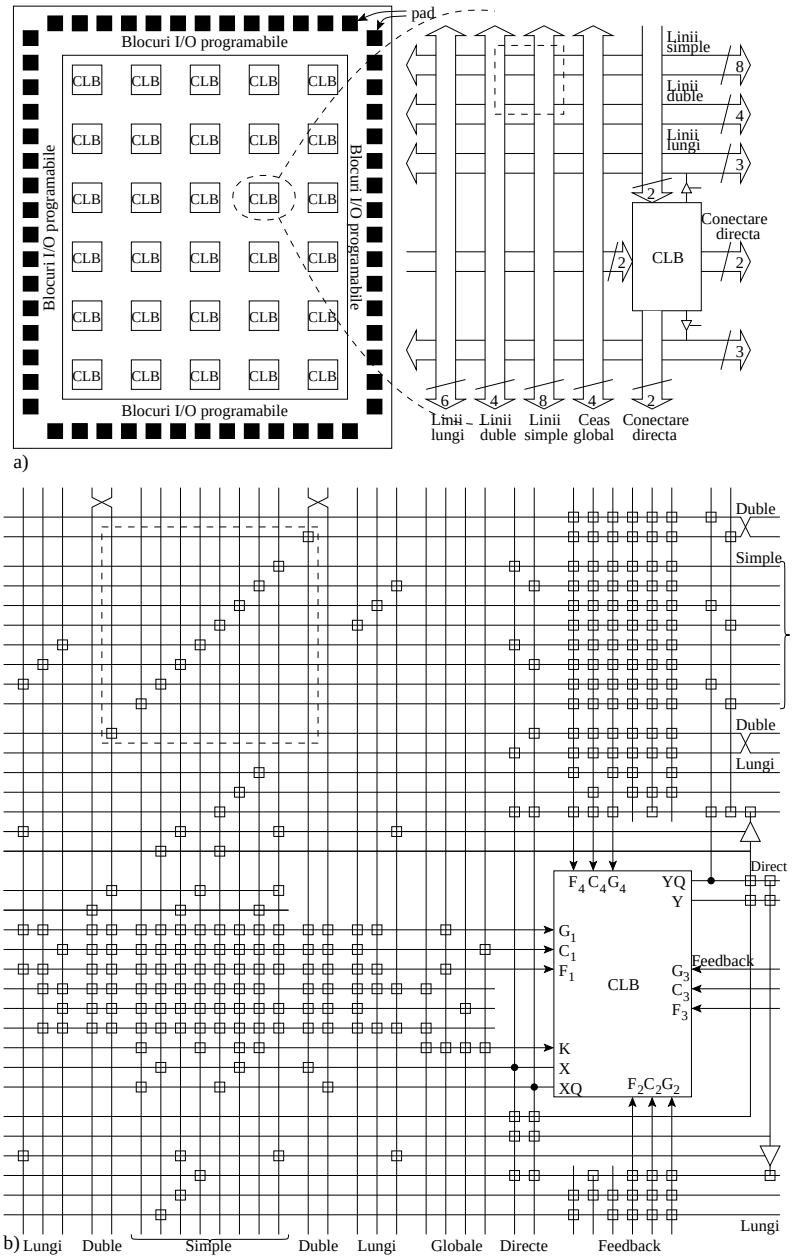


Figura 4.15 Resursele de interconectare pentru seria XC400 (Xilinx): a) plasamentul componentelor pe cip cu reprezentarea bloc a liniilor de interconectare în jurul unui CLB (în medalion); b) detalierea interconectărilor programabile în jurul unui CLB.

(la intersecţia dinte două linii dacă există punct de interconectare acesta s-a figurat cu un mic pătrat).

Există următoarele grupe de linii de interconectare:

1. Linii de interconectare directă. Fiecare CLB, din matrice, are două ieşiri care se conectează direct la celula următoare din dreapta şi la celula de dedesubt. De asemenea, celula este conectată direct la grupul de trei linii de deasupra, de dedesubt (verticale) şi în partea stânga (orizontale).

2. Linii de o lungime (linii simple). Aceste grupuri de câte 8 linii de o lungime formează o reţea pe orizontală şi vericală în jurul unui CLB, si se intersectează între ele printr-o matrice de interconectare, (conturul pătratic trasat cu linie întreruptă din Figura 4.15-b şi Figura 4.15-a medalion). Aceste linii simple asigură în primul rând, o conectare flexibilă între CLB-uri adiacente în afara puţinelor linii de conectare directe şi, în plus, faţă de acestea sunt suport pentru un transfer bidirecţional. Structura unei matrice de interconectare este prezentată în Figura 4.16-a. Fiecare punct de interconectare al matricei constă dintr-o reţea de şase tranzistoare de trecere (sau porţi de transmisie), Figura 4.16-b, a căror comandă este aplicată de la şase latch-uri care constituie memoria de configurare (neevidenţiată—ascunsă—în structura pentru un CLB prezentată în Figura 4.13-b şi în Figura 4.15-a), simbolizată prin pătratele mici cu litera M. Un semnal aplicat pe un traseu care intră prin dreapta în matricea de interconexiuni poate fi rutat la o linie simplă fie în direcţia sus, jos sau stânga sau in toate cele trei directîi (în funcţie de biţii M înscrişi în latch-urile memoriei de configurare).

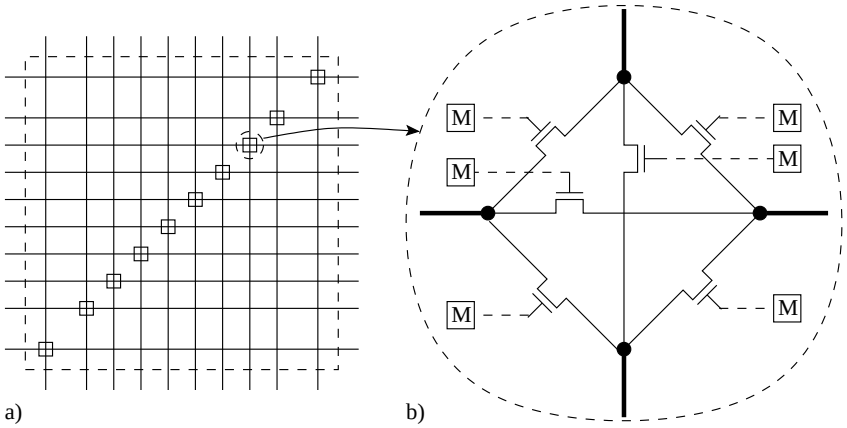


Figura 4.16 Resursele de interconectare pentru seria XC4000 (Xilinx): a) structura unei matrice de interconectare (pentru linii simple şi linii duble); b) structura unui nod nod al matricei (M reprezintă latch-urile din memoria de configurare).

3. Linii de două lungimi (linii duble). Aceste grupuri, la fel ca şi liniile simple, formează o reţea pe orizontală şi pe verticală în jurul unui CLB, dar fiecare linie dintre aceste grupuri intră tot la a doua matrice de interconectare, deci se pot conecta CLB-uri din doi în doi. Pentru trasee mai lungi, liniile duble asigură o întârziere mai

mică decât o configurare cu linii simple. Unele circuite FPGA au şi linii de lungime cvadruplă sau sixtuplă.

4. Linii lungi. Acest grup de linii (verticale şi orizontale) sunt pe toată dimensiunea matricei de CLB-uri şi nu trec prin matrice de interconectare. Utilitatea principală a acestora este de a fi folosite pentru semnale de fan-out mare, cu timp de propagare critic (fără defazaj) sau reţele de distribuţie la nivelul întregului circuit; pot fi comandate cu drivere TSL în apropiere de CLB-uri.

5. Linii globale. Sunt utilizate pentru distribuţia fără defazaj a semnalelor de ceas; realizările actuale inculd circuite pentru compensarea defazajului de ceas (clock skew), vezi secţiunea 3.5.7.

Fluxul dezvoltării unei aplicaţii pe un circuit FPGA este destul de asemănător cu cel pentru un circuit CPLD. Ponind de la o idee se elaborează o arhitectură şi anumite specificaţii electrice, apoi etapele sunt: descrierea proiectului, sinteza, implementarea şi configurarea circuitului FPGA.

Descrierea proiectului-sinteză se poate realiza schematic (desene) sau printr-un limbaj de descriere, iar rezultatul rulării etapelor de descriere şi sinteză este un fişier într-un format specific circuitului FPGA care va fi suportul pentru aplicaţie.

Implementarea se compune din următoarele trei faze: partiţionarea, plasarea şi rutarea. **Partiţionarea** constă în împărţirea proiectului pe blocuri care să poată fi mapate în CLB-uri. **Plasarea** constă în alocarea acestor blocuri la anumite elemente fizice ale circuitului FPGA (CLB-uri, blocuri I/O).

Rutarea este faza prin care elementele fizice fixate sunt conectate prin intermendiul liniilor de interconectare. Determinarea traseelor optime prin joncţionarea de segmente de linii de interconectare este un proces iterativ (mai dificil dacât printr-o programare prin mascare). Fazele de partiţionare, plasare şi rutare sunt realizate automat prin soft, însă utilizatorul poate impune constrângeri specifice. Rezultatul etapei de inplementare este un şir de biţi, de ordinul 10^5 biţi (bitstream).

Configurarea constă în transmiterea (înscrierea) şirului de biţi obţinut în etapa anterioară, în memoria de configurare. Fiecare latch, Figura 4.1-a (simbolizat prin litera M în Figura 4.16-b), al memoriei de configurare, distribuită pe întreaga suprafaţă a cipului, va fi înscris prin şirul de biţi cu 1 sau 0 logic după cum este necesar să comande în starea activă sau inactivă punctul programabil, la care este conectat, din blocurile I/O, din CLB-uri sau din reţeaua de interconectare. În interiorul unui CLB punctele configurabile sunt pentru: biţii din LUT-urile generatoarelor de funcţii, biţii de selectare ai multiplexoarelor. Acest şir poate fi trimis din calculator direct în memoria de configurare a circuitului FPGA, deja montat pe placa de circuit imprimat a aplicaţiei, sau este înscris într-o memorie PROM serială ce se va alătura circuitului FPGA de pe placa de circuit inprimat (FPGA-ul trebuie configurat la fiecare punere sub tensiune!).

Modalitatea de programare la un circuit FPGA este, esenţial, aceeaşi ca la programarea unui calculator prin stocarea programului în memoria principală. Dezvoltarea sau testarea unei noi aplicaţii poate fi realizată tot atât de uşor şi rapid ca şi pentru o aplicaţie software. Deci, se poate spune că FPGA-ul a adus hardul la acelaşi nivel de programabilitate ca şi softul. Reţelele logice realizate pe FPGA au performanţe de viteză mai reduse de până la două sau trei ordine de mărime în raport cu implementarea de tip full-custom, dar, în schimb, duc la performanţe mai ridicată cu câteva ordine de mărime în raport cu simularea în soft a aceleaşi reţele logice.

Exemplul 4.4 Acest exemplu are scopul de a completa noţiunile de bază referitoare la circuitul FPGA prezentate anteriorn cu noile facilităţi existente la circuitele actuale. Aceste noi facilităţi vor fi exemplificate la **circuitele FPGA ale familiei VIRTEX-II PRO** (**Xilinx**) în raport cu circuitul FPGA XC4000. Circuitele VIRTEX-II PRO care corespund tehnologiei elaborate în anul 2000 faţă de cele XC4000 (care aparţin tehnologiei anilor ′90) au fost proiectate astfel încât să fie o platformă pentru realizarea unor sisteme programabile, depăşind astfel nivelul numai de logică programabilă. Cu aceste noi circuite **se trece de la nivelul de logică programabilă la nivelul de sistem programabil**. Aplicaţiile complexe pot fi eficient repartizate între o implementare de tip logic (hardware) pentru a obţine o viteză ridicată şi o implementare de tip software pentru a obţine o flexibilitate ridicată. Această platformă VIRTEX-II PRO conţine în acelaşi timp trei componente:

A: **Un nucleu de microprocesor** (PowerPC 405)

B: **transievere** (transmitter-reciever, vezi Figura 3.82-f). Utilizate pentru interconectări între magistrale, la motherboard sau la alte sisteme printr-o conexiune paralel–serie respectiv paralel–serie. Fiecare dintre cele (până) la 24 canale de transmisie permite o rată de transfer a datelor de la 622Mb/s până la 3,125Gb/s

C: **FPGA**, care va fi analizat în continuare

1. CLB.

În VIRTEX-II PRO blocurile logice configurabile, CLB, au o organizare tip matriceal (vezi Figura 4.15-a) şi sunt suport pentru proiectările de tip combinaţional şi/sau secvenţial. Fiecare CLB posedă o proprie matrice de comutaţie prin care poate accesa la căile generale de rutare, Figura 4.16-a. Un CLB, spre deosebire de cele din familia XC4000, este compus din patru secţiuni (slice) identice S_1, S_2, S_3, S_4 conectate la propria matrice de comutaţie, precum şi la o conexiune directă (rapidă) prin care se conectează cu CLB-urile vecine, Figura 4.17-a. Cele patru secţiuni sunt organizate pe două coloane, fiecare coloană având câte un un circuit independent pentru propagarea transportului (C_{in}, C_{out}) dar un singur circuit comun (ambelor coloane) pentru deplasare (Shift). Există de asemenea, două buffere TSL care au fiecare pin propriu pentru intrare şi pentru controlului regimului TSL, ieşirile acestor buffere, TBUF, comandă resurse de rutare orizontale pentru realizarea de magistrale TSL pe cip. Fiecare dintre cele patru secţiuni ale unui CLB are acces, prin intermediul matricei de comutaţie, la intrarea şi/sau la controlul celor două buffere TSL. În fiecare secţiune S_i, $i = 1, 2, 3, 4$, a CLB-ului sunt incluse două generatoare de funcţii de patru variabile, două elemente de stocare, un circuit pentru propagarea transportului, porţi pentru funcţii aritmetice şi o mutlitudine de multiplexoare. Numărul total de secţiuni pentru membrul cel mai de jos al familiei, XC2VP2, este de 1408, iar pentru membrul cel mai dezvoltat este de 55616. În Figura 4.17-b este prezentată o schemă simplificată pentru structura unei secţiuni (de exemplu, unele simboluri de MUX indică faptul că asupra semnalului se realizează o funcţie de selectare fară a specifica toate intrările de date şi de selectare).

- **Generatoarele de funcţii** G şi F sunt de tipul LUT cu patru intrări ($G_4G_3G_2G_1$ şi $F_4F_3F_2F_1$). Se poate implementa orice funcţie de patru variabile iar utilizând anumite multiplexoare se poate extinde până la funcţii de 9 variabile de intrare. Semnelele de ieşire din generatoarele de funcţii pot fi obţinute la ieşirea secţiunii respective (X, Y) sau: pot fi aplicate pentru calcul aritmetic (intrare la poarta XOR); aplicate în lanţul de propagarea transportului; aplicate la intrarea elemenelor de stocare.

-**Elementele de stocare** pot fi configurate fie ca două bistabile D cu comutaţie pe front, fie ca două latch-uri (active pe palier). Pe intrarea D semnalele directe X, Y, de la ieşirea generatoarelor de funcţii, via ieşirile DX, DY sau semnalele BX, BY care intră direct în secţiune, ocolind generatorul de funcţii. Pentru ambele elemente de stocare ale unei secţiuni sunt comune următoarele semnale de control: CLK - ceasul; CE - validare ceas;

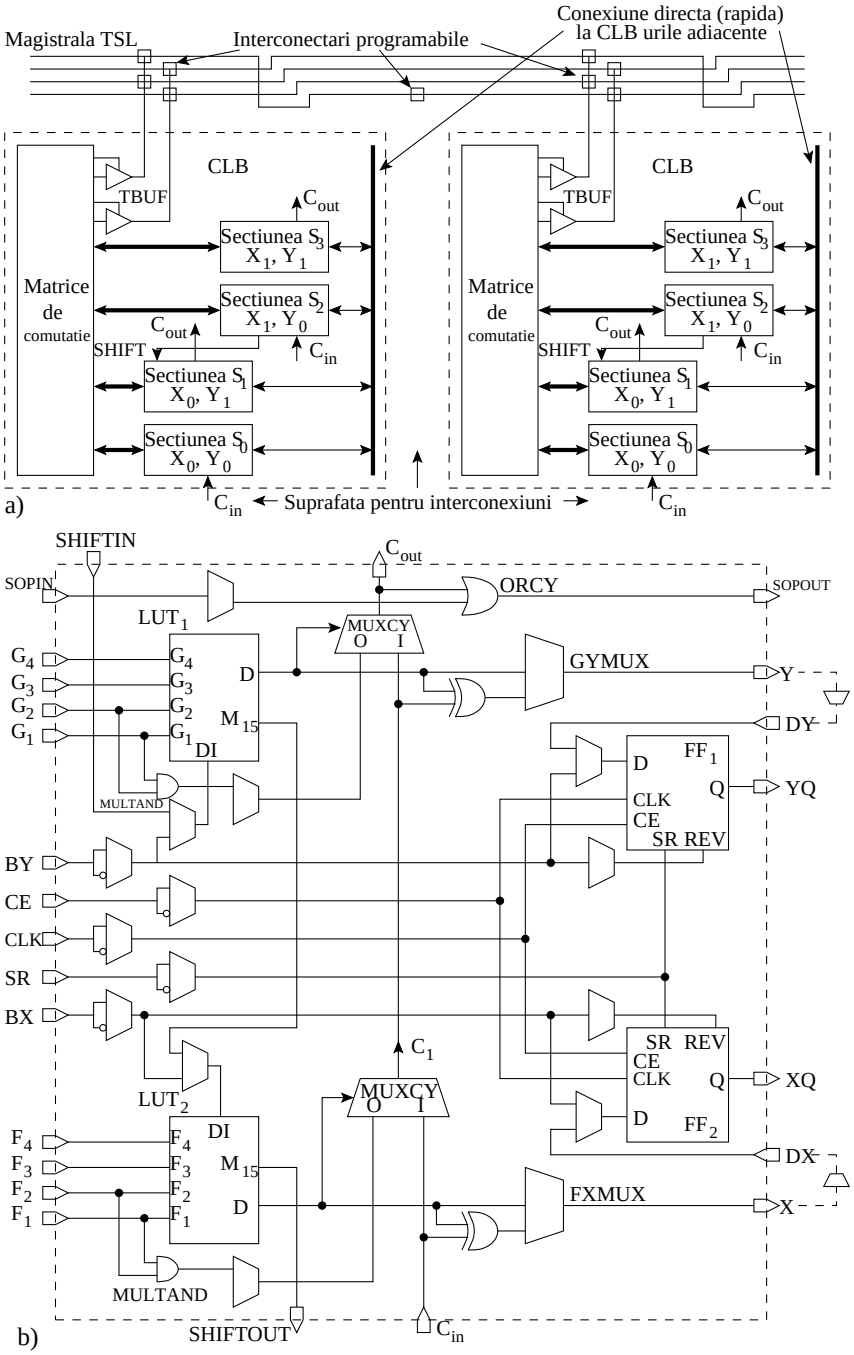


Figura 4.17 FPGA din platforma VIRTEX-II Pro: a) componentele din structura unui bloc logic configurabil, CLB (matrice de comutaţie, patru secţiuni plasate pe două coloane S_0S_1 şi S_2S_3 şi două TBUF); b) structura simplificată a unei secţiuni.

SR prescriere în stare 0 sau 1; REV - comanda în starea opusă faţă de cea comandată prin SR (polaritatea fiecăruia dintre aceste semnale se poate selecta independent prin MUX2:1). Registrele/latch-urile pot fi configurate pentru următoarele tipuri de funcţionare: fără set şi reset; set asincron; reset asincron; set si reset asincron; set (PRESET) asincron; reset (CLEAR) asincron; set (PRESET) şi reset (CLEAR) asincron. Ştergerea (reset, CLEAR) este predominantă faţă de înscriere (set, PRESET).

-**Logica artimetică.** Prin utilizarea celor două generatoare de funcţii F şi G, a celor două porţi AND, MULTAND, a celor două porţi XOR şi a multiplexoarelor MUXCY se poate implementa pe o secţiune un sumator de două cuvinte de doi biţi $G_2F_2 + G_1F_1$, vezi relaţia 2.25. Se obţine:

$$\begin{aligned} s_1 &= F_2 \oplus F_1 \oplus C_{in} && \text{la ieşirea } X \\ C_1 &= \overline{(F_2 \oplus F_1)} \cdot (F_2 \cdot F_1) + (F_2 \oplus F_1) \cdot C_{in} && \text{la ieşirea primului MUXCY} \end{aligned}$$

$$\begin{aligned} s_1 &= G_2 \oplus G_1 \oplus C_1 && \text{la ieşirea } Y \\ C_{out} &= \overline{(G_2 \oplus G_1)} \cdot (G_2 \cdot G_1) + (G_2 \oplus G_1) \cdot C_1 && \text{la ieşirea } C_{out} \end{aligned}$$

Direcţia de propagare a transportului este de la LUT-ul F la LUT-ul G; există astfel două căi de propagare ale transportului într-un CLB; una pe secţiunile din prima coloană (S_0 şi S_1) şi a două cale pe secţiunile din a doua coloană (S_2 şi S_3), conform cu notaţiile din Figura 4.17-a.

Pentru a implementa operaţia de înmulţire porţile MULTAND pot genera produsul aritmetic (= produsul logic) a două variabile.

Se poate obţine o formă normală disjunctivă, FND, relaţia 1.12 (sumă de produse, **SOP**) de oricâţi termeni produs prin utilizarea porţii ORCY care are ca intrări semnalul SOPIN (de la CLB-ul anterior) şi ieşirea de la multiplexorul superior MUXCY. Termenii produs generaţi de un generator de funcţii din secţiunea S_i sunt notaţi cu P_{XS_i}, P_{YS_i}; intrarea 0 a multiplexoarelor MUXCY, pentru propagarea transportului, se conectează la masă. Prin colectarea termenilor produs de pe prima coloană la ieşirea porţii ORCY din secţiunea S_1 se obţine:

$$SOUT_{S1} = P_{XS0} + P_{YS0} + P_{XS1} + P_{YS1} + SOPIN$$

iar prin colectarea acestora cu termenii produs generaţi şi în a doua coloană se obţine:

$$SOUT_{S3} = P_{XS2} + P_{YS2} + P_{XS3} + P_{YS3} + SOUT_{S1}$$

Această sumă de porduse se poate aplica la intrarea SOPIN de la CLB-ul următor ş.a.m.d.

De asemenea, prin configurarea multiplexoarelor interne,se poate realiza un MUX16:1, un MUX32:1 pe două CLB-uri, pentru selectarea funcţiilor generate de LUT-uri. Două secţiuni, folosind LUT-urile şi MUXCY, pot implementa un AND cu 16 intrări (încercaţi!).

- **Memoria distribuită.** Fiecare LUT dintr-o secţiune, fiind în fond o memorie RAM, poate fi utilizat ca o memorie RAM asincronă, simplu port (pe ieşire), de capacitate 16×1 bit (înscriere sincronă, ieşirea cu citire asincronă). Cu cele două LUT-uri dintr-o secţiune se poate configura şi o memorie RAM de capacitate 16×1 bit dar dublu port pe ieşire. Adresarea unui LUT, ca memorie RAM simplu port, dintr-o secţiune se face pe intrările F_4, F_3, F_2, F_1 sau G_4, G_3, G_2, G_1. Fiecare LUT mai are încă patru intrări de adresare WF_4, WF_3, WF_2, WF_1 sau WG_4, WG_3, WG_2, WG_1 (nedesenate în structura simplifacată din Figura 4.17-b). Când o secţiune este configurată ca o memorie RAM dublu port 16×1 bit, atunci cuvântul de adresă de înscriere se aplică simultan la ambele LUT-uri pe intrările F_4/WF_4, F_3/WF_3, F_2/WF_2, F_1/WF_1 de la F_1 şi pe intrările WG_4, WG_3, WG_2, WG_1 de la G; pentru citirea (asincronă) a portului F cuvântul de adresă se aplică pe aceleaşi intrări ca şi la la înscriere, în schimb pentru citirea (asincronă) a portului G cuvântul de adresă se

aplică numai pe intrările G_4, G_3, G_2, G_1. Cu cele opt LUT-uri dintr-un CLB se pot configura următoarele structuri de memorie RAM: ca simplu port 16×8 biţi, 32×4 biţi, 64×2 biţi şi 128×1 bit, iar ca dublu port 16×4 biţi, 32×2 biţi, 64×1 bit. Memoria RAM distribuită când se utilizează toate CLB-urile, în acest scop, este de 45056biţi pentru XC2VP2 şi de 1779712 pentru XC2VP125.

- **Registru de deplasare.** Fiecare LUT poate fi configurat ca un registru de deplasare cu intrarea date (care poate fi SHIFTIN) pe intrarea DI, celula zero a registrului, şi ieşirile de date pe M_{15}, celula a 16-a a registrului. Înscrierea în registru de deplasare se face serial, sincron pe semnalul de ceas CLK în conjuncţie cu semnalul de validare ceas CE; citirea este asincronă din oricare celulă prin aplicarea pe intrările LUT-ului a cuvântului de adresă pentru celula care se doreşte a fi citită. Una din cele 16 celule ale registrului de deplasare poate fi citită şi sincron dacă ieşirea acestei celule se conectează la intrarea bistabilului D. Într-o secţiune, prin inserierea celor două LUT-uri (M_{15} de la G se conectează la DI de la F) se obţine un registru serie de 32 biţi (ieşirea fiind SHIFTOUT); pe un CLB registrul de deplasare poate fi de 128 biţi.

Resursele logice dintr-un CLB sunt următoarele:

Secţiuni	LUT-uri	Bistabile	MULTAND	Circuite pentru propagarea transportului	Circuite SOP	Memorie RAM distribuită	Registru serie	TUBF
4	8	2	2	2	2	128 biţi	128 biţi	2

- **Blocuri independente de memorie RAM şi de multiplicatoare.** Platforma VIRTEX-II PRO are ca resursă pentru memorare, în afara utilizării LUT-urilor ca memorie RAM distribuită, în plus şi blocuri independente de memorie RAM, dublu port, de capacitate 18kb. De asemenea, în afară de realizarea de multiplicatoare pe baza CLB-urilor, există şi blocuri independete de multiplicare pentru înmulţirea a două cuvinte, fiecare cuvânt de 18 biţi, în complement de doi. În general, fiecare bloc multiplicator se ataşează la un bloc de memorie dar poate fi utilizat şi separat. Combinaţia bloc independent de memorie + multiplicator + acumulator (realizat în LUT-uri) permite implementarea **funcţiei de multiplicare-acumulare** din procesoarele de semnal, funcţie care este de o utilizare comună în calculul răspunsului la impuls finit, **FIR**, (**F**inite **I**mpulse **R**esponse), a răspunsului la impuls infinit, **IIR**, (**I**nfinite **I**mpulse **R**esponse) pentru filtrele digitale. Pe fiecare membru al familiei VIRTEX-II PRO numărul de blocuri de memori RAM este egal cu numărul de blocuri de multiplicatoare şi variază între 12 (pentru XC2VP2) până la 556 (pentru XC2VP125).

2. Blocul de intrare/ieşire, IOB

Pe măsură ce FPGA-urile cresc în numărul de componente integrate şi se lărgeşte gama de facilităţi oferite, cresc şi în dimensiune, viteză, performanţă şi sistemele implementate cu acestea. Într-un astfel de sistem circuitul FPGA comunică cip-la-cip în exterior pe placa de circuit imprimat cu alte componente; această comunicaţie prin întârzierile introduse are un impact substanţial asupra performanţelor. În general, această comunicaţie se realizează pe magistrala pentru care s-au elaborat o serie de standarde I/O de tensiuni reduse (în tendinţa de a reduce puterea disipată). Fiecare standard de I/O are propriul său set de specificaţii referitoare la tensiune, curent, modul de bufferare (intrare, ieşire), modalitatea de terminaţie (intrare, işire, la capăt). Blocul de intrare ieşire la un FPGA trebuie să aibă suficiente resurse şi o mare configurabilitate pentru bufferele de intrare, (driverele) de ieşire astfel încât să suporte cât mai multe standarde I/O. Un buffer de intrare, IBUF, trebuie să permită o configurare fie ca simplu buffer (cu o intrare pe un singur fir), fie ca un buffer cu intrare diferenţială. Un buffer de ieşire, OBUF, trebuie să fie configurat cu o ieşire

push-pull (contratimp) sau cu o ieşire dren în gol (open dren). De asemenea, fiecare tip de buffer trebuie să suporte o varietate de valori de curenţi şi de tensiuni încât să poată fi utilizat pentru cât mai multe standarde I/O. În continuare se vor enumera standardele I/O suportate de familia VIRTEX-II PRO, specificându-se pentru fiecare doar tensiunea (descrierea completă a acestor standarde poate fi găsită la Electronic Industry Alliance JEDEC: http://www.jedec.org)

- **PCI** (**P**eripheral **C**omponent **I**nterface). Este în variantele de magistrale comandate de o frecvenţă de ceas de 33MHz şi 66MHz. Necesită: $V_{CCO} = 3,3V$, IBUF simplu (ca la LVCMOS), OBUF cu ieşirea în push-pull. Nu necesită V_{REF} (tensiune de referinţă) şi V_{TT} (tensiune pe terminaţie).

- **GTL** (**G**unning **T**ranceiver **L**ogic Terminated). Este un standard de mare viteză. Necesită: OBUF de tip dren în gol (deci nu se specifică valoarea pentru V_{CCO}), IBUF de tip diferenţial ($V_{REF} = 0,8V$), $V_{TT} = 1,2V$. Există şi în varianta **GTLP** (**G**unning **T**ranceiver **L**ogic **P**lus) cu $V_{REF} = 1,0V$ şi $V_{TT} = 1V$.

- **HSTL** (**H**igh-**S**peed **T**ranceiver **L**ogic). Necesită: IBUF de tip diferenţial şi OBUF de tip push-pull, $V_{CCO} = 1,8V$ sau $1,5V$. În funcţie de valorile pentru V_{CCO}, V_{REF} şi V_{TT} există patru variante ale standardului.

- **SSTL2** (**S**tub **S**eries **T**erminated **L**ogic for 2,5V). Necesită: IBUF de tip diferenţial (deci se specifică V_{REF}), OBUF cu ieşirea push-pull (se specifică V_{CCO}), V_{TT}. SSTL18 este varianta cu $V_{CCO} = 1,8V$.

Blocurile I/O sunt plasate pe cele patru laturi (perimentrul) ale cipului şi sunt grupate câte două sau câte patru. Fiecare IOB din aceste grupuri poate fi utilizat independent, pentru o transmisie pe un singur fir, sau poate fi utilizat împreună cu un alt IOB din grup realizând o pereche diferenţială. Conectarea unui IOB la căile de rutare ale FPGA-ului se face prin intermediul unei matrice de comutaţie, Figura 4.18-a; o pereche diferenţială trebuie să fie totdeauna compusă din IOB-uri care sunt conectate la aceeaşi matrice de comutaţie. Blocurie I/O de pe fiecare latură a cipului sunt împărţite în două părţi egale, fiecare parte fiind referită prin termenul de bancă, deci toate IOB-urile de pe cele patru laturi formează opt bănci. Tensiunea internă de alimentare, V_{CCINT}, a nucleului circuitului FPGA este $V_{CCINT} = 1,5V$; în exteriorul circuitului, pentru comunicaţia cu alte circuite din sistem valoarea tensiunii V_{CCO} depinde de standardul de I/O utilizat. Regula este următoarea: toate IOB-urile dintr-o bancă utilizează aceeaşi valoare pentru V_{CCO} şi aceeaşi valoarea pentru V_{REF}, deci de la pinii bancii respective pot exista comunicaţii cu elemente exterioare numai pe baza standardelor ale căror tensiuni sunt compatibile cu valorile V_{CCO} şi V_{REF} aplicate la banca respectivă.

Resursele logice ale unui IOB sunt prezentate în Figura 4.18-b, care se compun din: o pereche de elemenete de stocare plus un buffer de intrare, IBUF, pentru calea de intrare şi două perechi de elemente de stocare plus un buffer TSL de ieşire, OBUF, pentru calea de ieşire (o pereche de elemente de stocare pentru intrarea în OBUF şi o pereche pentru comanda TSL a OBUF). Organizarea electrică (simplificată) a unui IOB este prezentată în Figura 4.18-c. Un element de stocare, FF, poate fi configurat fie ca bistabil de tip D fie ca latch D activ pe palier. Semnalele de ceas, CLK_1 şi CLK_2, aplicate unei perechi de elemente de stocare, sunt (de regulă) defazate cu 180^o, în schimb semnalul de validare ceas, CE, este acelaşi pentru o pereche. Este comun pentru toate cele şase elemente de stocare semnalul SR de prescriere a stării, la fel şi semnalul REV (REVerse) care prescrie elementul de stocare în starea opusă în raport cu prescrierea cu semnalul SR; polaritatea fiecăruia dintre aceste semnale de control se poate alege independent (prin MUX2:1). Fiecare FF poate fi configurat pentru următoarele tipuri de funcţionare: fără set şi reset; set asincron; reset asincron; set si reset asincron; set (PRESET) asincron; reset (CLEAR) asincron; set

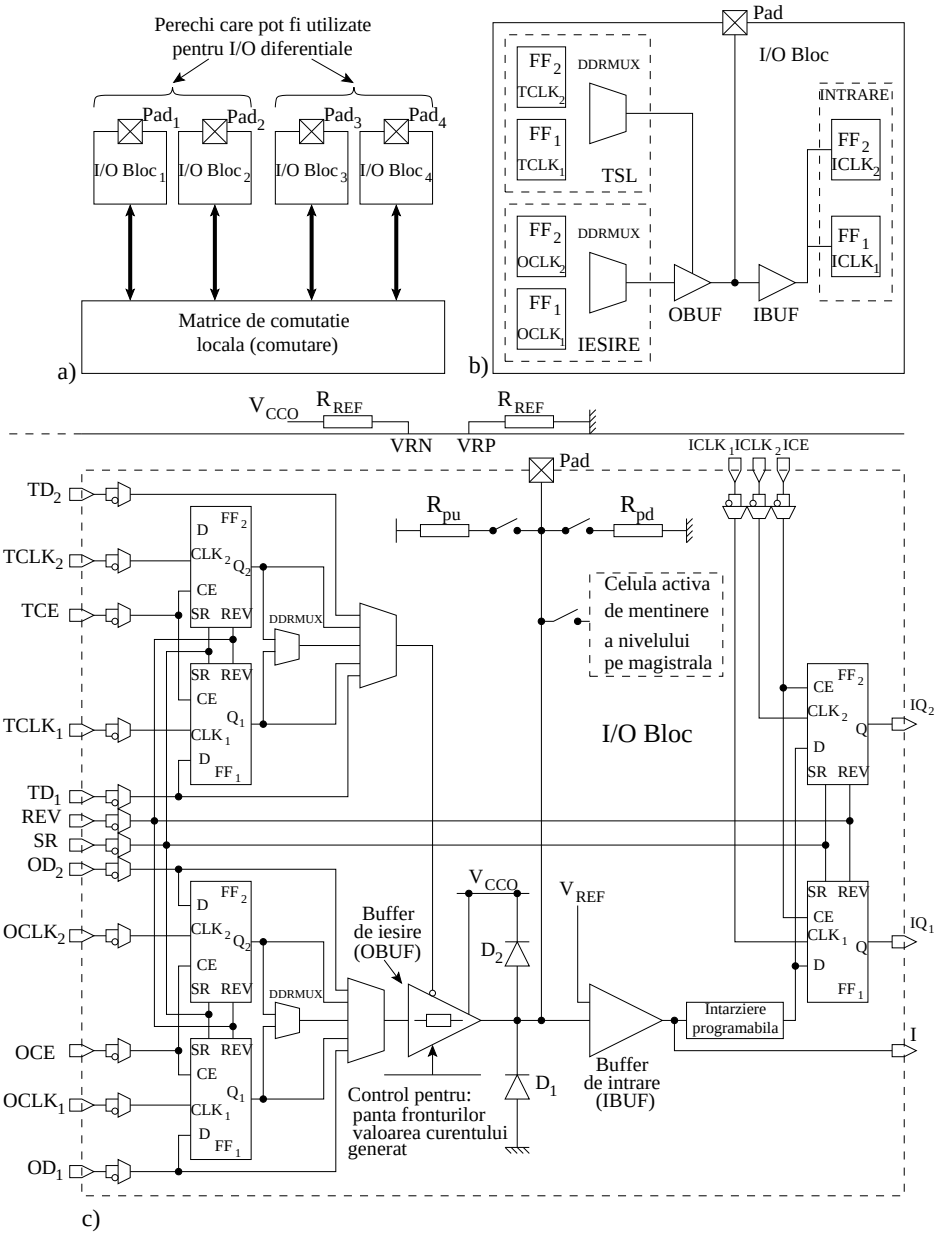


Figura 4.18 Blocul intrare/ieşire, IOB, pentru familia VIRTEX-II PRO: a) gruparea IOB-urilor la o matrice de comutaţie; b) resursele logice conţinute într-un IOB; c) structurarea (simplificată) pentru cele trei căi (intrare, ieşire şi TSL) ale unui IOB.

(PRESET) şi reset (CLEAR) asincron. Stergerea (reset, CLEAR) este predominantă faţă de înscriere (set, PRESET).

Un semnal exterior aplicat la pad este directionat la IBUF, care poate fi configurat cu o singură intrara sau cu intrare diferenţială (V_{REF} se aplică pe una dintre intrări). Semnalul bufferat este rutat la partea logică (CLB-uri) fie direct (I), fie după ce a fost sincronizat în FF_1 (IQ_1) sau FF_2 (IQ_2); pe calea prin FF-uri există introdusă o întârziere programabilă (vezi explicaţia de la Figura 4.9-a).

Pentru calea de ieşire, la intrarea OBUF se pot selecta de la partea logică (CLB-uri) fie direct semnalele (OD_1 sau OD_2), fie aceleaşi semnale dar sincronizate (Q_1 sau Q_2) sau fie aceleaşi semnale dar cu rata dublă (Q_1 şi Q_2). La OBUF se poate fixa valoarea maximă a curentului generat la ieşire şi de asemenea, se poate controla valoarea pantei fronturilor semnalului la ieşire (pentru performanţă de viteză sunt necesare pante de valori ridicate, iar pentru transferuri cu putere consumată mai mică sunt necesare pante de valori mai mici). Semenul de control TSL (HZ/normal) pentru OBUF se obţine de la o pereche de FF-uri în aceeaşi modalitate ca şi obţinerea semnalului de intrare în OBUF. Pe ieşire mai sunt prezente următoarele elemente: diodele D_1 şi D_2, rezistenţele R_{pu} şi R_{pd} şi opţional o celulă activă de menţinere a nivelului pe magistrală; rolul fiecăruia dintre acestea a fost explicat în Figura 4.9-a (diodele sunt protecţie la supratensiuni).

Facilităţi inovative introduse la IOB sunt: posibilitatea de control digital a impedanţei fiecărui pin al unui IOB, transferuri la o rată dublă atât pentru calea de intrare cât şi pentru calea de ieşire (aceasta este motivaţia de ce pentru calea de intrare şi pentru calea de ieşire sunt necesare câte o pereche de FF, iar pentru varianta de ieşire TSL sunt necesare două perechi de FF).

Transferul datelor pe intrare la o rată dublă, DDR (**D**ouble **D**ata **R**ate). Mecanismul de obţinere a unei rate duble de transfer a datelor, în raport cu frecvenţa (rata) semnalului de ceas (aplicată deja la DDR SDRAM, vezi secţiunea 3.6.3) se bazează pe utilizarea a două semnale de ceas CLK_1 şi CLK_2 de aceeaşi frecvenţă dar defazate cu 180^o. În figura 4.19-a sunt prezentate structura căii de date pentru modalitatea DDR şi diagramele de semnale. Datele de intrare (DATA), aplicate la un pin I/O configurat ca pin de intrare, sosite din exterior la o rată dublă faţă de frecvenţa de ceas din IOB, sunt stocate alternativ pe fronturile pozitive ale semnalelor de ceas CLK_1 şi CLK_2 (decalate cu 180^o) respectiv în FF_1 sau FF_2. Deci la partea logică (CLB-uri) datele de intrare sincronizate pe fronturile pozitive ale semnalului CLK_2, IEŞIRE_Q_2, se obţine de la FF_2, iar cele sincronizate pe fronturile pozitive ale semnalului CLK_1, IEŞIREA_Q_1, se obţine de la FF_1 (semnale de control CE, Set/PRESET şi Reset/CLEAR sunt comune pentru ambele bistabile).

Transferul datelor la ieşire la o rată dublă. Structura căii de date de ieşire şi diagramele de semnale pentru o generare în exteriorul unui pin I/O, de la un IOB, la o rată dublă a datelor faţă de frecvenţa de ceas din interiorul IOB sunt prezentate în Figura 4.19-b. Cele două şiruri de date DATA1 şi DATA2, obţinute de la partea logică a FPGA-ului, sunt înregistrate în FF_1 respectiv FF_2 pe fronturile pozitive ale semnalelor de ceas CLK_1 şi CLK_2 (defazate cu 180^o). În multiplexorul de ieşire, DDRMUX, selectarea datelor de la cele două intrări (elemente de stocare FF_1 şi FF_2) se realizează alternativ cu semanlele CLK_1 şi CLK_2.

Transferul datelor la ieşire prin buffer TSL la o rată dublă. Pentru un astfel de transfer structurarea căii de date de ieşire şi diagramele de semnal sunt prezentate în Figura 4.19-c. Organizarea poate fi considerată ca fiind compusă din două structuri identice cu cea din Figura 4.19-b, din care una comandă intrarea OBUF iar cealaltă generează semnalul de control pentru trecerea OBUF din starea normală de funcţionare în starea de înaltă impedanţă, HZ, şi invers. Selectarea celor două ieşiri, ale fiecărei pereche de elemente de stocare, se realizează cu DDRMUX-ul respectiv utilizând cele două paliere ale semnalului de ceas (semnalele specifice numai pentru perechea de FF-uri care comandă intrarea bufferului

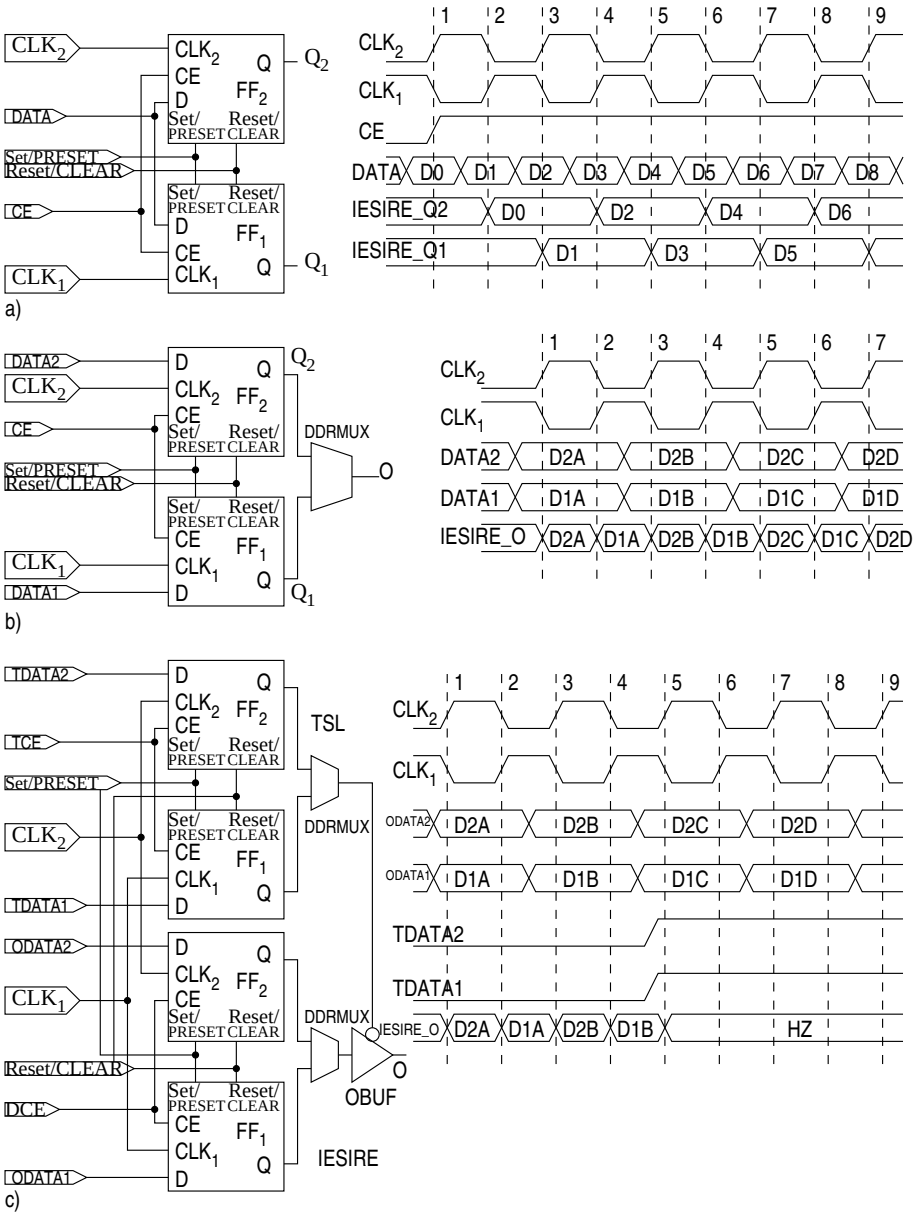


Figura 4.19 Transferul cu rată dublă, DDR, în cadrul familiei VIRTEX-II PRO: a) structurare şi diagrame de semnale pentru DDR pe intrare; b) structurare şi diagrame de semnal pentru DDR pe ieşire; c) structurare şi diagrame de semnal pentru DDR pe ieşire când OBUF este de tip TSL.

de ieşire s-au prefixat cu litera O, iar cele specifice numai pentru perechea de FF-uri care comandă intrarea de control TSL a OBUF s-au prefixat cu litera T). Semnalele de control Set/PRESET, Reset/CLEAR sunt comune pentru ambele perechi de elemente FF, dar semnalele de validare ceas, TCE şi DCE, sunt diferite. Semnalul de ceas CLK_2 se aplică la elementul FF_2 din ambele perechi, iar semnalul de ceas defazat cu 180°, CLK_1 se aplică celălalt element FF_1 din ambele perechi. În cazul în care ambele date obţinute de la partea logică , TDATA1 şi TDATA2, aplicate la perechea de control TSL, au valoarea H rezultă că ieşirea OBUF este în HZ, iar în cazul când TDATA1 şi TDATA2 au ambele valoarea L rezultă la ieşirea OBUF o succesiune de date alternante între ODATA2 şi ODATA1, cu o rată dublă faţă de frecvenţa de ceas.

Când TDATA2 = H şi TDATA1 = L, pe semiperioada $\mathrm{CLK}_2 = 1$, $\mathrm{CLK}_1 = 0$, semnalul de control de la OBUF este H, deci ieşirea bufferului este comandată în HZ, iar pe celaltă semiperioadă a semnalului de ceas, $\mathrm{CLK}_2 = 0$ şi $\mathrm{CLK}_1 = 1$, semnalul de control este L, ieşirea bufferului este IEŞIRE_O = ODATA1. În opoziţie, când TDATA2=L şi TDATA1=H pentru semiperioada de ceas $\mathrm{CLK}_2 = 0$, $\mathrm{CLK}_1 = 1$ ieşirea bufferului este în HZ, iar pentru semiperioada $\mathrm{CLK}_2 = 1$, $\mathrm{CLK}_1 = 0$ ieşirea bufferului este IEŞIRE_O = ODATA2. Pentru aceste ultime două cazuri rezultă la ieşirea OBUF, fie succesiuni ODATA1A, HZ, ODATA1B, HZ, ODATA1C, HZ,..., fie succesiuni ODATA2A, HZ, ODATA2B, HZ, ODATA2C, HZ la o rată dublă faţă de frecvenţa de ceas.

Controlul digital al impedanţei pe terminale. Un circuit al familiei VIRTEX-II PRO implantat pe o placă de circuit imprimat este conectat cu celelalte circuite ale sistemului prin sutele de pini I/O. Transferul semnalelor între aceste circuite, fie pe un singur fir, fie diferenţial, sub unul din standardele I/O amintite anterior, trebuie realizat cu o integritate cât mai ridicată. O astfel de integritate a semnalului se poate obţine numai dacă în punctele de terminaţie (sursă/emisie, recepţie/destinaţie) se face o adaptare cu impedanţa caractaristică, Z_0, a liniei/trasei de pe placa de circuit imprimat, vezi secţiunea 1.6.2.2. Fizic, o astfel e adaptare se realizează printr-o egalizare a rezistenţei echivalente a divizorului de tensiune sau a rezistenţei echivalente serie din punctul de terminaţie (I/O), cu valoarea Z_0. Prin modul de organizare, **BGA** (**B**all **G**rid **A**rray) al sutelor de pini I/O de la un circuit FPGA ar fi aproape imposibil de a conecta astfel de rezistenţe de adaptare. Evitarea acestei multitudini de rezistenţe, în cadrul VIRTEX-II PRO, s-a realizat prin conectarea în exteriorul cipului, pentru fiecare dintre cele opt bănci de blocuri I/O, doar a câte două rezistenţe de referinţă, R_{REF} (cu abateri de sub 1%), deci în total pot fi maxim opt perechi de rezistenţe de referinţă. Una din rezistenţele de referinţă ale perechii de la o banca se conectează între pinul I/O notat cu VRP şi tensiunea V_{CCO}, iar cealaltă rezistenţă de referinţă se conectează între VRP şi masă, Figura 4.20-c. Rezistenţele de referinţă se aleg de valoare egală sau valoare dublă a impedanţei caracteristice, deci $R_{REF} = Z_0$ sau $R_{REF} = 2Z_0$.

La fiecare bancă există o circuistică pentru controlul digital al impedanţei, **DCI** (**D**igital **C**ontrolled **I**mpedance), care poate controla/stabili la fiecare IOB impedanţa de terminaţie la valoarea Z_0. Controlul constă în modificarea impedanţei de ieşire a OBUF sau a unor rezistenţe interne existente la fiecare pad I/O, care sunt conectate unele la V_{CCO} altele la masă. Fizic, modifcarea valorii rezistenţei se realizează prin conectarea a mai multe sau mai puţine din aceste rezistenţe interne, prin intermediul unor comutatoare (tranzistoare de trecere), până când rezistenţa echivalentă de terminaţie devine egală sau jumătate din R_{REF}. Considerând cele două modalităţi de funcţionare (sursă/emisie sau recepţie/destinaţie) ale unui IOB se disting următoarele trei cazuri/modalităţi de adaptare a terminaţiei,Figura 4.20.

- Terminaţia pe sursă (pentru emisie). Prin DCI se modifică impedanţa internă de ieşire a OBUF, fie la $R = R_{REF} = Z_0$, Figura 4.20-a, fie la $R = R_{REF}/2 = Z_0$, Figura 4.20-b.

- Terminaţia pe ieşire (IOB este emiţător); în acest caz DCI nu modifică impedanţa internă a OBUF ci modifică rezistenţe interne (din cip). Tensiunea pe terminaţia de ieşire, V_{TT}, poate fi ajustată fie la tensiunea V_{CCO} (aplicată printr-o rezistenţă conectată la V_{CCO},

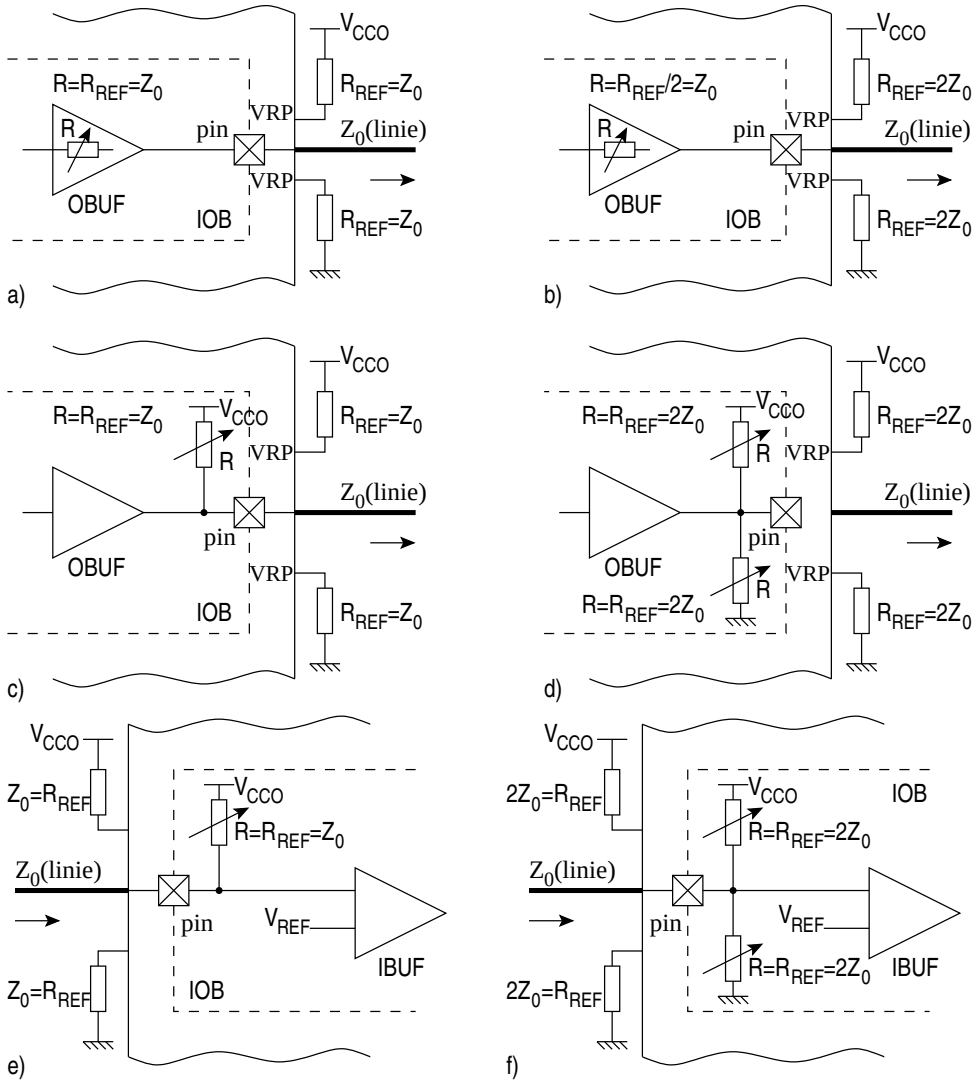


Figura 4.20 Controlul digital al impedanţei într-un IOB din familia VIRTEX-II PRO: a,b) controlul terminaţiei prin impedanţa de ieşire a OBUF pentru R_{REF} respectiv $R_{REF}/2$; c,d) controlul terminaţiei de ieşire pentru o tensiune de V_{CCO} respectiv de $V_{CCO}/2$; e,f) controlul terminaţiei de intrare pentru o tensiune de V_{CCO} respectiv de $V_{CCO}/2$.

valoarea rezistenţei se modifică până se realizează relaţia $R = R_{REF} = Z_0$) Figura 4.20-c, fie la tensiunea $V_{CCO}/2$ (aplicată printr-un divizor între V_{CCO} şi masă, ale cărui rezistenţe se modifică până se realizează relaţia $R = R_{REF} = 2Z_0$), Figura 4.20-d.

- Terminaţie pe intrarea unui IBUF; la cealaltă intrare a bufferului se conectează o tensiune de prag V_{REF} a cărei valoare depinde de standardul I/O sub care se realizează comunicaţia. Şi de data aceasta se poate realiza tensiunea V_{TT} pe terminaţia de intrare pentru V_{CCO} sau $V_{CCO}/2$. Pentru terminaţie la V_{CCO} valoarea rezistenţei se modifică până se realizează relaţia $R = R_{REF} = Z_0$, Figura 4.20-e, iar pentru terminaţie la $V_{CCO}/2$, Figura 4.20-f, rezistenţele divizorului se modifică încât să se realizeze relaţia $R = R_{REF} = 2Z_0$ (rezistenţa Thevenin echivalentă este $R_{Thev} = (2Z_0 \times 2Z_0)/(2Z_0 + 2Z_0) = Z_0$, vezi Figura 1.77-b).

3. Resursele de interconectare.

Resursele de interconectare la VIRTEX-II PRO sunt, în fond, o extensie şi o îmbogăţire a resurselor de interconectare de la XC4000 (Figura 4.15 şi 4.16). Aceste resurse sunt îmbogăţite pe lângă liniile de două lungimi şi cu 120 de linii vericale şi 120 de linii orizontale (pentru întregul cip) de şase lungimi (sixtuplă); iar liniile de conectare directă ale unui CLB sunt imbogăţite cu conectări directe pe orizontală, verticală şi diagonală cu CLB-uri vecine.

Extensia resurselor de conectare apare şi prin introducerea unor noi trasee (dedicate) în cadrul unui CLB, în modul următor, Figura 4.17:

- 4 linii de magistrală TSL care conectează CLB-urile de pe o linie orizontală;
- 2 linii dedicate pentru propagarea transportului, C_{out} (câte una pentru fiecare coloană formată din câte două secţiuni S_0, S_1 şi S_2, S_3);
- 1 linie dedicată pentru propagarea sumei de produse (SOPIN, SOPOUT);
- 1 linie dedicată pentru registrul de deplasare (SHIFTIN, M_{15})

De asemenea, fiecare element CLB sau IOB prezintă o matrice de comutaţie proprie prin intermediul căreia fiecare dintre aceste elemente este legat la matricele de conexiuni programabile (de tipul celor din Figura 4.15 şi 4.16) de pe suprafaţa cipului.

Dar elementul inovator pentru resursele de interconectare este **interconectarea activă**, adică existenţa a câte un buffer pentru toate semnalele care trec prin interconectările programabile (interconectări bufferate). Realizarea căilor de rutare prin intermediul interconectării bufferate determină ca întârzierile pe aceste căi de rutare să fie, relativ, neafectate de încărcarea semnalelor (fan-out).

Matrice de porţi logice programabile prin vias-uri, V*PGA*. Pentru circuitul FPGA, acest cameleon al lumii circuitelor integrate, atât de preferat pentru procesul de dezvoltare de sisteme, programabilitatea pe baza unei memorii RAM de configurare constituie "mărirea şi căderea". "Mărirea" constă în faptul că a adus hardul la nivelul de programabilitate egal cu al softului iar "căderea" că intârzierile de propagare sunt cu mult peste cele obţinute la programarea prin mascare. Soluţia? A apărut o serie de circuite care înlocuiesc parţial sau total programabilitatea celulelor SRAM (atât pentru fixarea conţinuturilor din LUT-uri cât şi pentru conexiunile programabile cu tranzistoare de trecere) cu o programabililtate pe baza unor matrice de vias-uri realizate în straturile metalizate pentru interconexiuni. Evident, că pentru programarea prin vias-uri proiectantul trebuie să transmită la turnătoria de siliciu, înainte de manufacturarea conexiunilor metalizate (vezi Figura 4.5), care din vias-urile din matricea de pe fiecare strat metalizat – strat izolator se realizează şi care se elimină. Odată programate şi realizate aceste vias-uri din matrice, circuitul nu mai poate fi modificat în acest sens, doar dacă mai există în circuit şi LUT-uri

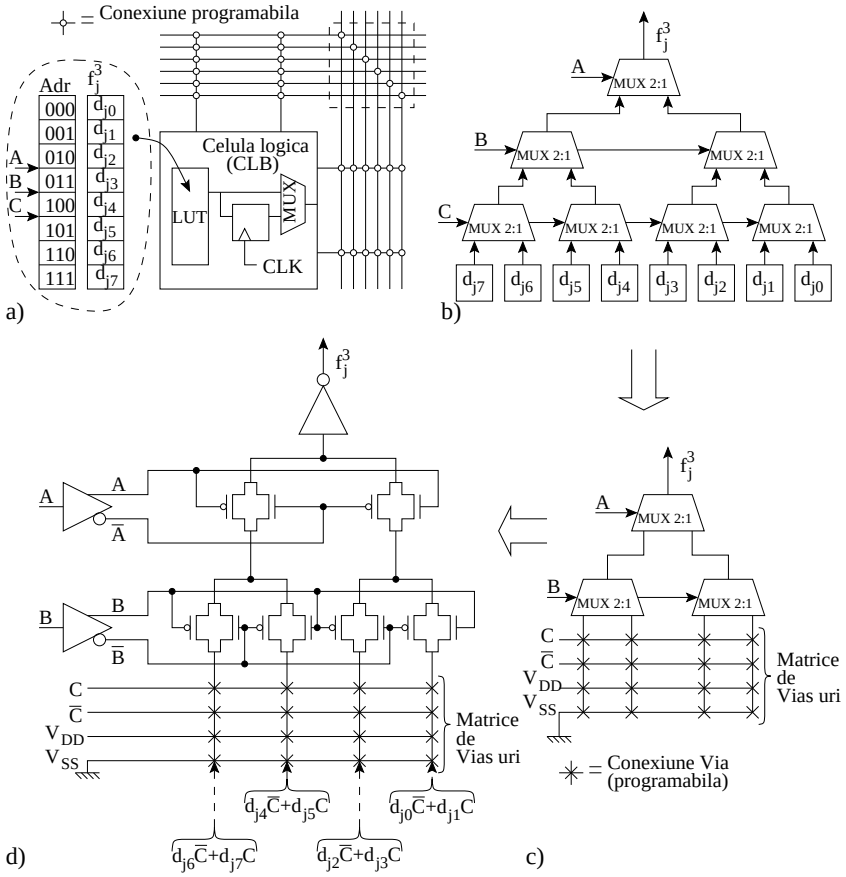


Figura 4.21 VPGA. Conversia unei celule logice de tip FPGA (a) într-o structură de celulă programabilă prin vias-uri b), c), d), VPGA.

pe bază de celule SRAM şi conexiuni programabile pe bază de tranzistoare de trecere care se programează de beneficiar. Aceste circuite hibride care se realizează atât prin programare prin mascare cât şi prin programare printr-un bitstream sunt referite prin **VPGA**: **V**ia-**P**atterned **G**ate **A**rray, sau **FPGA structurate** sau mai general **ASIC-uri structurate**.

Un exemplu de VPGA, cu particularizare la realizarea unei celule logice (CLB), este prezentat în Figura 4.21. O reprezentare simplificată a unei CLB, care conţine un LUT pentru o funcţie de trei variabile, f_j^3, $0 \leq j \leq 255$, este în Figura 4.21-a, iar varianta de realizare a aceleiaşi funcţii dar pe bază de MUX2:1 este dată în Figura 4.21-b; $d_{j_0}, d_{j_1}, ..., d_{j_7}$ sunt coeficienţii funcţiei f_j^3 (biţii înscrişi în LUT). Deoarece ieşirile din primul nivel de multiplexoare pot avea valori care aparţin numai mulţimii $\{0, 1, C, \overline{C}\}$, aceste valori pot fi produse prin matricea de conexiuni (vias-uri programate) ca în Figura4.21-c (nivelul de 4×MUX2:1 corespunzător variabilei C este realizat prin programarea matricei de vias-uri, deci se elimină acest nivel). La nivel

de tranzistor (porţi de transmisie) implementarea celulei logice de tip VPGA este cea din Figura 4.21-d. Celula de tip VPGA obţinută este mai rapidă decât celula FPGA pe bază de SRAM deoarece are un nivel logic mai puţin în arborele LUT, iar conectarea la variabilele funcţiei se face printr-o conexiune metalizată de tip via care este mai rapidă decât conexiunea, cu rezistenţă destul de ridicată, a unui tranzistor de trecere de la FPGA.

VPGA reprezintă un compromis între tehnologia FPGA şi tehnologia cu celule standard, customizând aplicaţia printr-o programare prin mascare (total sau parţial) a rutării şi a implementării funcţiilor logice în locul programării prin bitstream a tranzistoarelor de trecere şi a LUT-urilor. În raport cu tehnologia cu celule standard, tehnologia VPGA realizează: un cost mai scăzut, deoarece nu sunt necesare toate etapele de mascare ale procesului (se elimină cele pentru procesarea componentelor diferite deoarece de data aceasta toate componentele sunt identice) şi datorită regularităţii structurii; o suprafaţă de siliciu şi un timp de propagare comparabil. În raport cu tehnologia FPGA: preţul de cost este mult mai scăzut (nu şi pentru serii foarte mici!); un timp de propagare mult mai bun, dar în schimb se diminuează mult flexibilitatea (reconfigurabilitatea, vezi Figura 4.26-c) în proiectare. Un tip de celulă, în acest sens, care are atât LUT-uri cât şi programabilitate prin vias-uri este cea concepută şi produsă de firma eASIC.

4.7 PROIECTAREA PENTRU TESTABILITATE

Testarea/verificare funcţionalităţii unui circuit digital se face prin aplicarea unor semnale (vectori) de intrare şi observarea semnalelor de ieşire care se compară cu valorile aşteptate de semnale; sau, altfel, pentru a obţine anumite valori pe ieşire se comandă intrarea cu un anumit set de vectori de test. De fapt, acest proces se reduce la realizarea unor acţiuni de control a setului de vectori de test şi de observare (şi analizare) a setului de valori obţinute la ieşire. Aceastora le corespund două noţiuni în teoria sistemelor, cea de controlabilitate şi cea de observabilitate, noţiuni ce vor fi definite în continuare.

Definiţia 4.1 Controlabilitatea stărilor este completă pentru un sistem dacă pentru oricare stare a sa $q(k_0T)$, din momentul k_0T există o succesiune de vectori de intrare $X(kT),\ k = k_0, k_1, ..., k_{N-1}$ care realizează tranziţia sistemului în oricare stare $q(k_N)$, într-un număr finit de tacte $k_N > k_0$. $\diamond$

Definiţia 4.2 Controlabilitatea ieşirilor este completă pentru un sistem dacă din oricare moment de timp k_0T există o succesiune de vectori de intrare $X(kT),\ k = k_0, k_1, ..., k_{N-1}$ după care ieşirea atinge valoarea $Y(k_N)$, într-un număr finit de tacte $k_N \geq k_0$. $\diamond$

Definiţia 4.3 Un sistem este **total controlabil** dacă este complet controlabil pentru stare şi ieşire pentru oricare $k_N \geq k_0$. $\diamond$

Dual\u{a} noţiunii de controlabilitate este noţiunea de **observabilitate**.

Definiţia 4.4 Un sistem este **complet observabil** dacă oricare stare $q(k_0T)$ din momentul de timp k_0T, poate fi determinată prin cunoaştera setului de vectori

de intrare $X(kT)$ şi observarea ieşirii $Y(kT)$, $k_0 \leq k < k_N$, într-un număr (k_N) finit de tacte. Şi este **total observabil** dacă este complet observabil pentru oricare k_0 şi oricare $k_N > k_0$. $\diamond$

Un circuit combinaţional "trece" testul dacă ieşirea sa este corectă pentru toţi cei 2^n vectori de test aplicaţi pe intrare. Un circuit logic secvenţial, care are 2^k stări, necesită, teoretic, aplicarea a câte 2^n vectori de test pe intrare pentru fiecare stare, deci în total 2^{n+k} teste (toate elementele produsul cartezian $X \times Q$); ceea ce uneori duce la un consum de timp/calcul inacceptabil. Se poate reduce timpul te testare, prin eliminarea unui număr de teste, dacă se exploatează anumite particularităţi ale circuitului.

Practic, pentru testare se utilizează punctele de intrare şi ieşire din circuit şi eventual puncte interne de test, la un sistem realizat pe o placă de circuit imprimat sau la un circuit implementat cu componente discrete. La circuitele integrate punctele de test se reduc în exclusivitate la pini. Pentru circuitele VLSI, cu sute de terminale şi aplicare prin **tehnologie de lipire pe suprafaţă**, **SMT** (**S**urface **M**ount **T**echnology), cu circuite pe ambele părţi ale plăcii de circuit imprimat, accesul la toţi pinii necesari pentru procesul de testare devine o problemă. Pentru a elimina această problemă şi a uşura controlabilitatea şi observabilitatea, pe care se bazează procesul de testare, circuitele VLSI actuale sunt proiectate cu un **suport pentru testabilitate**, **DFT** (**D**esigned **f**or **T**estability). Un circuit cu suport DFT necesită puţini pini speciali, maxim cinci, dar cu tot acest număr redus de pini accesibili se pot accesa o mulţime de puncte din interior care nu au ieşirea directă la un pin I/O. În acest sens, în continuare, se va prezenta în ce constă o proiectare pentru testabilitate şi apoi, standardul care descrie DFT pentru circuitele integrate.

Pentru un automat, implementat ca un circuit integrat autonom, există acces doar la pinii săi care sunt intrările principale, IP, şi ieşirile principale, OP, biţii cuvântului de stare $z_{n-1}, z_{n-2}, ..., z_1 z_0$ nu sunt accesibili, starea este o mărime internă a automatului. Se pot explora (scana) şi biţii stării interne dacă se realizează, în acest sens, un **traseu de explorare** (**scan path**) în interiorul automatului la registrul de stare, Figura 4.23-a. Acest traseu este format din celulele registrului de stare cărora li se modifică puţin structura în scopul obţinerii a celor patru facilităţi ale unui registru general: conversie serie-serie, serie-paralel, paralel-serie, si paralel-paralel, Figura 3.81. Structurarea, de principiu al unui astfel de **registru scan** este prezentată în Figura 4.23-b. O **celulă scan** este, în fond, un bistabil D, similar cu cel din Figura 3.80-a, care în funcţie de **semnalul de selectare TMS** (**T**est **M**ode **S**election) poate multiplexa intrarea D, fie într-o intrare paralelă, fie într-o **intrare serie TDI** (**T**est **D**ata **I**n) de la celula scan anterioară; pentru regimul normal, TMS = 0, celula scan se încarcă paralel, iar pentru regimul de testare, TMS = 1, celula se încarcă serie. Un grup de celule scan pot forma fie un registru paralel, cu încărcare pe semnalul de ceas CLK al sistemului, fie un registru serie, cu deplasare pe semnalul de **ceas de testare TCLK**; sunt necesare două semnale de ceas deoarece un registru de deplasare necesită o frecvenţă mai mică $T_{TCLK} > T_{CLK}$. Această celulă de registru este completată pe ieşire cu un latch, astfel încât în registru să poată fi menţinut la ieşire un cuvânt (paralel) în timp ce alt cuvânt, sau chiar cuvântul stocat în latch-urile pentru ieşirea paralelă a registrului, este deplasat în registru, pe semnalul de ceas TCLK, dinspre intrarea TDI spre **ieşirea de date serie**, **TDO** (**T**est **D**ata **O**utput).

Traseul scan, pentru automatul cu un cuvânt de stare de trei biţi $z_2 z_1 z_0$, din Figura

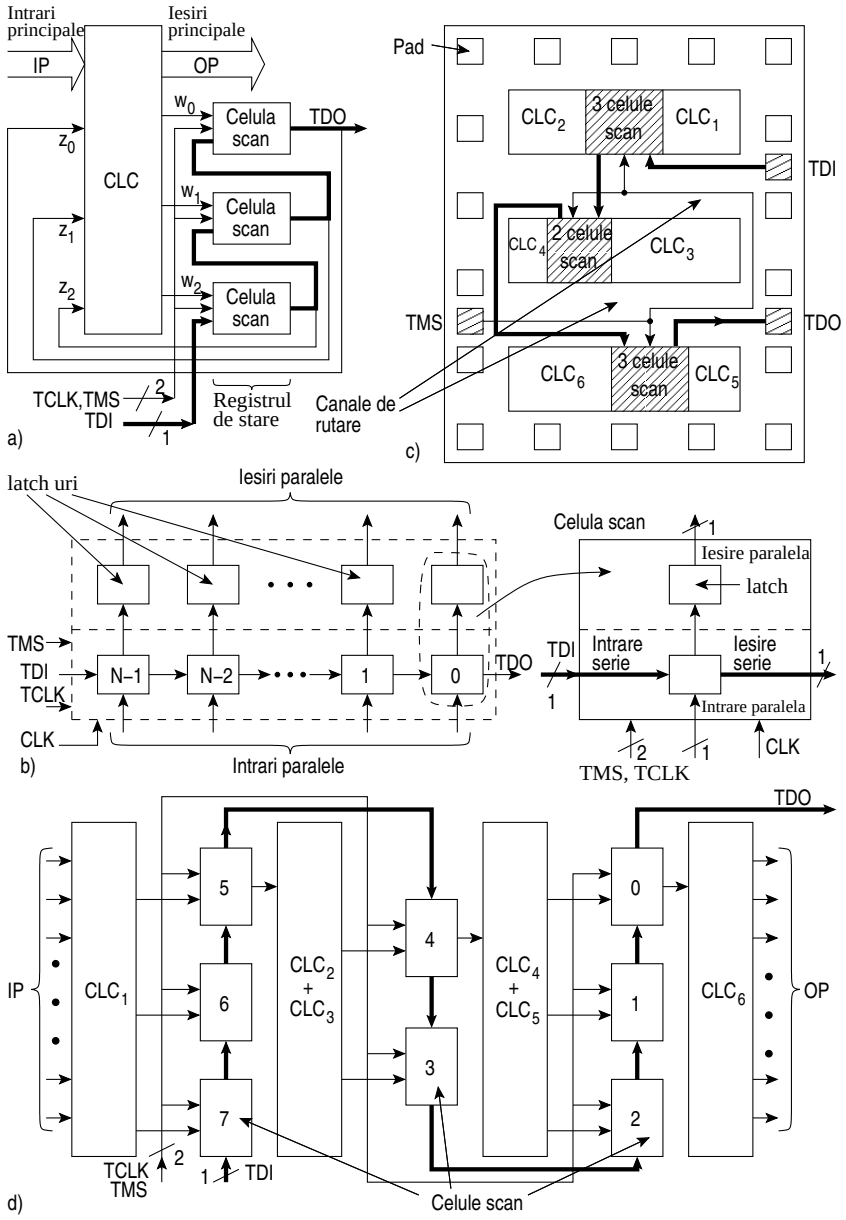


Figura 4.22 Metoda traseului scan pentru testare: a) transformarea registrului de stare al unui automat într-un traseu scan; b) structura de principiu a unui traseu scan cu figurarea semnalelor de control; c) exemplificarea unei posibile poziţionări a punctelor de test pe un layout; d) detalierea traseului scan pentru circuitul de la figura (c).

4.23-a, este format din trei celule scan, interconectarea de tip serie dintre celule este desenată cu linie îngroşată. În fiecare moment se poate citi starea automatului: se trece în modul test, TMS = 1, şi prin aplicarea a trei tacturi ale semnalului TCLK se obţine serie, pe ieşirea TDO, valoarea cuvântului de stare $z_2z_1z_0$ (observabilitate). De asemenea, in fiecare moment se poate injecta o stare în felul următor: se trece în modul de test TMS = 1, cuvântul de stare $z_2z_1z_0$ se aplică la intrarea serială TDI pe durata a trei impulsuri de ceas (controlabilitate).

La o proiectare pentru testabilitate o celulă sau grupuri de celule scan vor fi plasate în interiorul circuitului integrat, în punctele de testare, ca în Figura 4.23-c. Metoda traseului scan se bazează pe faptul că orice circuit digital poate fi privit ca fiind format din blocuri de circuite combinaţionale interconectate prin elemente de stocare (latch-uri, bistabile), Figura 3.76-a. Dacă aceste elemente de stocare sunt de tipul celulă scan, prin proiectare se poate realiza ca toate să fie celule scan, atunci când circuitul este trecut în regim de test, TMS = 1, poate să formeze un registru de deplasare. În această abordare, pentru circuitul cu layoutul de principiu din Figura 4.23-c, în Figura 4.23-d este prezentat traseul scan compus din registrul de deplasare cu opt celule scan. Semnalele din cele opt puncte de test pot fi citite serie la ieşirea TDO, pe durata a opt semnale de ceas TCLK, sau în fiecare punct de test poate fi înscrisă o valoare logică prin aplicarea pe intarea serie TDI, a cuvântului format din opt valori binare, pe durata a opt semnale de tact.

Proiectarea circuitelor VLSI pentru testabilitate poate utiliza standardul IEEE 1149. (Boundary-Scan Test) elaborat în 1990 şi apoi completat cu fiecare variantă lansată. Recent a fost introdus standardul IEEE 1532 pentru dispozitivele care pot fi configurabile/programabile în sistem, standard ce este compatibil cu 1149. Arhitectura suportului standardului IEEE 1149.1 pusă la dispoziţia proiectantului pentru DFT este prezentată (simplificat) în Figura 4.23-a. Fiecare pin I/O al circuitului integrat este "inzestrat" în interior cu o celulă scan, acestea sunt unite într-un registru scan (boundary register); acest registru scan controlează şi observă activitatea la pinii I/O ai circuitului. Circuitului integrat, pe lângă pinii proprii I/O mai trebuie să i se adauge patru pini pentru semnalele TCLK, TDI, TDO, TMS; optional este şi al cincelea pin pentru un **semnal de resetare**, **TRST_L**.

Cuvintele serie introduse pe intrarea TDI pot fi **cuvinte dată** care conţin date ce se încarcă serie în registrul scan sau pot fi **cuvinte intstrucţiune** care se încarcă serie în registrul de instrucţiuni (care are o structurare ca şi registrul scan). Există un automat (ASM cu 16 stări) care, în funcţie de valoarea semnalului TMS aplicat din exterior, generează în fiecare stare semnale pentru procesul de decodificare a instrucţiunii care a fost încărcată în registrul de instrucţiuni. În funcţie de cuvântul specific al instrucţiunii şi de starea prezentă a automatului, pe durata fiecărui semnal TCLK, se generează semnalele de control (TMS_0,TMS_1, ..., TMS_{N-2}, TMS_{N-1}), care prin aplicare la registrul scan fixează modul de funcţionare a fiecărei componente; deci fiecare celulă scan poate să îşi modifice funcţionarea de la tact la tact în funcţie de instrucţiunea primită în registrul de instrucţiuni. De fapt, această aducere succesivă a instrucţiunilor în registrul de instrucţiuni, decodificarea şi execuţia instrucţiunii (generând comenzi la registrul scan) constituie o funcţionare de procesor, iar suportul circuistic introdus de standard nu este altceva decât un procesor rudimentar/simplificat. Testarea circuitului integrat, generarea semnalelor TCLK, TMS, TDI, TRST_L şi colectarea semnalului TDO, se realizează în exterior pe un

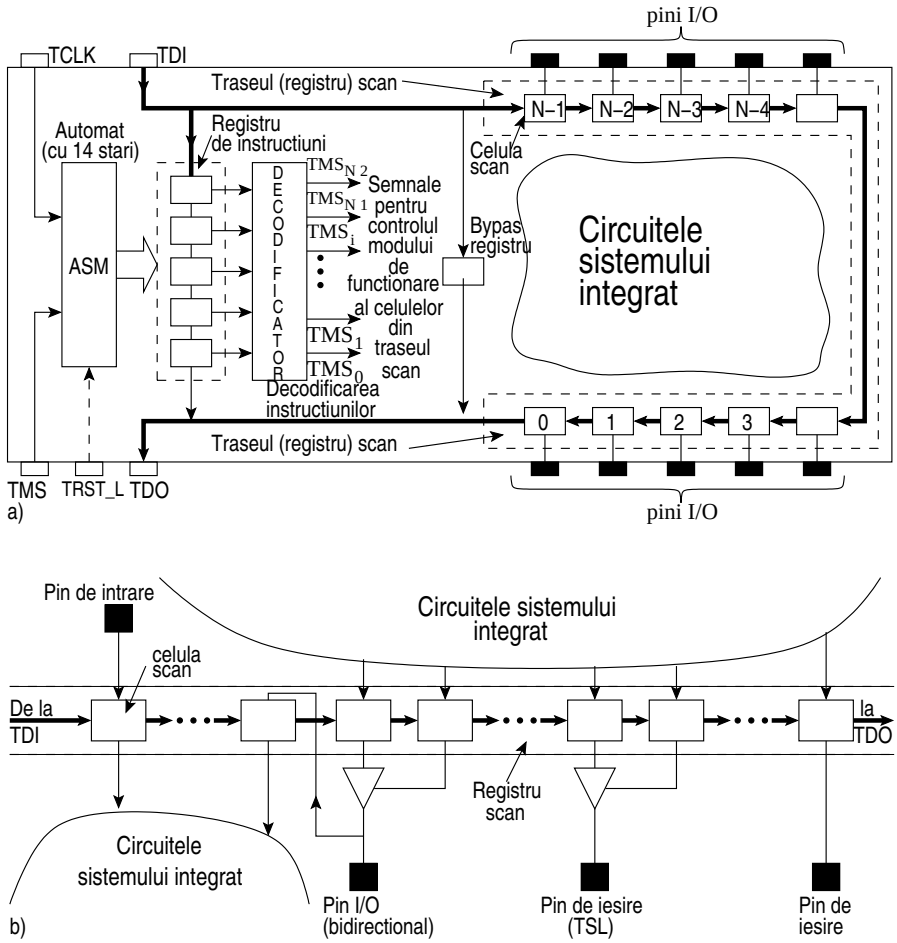


Figura 4.23 Suportul pentru DFT oferit prin standardul IEEE 1149 (Boundary scan): a) arhitectura circuisticii suplimentare introdusă ca suport DFT într-un circuit integrat; b) adaptarea celulelor din registrul scan în funcţie de modul de utilizare al unui pin.

calculator – echipament de testare (tester).

Programul de testare constă în secvenţe de cuvinte intrucţiuni şi secvenţe de cuvinte dată care se introduc serial, prin pinul TDI, în suportul hardware (registrul de instrucţiuni, registrul scan) inculs în circuitul integrat. Pentru dezvoltarea programelor de testare, pe echipamentul exterior de testare, există limbaje specializate în acest sens, **BSDL** (**B**oundary-**S**can **D**ecription **L**anguage).

Un alt element necesar în suportul circuistic implementat pe circuitul integrat este registrul de scurtare (bypas). Acest registru, compus dintr-o singură celulă scan când este selectat, reduce (prin conectarea intrării TDI la ieşirea TDO) la un singur tact numărul de tacturi de deplasare între TDI şi TDO.

Monitorizarea unui pin al circuitului printr-o celulă scan se adaptează în funcţie de modul de utilizare al pinului I/O, Figura 4.23-b. Pentru un pin utilizat numai pentru intrare semnalul cules de la acesta se aplică la celula scan ataşată ca intrare paralelă iar ieşirea paralelă (de la latch) se aplică la circuitele sistemului integrat. Iar pentru un pin utilizat numai pentru ieşire sensul de aplicare al semnalelor se inversează. Pentru un pin de ieşire tip TSL sunt necesare două celule scan, iar pentru un pin cu transfer bidirecţional sunt necesare trei celule (structuri mai complexe pot include aceste funcţionări într-o singură celulă scan). La circuitele CPLD, FPGA care prezintă pe perimetru exterior blocuri I/O configurabile, în care există celule de stocare, aceste celule de stocare pot fi proiectate astfel încât să permită şi o configurare de celulă scan ce poate fi inclusă intr-un (boundary) registru scan.

4.8 COMBINAŢIONAL SAU SECVENŢIAL?

Cu suportul circuistic prezentat, în acest capitol, pot fi implementate atât circuite combinaţionale cât şi circuite secvenţiale. O funcţie logică, aritmetică sau de comunicaţie poate fi implementată pe un circuit combinaţional sau pe un circuit secvenţial; între cele două modalităţi se poate realiza o conversie în ambele sensuri, atât de la combinaţional la secventîal cât şi de la secvenţial la combinaţional. Dar pentru decizia combinaţional/secvenţial proiectantul trebuie să aibă un criteriu; în această secţiune se va explica ce ar putea fi considerat un astfel de criteriu. Pentru aceasta, între cei trei parametrii ai unui circuit implementabil: dimensiunea, $S(n)$, adâncimea, $D(n)$, şi complexitatea $C(n)$, să analizăm, pentru început, care este balansul între primii doi parametrii $S(n) \leftrightarrow D(n)$.

Teorema 4.1 (lui Spira [Spira′ 71]). Un circuit arbore degenerat, de dimensiune şi adâncime constante, $D(n) \in O(n)$ şi $S(n) \in O(n)$, poate fi transformat într-un circuit echivalent având adâncimea în $O(log\ n)$ şi dimensiunea în $O(n^{\alpha})$ cu $\alpha \in (1)$.

Pentru demonstrarea teoremei se va considera arborele degenerat prezentat în Figura 4.24-a în care fiecare CLC_i, $i = 1, 2, \ldots, n$, este un circuit combinaţional cu două intrări, o ieşire şi aceeaşi structură, iar numărul n este o putere a lui 2; această restricţionare nu reduce generalitatea teoremei (fiecare circuit CLC_i ar putea fi diferit, prezentând k intrări şi m ieşiri, de exemplu). Succesiunea demonstraţiei reflectă şi modalitatea practică de transformare a circuitului.

Arborele degenerat se secţionează în doi subarbori primul cu $n/2$ circuite,$CLC_1 \div CLC_{n/2}$, şi cel de al doilea tot cu $n/2$ circuite, $CLC_{n/2+1} \div CLC_n$. Ieşirea primului subarbore se aplică pe intrarea de selectare a unui MUX2:1 (MUXE), iar pe intrările de date ale multiplexorului se aplică câte o copie a celui de al doilea subarbore. Una dintre intrările celui de al doilea subarbore (din structura nesecţionată) este ieşirea $y_{n/2}$ care poate avea, indiferent de valorile aplicate variabilelor de intrare $x_0, x_1, \ldots, x_{n/2}$, de la primul subarbore, fie valoarea 0, fie valoarea 1. Pentru ca funcţionarea circuitului să nu se modifice, prin această secţionare, se aplică la intrările corespunzătoare lui $y_{n/2}$ de la cele două copii ale celui de-al doilea subarbore, la una valoarea 0 iar la cealaltă valoarea 1. Evident că, atunci când primul subarbore generează la ieşirea sa valoarea 0 ($y_{n/2} = 0$) se va selecta la ieşirea multiplexorului copia celui de al doilea subarbore care are pe intrare valoarea 0, respectiv se va

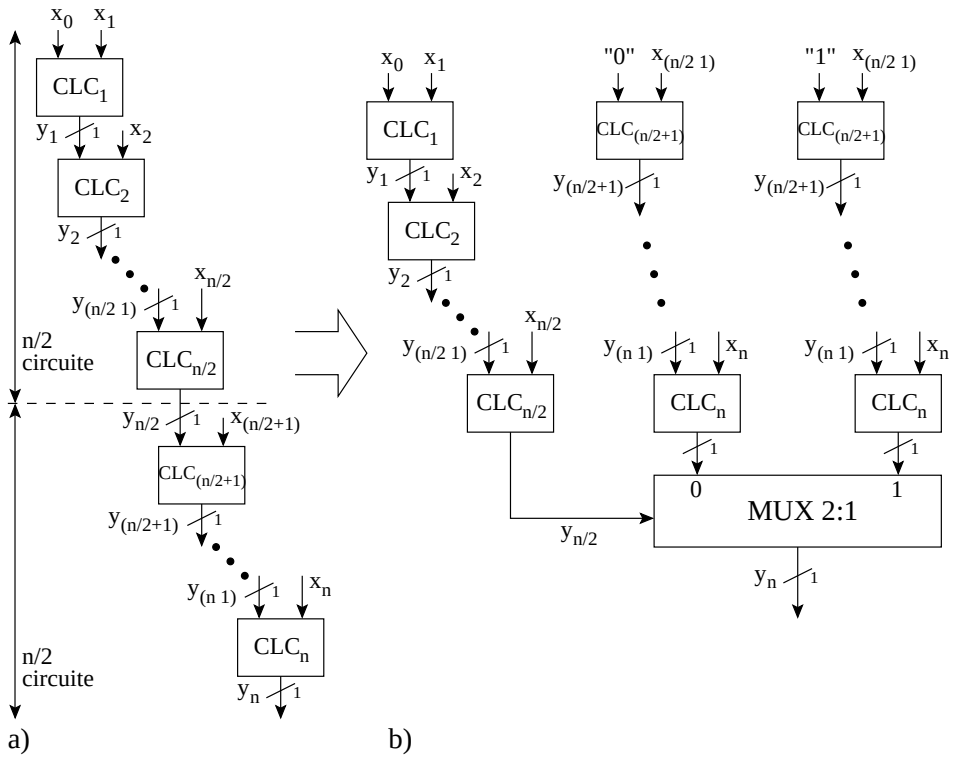


Figura 4.24 Conversia unui circuit de la adâncimea în O(n) într-un circuit cu adâncime în O(log n) prin aplicarea teoremei lui Spira: a) circuitul iniţial cu $D(n) \in O(n)$; b) circuitul obţinut, cu adâncime înjumătăţită, dupa prima secţionare (procesul de înjumătăţire a adâncimii poate continua).

selecta copia celui de al doilea subarbore care are pe intrare valoarea 1 când $y_{n/2} = 1$. Adâncimea circuitului rezultat s-a înjumătăţit iar dimensiunea a crescut cu jumătate din cea a arborelui de intrare plus un MUXE.

Celor trei subarbori rezultaţi după prima secţionare ($i = 1$) li se poate aplica fiecăruia, similar o nouă secţionare ($i = 2$) pe la jumătatea circuitelor CLC componente obţinându-se din nou aproximativ o înjumătăţirea adâcimii circuitului rezultat; numărul de secţionări posibile succesive este $i = log_2\ n$, deci o adâncime $D(n) \in O(log\ n)$. De fiecare dată când se aplică o secţionare numărul de circuite CLC_i se multiplică cu 1,5 şi se adugă un număr de 3^i circuite MUXE, în consecinţă dimensiunea circuitului ramăne în $O(n^2)$.

Pentru cazul când CLC_i sunt cu ieşiri multiple, de exemplu un număr de m ieşiri, multiplexarea se realizează cu $m\times$ MUX2^m:1. După prima secţionare cele m ieşiri ale $CLC_{n/2}$ se aplică la cele m intrări de selectare de la fiecare dintre cele m multiplexoare $2^m : 1$. Pentru copia subarborelui al doilea, ale cărui ieşiri sunt aplicate la intrările de date de ordin i, $0 \leq i \leq 2^m - 1$, de la cele m multiplexoare, la intrarea sa se aplică cuvântul care este codul binar natural al numărului i.

Exemplul 4.5 [Stefan ′00] Circuitul detector de paritate din Figura 4.25-a să i se aplice teorema lui Spira.

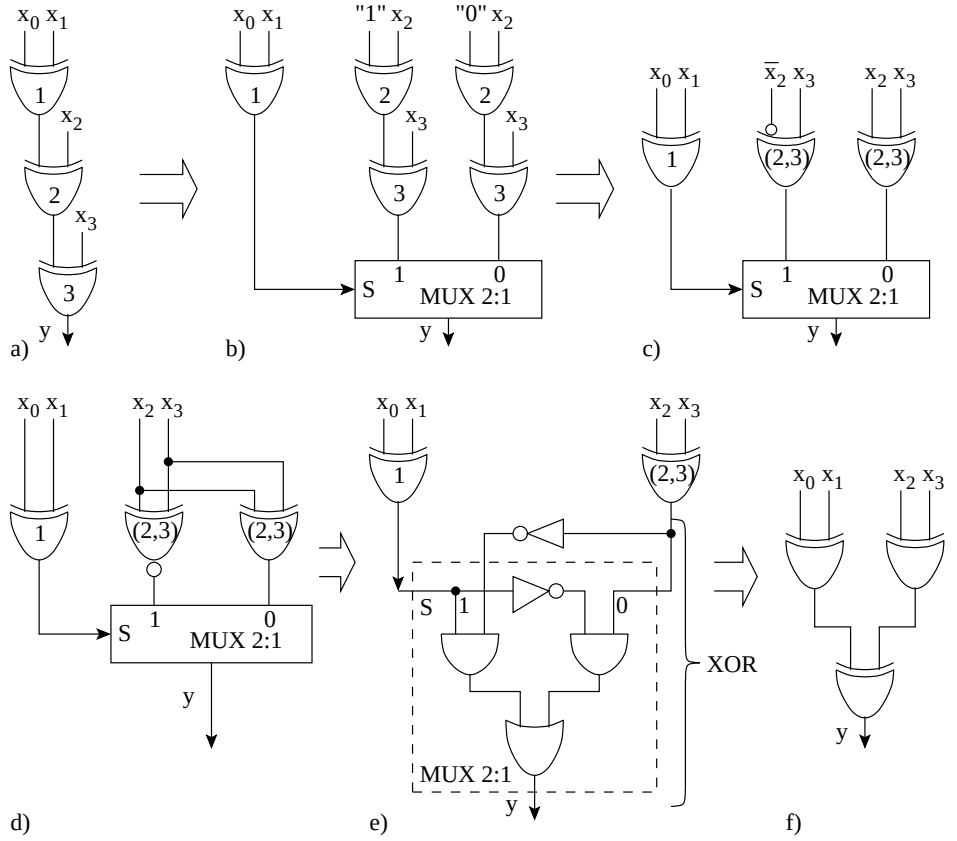


Figura 4.25 Exemplu de aplicare a teoremei lui Spira pentru conversia unui structuri serie de generator de paritate pentru un circuit de patru biţi într-o structură paralelă (pe baza de XOR2)

Soluţie: Arborele degenerat format din 3 porţi XOR este secţionat după prima poartă; primul subarbore format din poarta XOR, notată cu 1, se aplică la intrarea de selectare S a multiplexorului 2:1, iar cele două copii ale celui de-al doilea subarbore, format din porţile XOR notate cu 2 şi 3, se aplică la intrările de date 0,1 ale multiplexorului, Figura 4.25-b. Copia celui de-al doilea subarbore care are pe intrare valoarea 1 se transformă într-o singură poartă NXOR, notată cu (2,3) pentru că $1 \oplus x_2 \rightarrow \overline{x_2} \oplus x_3 = \overline{x_2 \oplus x_3}$, iar copia care are pe intrare valoarea 0 se transformă într-o singură poartă XOR, notată cu (2,3) pentru că: $0 \oplus x_2 \rightarrow x_2 \oplus x_3$, Figura 4.25-c şi d. Operatorul de inversare de la NXOR împreună cu structrura MUX2:1, Figura 4.25-e, formează o structură de XOR, obţinându-se în final, Figura 4.25-f; structură paralelă de detector de paritate pentru cuvinte de patru biţi (vezi Figura 2.19-c şi d). Sugerăm, ca exerciţiu, aplicarea teoremei lui Spira pentru conversia arborilor degeneraţi formaţi din: 1 – şapte porţi XOR (detector de paritate pentru cuvinte

de un byte, Figura 2.19-b); 2 – înserierea a şapte porţi AND2; 3 – înserierea a şapte porţi OR2.

Teorema lui Spira ne arată că la conversia "serialului" în "paralel" scade timpul (adâncimea) dar creşte dimensiunea. Dar cum se manifestă acest "balans" când se trece de la secvenţial la combinaţional şi invers ? La un circuit combinaţional timpul de procesare $T(n)$ se va considera proporţional cu adâncimea $D(n)$.

Un bloc de stare într-o organigramă ASM în care se comută în aceeaşi stare de n ori, adică se parcurge repetitiv o bucla de n ori, cum este prezentat în Figura 2.16-a poate fi modelat printr-o înseriere de n celule identice obţinându-se un circuit combinaţional, Figura 2.16-c.

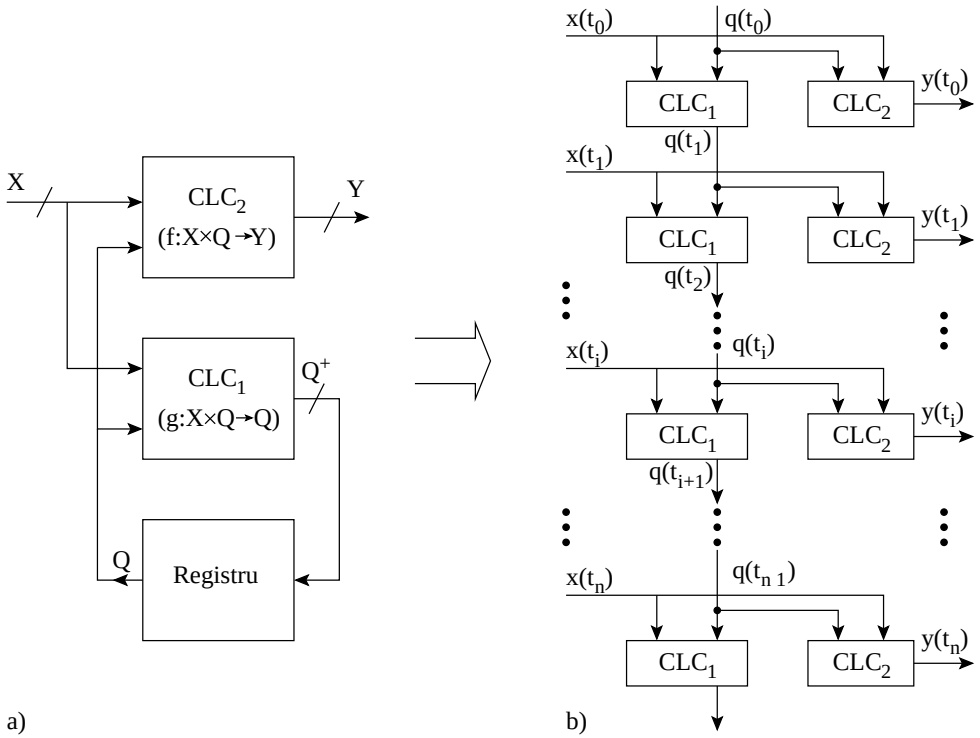


Figura 4.26 Conversia unui automat Mealy (a) într-un circuit combinaţional (b) cu $S(n) \in O(n)$ şi $D(n) \in O(n)$. Conversia inversă este posibilă dacă se poate structura CLC_1 şi CLC_2 conform reţelei din (b).

Să considerăm cazul general, Figura 4.26-a, al unui automat Mealy care calculează funcţia de transfer $f : (X \times Q) \rightarrow Y$ pe circuitul combinaţional CLC_2 şi funcţia de tranziţie $g : (X \times Q) \rightarrow Q$ pe cicruitul combinaţional CLC_1. Aplicând la intrare, sincron cu ceasul, o succesiune de valori ale semnalului de intrare $x(t_0), x(t_1), \ldots, x(t_i), \ldots, x(t_n)$ automatul va parcurge o traiectorie compusă respectiv din stările $q(t_0)$, $q(t_1), \ldots, q(t_i), \ldots, q(t_n)$ generând ieşirile $y(t_0), y(t_1), \ldots, y(t_i), \ldots, y(t_n)$; cu $x(t_i) \in X$, $q(t_i) \in Q$ şi $y(t_i) \in Y$. Acest automat rezolvă problema prelucrării şirului de

intrare $x(t_i)$ într-un timp $O(n)$ pe un circuit CLC_2 de dimensiune $D_{CLC_2} = O(1)$, iar pentru generarea stării următoare (semiautomat) pe circuitul CLC_1 de dimensiune $D_{CLC_1} = O(1)$ tot într-un timp $O(1)$. Dar aceeaşi prelucrare a şirului de intrare $x(t_i)$, $i = 0, 1, \ldots, n$ se poate realiza prin înserierea de n ori a părţii combinaţionale a automatului ($\mathrm{CLC_1}$ + $\mathrm{CLC_2}$) ca în Figura 4.26-b. La primul circuit combinaţional se aplică $x(t_0)$ şi starea iniţială $q(t_0)$, se generează ieşirea $y(t_0)$ şi starea viitoare $q(t_1)$, la al doilea circuit combinaţional se aplcă $x(t_1)$ şi starea $q(t_1)$, se generează ieşirea $y(t_1)$ şi starea următoare $q(t_2)$. Astfel, din aproape în aproape la aplicarea valorilor intrării $x(t_i)$ şi a stării $q(t_i)$ se calculează ieşirea $y(t_i)$, deci circuitul combinaţional pentru şirul de valori de intrare aplicat la succesiv, generează pe ieşiri aceeaşi succesiune pe care o realizează şi circuitul secvenţial de la care am pornit. La circuitul combinaţional obţinut prin "desfăşurarea" părţii combinaţionale a automatului, care se compune prin înserierea de n ori a circuitului combinaţional respectiv, atât adâncimea $D(n)$ cât şi dimensiunea $S(n)$ sunt în $O(n)$.

Circuitul combinaţional obţinut prin "desfaşurarea" părţii combinaţionale a automatului, prin aplicarea teoremei lui Spira, poate fi convertit într-un circuit combinaţional cu dimensiunea $D(n) \in O(n^\alpha)$ şi adâcimea $D(n) \in O(log\ n)$. Prin conversia automatului, cu o adâcime a părţii combinaţionale în $O(1)$ şi un timp de procesare în $O(n)$, deci un produs $S(n) \times T(n) = O(n)$, într-un circuit combinaţional, echivalent din punct de vedere al prelucrării, se obţine o dimensiune în $O(n^\alpha)$ şi o adâncime în $O(log\ n)$, deci în cel mai bun caz un produs $S(n) \times T(n) = O(n \cdot log\ n)$.

Conversia inversă, de obţinere a unui circuit secvenţial dintr-un circuit combinaţional, este posibilă dacă se poate structura partea combinaţională a automatului conform reţelei din Figura 4.26-b.

Pentru proiectant criteriul de decizie în alegerea combinaţional sau secvenţial poate fi produsul adâncime $\times$ dimensiune, $D(n) \times S(n)$, al circuitului, care este proporţional cu produsul dintre timpul de procesare şi dimensiune, $T(n) \times S(n)$. Costul unui circuit, în general, este proporţional cu dimensiunea $S(n)$, (vezi secţiunea 2.3 şi 4.1), iar performanţa (în general viteza de procesare) este invers proporţională cu $T(n)$, deci produsul $T(n) \times S(n)$ poate fi considerat ca fiind raportul cost/performanţă. Produsul $T(n) \times S(n)$ poate fi utilizat ca un criteriu în decizia secvenţial/combinaţional, varianta de circuit care asigură o valoare mai mică pentru acest produs poate fi selectată deoarece realizează un cost mai mic pentru unitatea de performaţă. Balansul dintre $S(n)$ şi $D(n)$ este exprimat de relaţia 2.9 care arată că mărirea performanţei (vitezei) de un număr de ori va determina creşterea costului cu un mai mare număr!

În *Tabelul 4.2* sunt colectate caracteristicile de circuit $D(n)$, $T(n)$ pentru circuite sumator şi circuite multiplicator în variante combinaţionale şi secvenţiale. Circuitul multiplicator secvenţial (poziţia 5 din tabel) nu a fost prezentat în această carte; acest circuit utilizează algoritmul clasic de înmulţire. Se adună succesiv produsele parţiale obţinute prin înmulţirea deînmulţitorului cu câte o cifră a înmulţitorului, dar fiecare produs parţial înainte de adunare se deplasează cu un rang spre stânga. Inspectând tabelul se constată că produsul $S(n) \times T(n)$ (raportul preţ/performanţă) este mai bun la variantele cu implementare secvenţială decât la variantele cu implementare combinaţională atât pentru sumatoare cât şi pentru multiplicatoare. În concluzie, apare că din punct de vedere al raportului preţ/performanţă sunt mai avantajoase circuitele mai lente! Un circuit în acest sens care "încetineşte" timpul de procesare dar în schimb creşte utilizarea componentei combinaţionale şi care poate fi suport

Tabelul 4.2 Valori ale produsului $S(n) \times T(n)$ pentru variante de circuite implementate combinaţional şi secvenţial

	TIPUL DE CIRCUIT	$S(n)$	$T(n)$	$S(n) \times T(n)$
1	Sumator cu transport progresiv, STP (secţiunea 2.5.2.1)	$O(n)$	$O(n)$	$O(n^2)$
2	Sumator cu transport anticipat, STA (secţiunea 2.5.2.1)	$O(n^3)$	$O(1)$	$O(n^3)$
3	Sumator serial cu transport succesiv (secvenţial) (problema P3.44 şi P3.76)	$O(1)$	$O(n)$	$O(n)$
4	Multiplicator matriceal (secţiunea 2.5.3.1)	$O(n^2)$	$O(n)$	$O(n^3)$
5	Multiplicator secvenţial	$O(1)$	$O(n)$	$O(n)$

de implementare atât pentru combinaţional cât şi pentru secvenţial, este FPGA-ul utilizat cu timp multiplexat.

Circuitul FPGA cu timp multiplexat, **TM-FPGA.** Modalităţile de organizare ale unui circuit FPGA sunt prezentate în Figura 4.10 şi dintre acestea varianta de organizare de tip matriceal este reluată în Figura 4.27-a. În "spatele" fiecărui punct programabil (din CLB-uri sau resurele de interconectare) există câte o celulă memorie (latch), a cărei valoare înscrisă determină programarea punctului respectiv; totalitatea acestor celule de memorie constituie memoria de reconfigurare a circuitului FPGA, această memorie va fi referită ca plan de memorie de reconfigurare sau context.

Un circuit TM-FPGA prezintă distribuit, în fiecare dintre punctele de programare nu o singură celulă de memorie (latch) ci un număr de k celule de memorie de reconfigurare care, toate, la nivelul întregii suprafeţe a circuitului, pot fi privite că formează k plane de memorie de configurare (k contexte), Figura 4.27-c. La un moment dat al funcţionării circutului doar conţinutul unui singur plan — **contextul curent** — comandă punctele de programabilitate. Schimbarea unui plan de memorie cu un altul, adică modificarea funcţiei realizată de către FPGA (prin acţiunea unui alt context), se poate realzia într-un interval de timp mai mic de $25ns$. Oricare dintre cele k plane de memorie de configurare poate fi încărcat, din exterior, în timp ce FPGA-ul funcţionează sub programarea (sub informaţia) din unul din planele de memorie (contextul curent).

Organizarea planelor de memorie de reconfigurare este prezentată în Figura 4.27-d. Conţinutul unui plan de configurare i, $i \leq i \leq k$, se aplică la liniile de bit (sub forma unui cuvânt foarte lung $> 10^5$ biţi) şi acest conţinut se va înscrie în celulele de memorie M ale planului în momentul când se activează semnalul de selectare, W_i. Citirea unui plan de memorie şi transformarea acestuia în context curent se face prin activarea liniei de cuvânt respective, W_i; prin tranzistoarele de trecere, comandate prin activarea liniei W_i, conţinutul celulelor de memorie se aplică la liniile de bit, iar pe frontul activ al semnalului de ceas, CLK, se înscrie în latch-urile D care fixează noul context.

O altă modificare a unui circuit FPGA, pentru a fi transformat într-un circuit

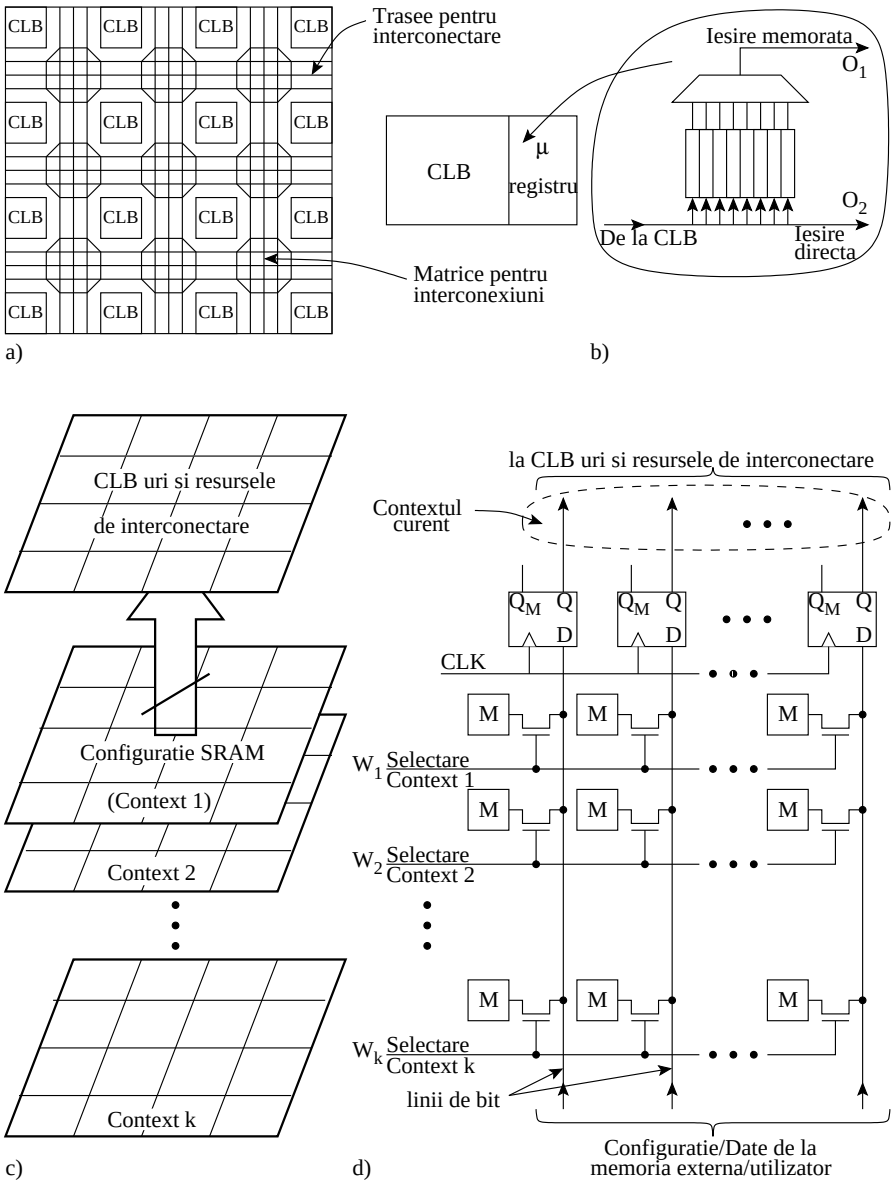


Figura 4.27 Circuitul FPGA cu timp multiplexat TM-FPGA: a) organizarea matriceală a unui circuit FPGA şi extensia acesteia la un TM-FPGA (b) prin adăugarea pe ieşirea fiecărui CLB a unui microregistru; c) reprezentarea intuitivă a planelor de memorie de reconfigurare (contexte); d) structurarea circuitelor din componenţa planelor de reconfigurare.

TM-FPGA, este introducerea microregistrelor. La ieşirea fiecărui CLB se ataşează un număr de k celule bistabile (microregistru) egal cu numărul de plane de memorie de configurare, Figura 4.27-b. În urma funcţionării unui CLB, fiind configurat cu un context, valoarea logică generată se poate obţine direct la ieşire, ieşirea notată cu O_2, sau se înscrie în una dintre celulele bistabil ale microregistrului. Microregistrul memorează valorile ieşirilor unui CLB (atât cele combinaţionale cât şi cele secvenţiale) pentru a fi păstrate după ce contextul FPGA-ului a fost schimbat. Înscrierea ieşirilor CLB-ului în microregistre poate fi validată prin acelaşi semnal de ceas care se aplică şi CLB-urilor.

O valoare de ieşire dintr-un CLB, memorată în microregistru poate fi aplicată (evident printr-o rutare prin interconexiunile programabile) la intrarea aceluiaşi CLB sau la un altul, pentru contextul următor sau pentru un context ulterior. Rezultă că microregistrul este suportul prin care se pot comunica semnale generate de FPGA între două contexte consecutive (**comunicare adiacentă**) sau între două contexte separate de mai multe tacturi de ceas (**comunicare directă**).

Implementarea unui circuit pe FPGA poate fi distribuita pe câteva CLB-uri, pe un întreg circuit FPGA sau chiar pe câteva circuite FPGA, aceasta depinzând de dimensiunea circuitului de implementat. O astfel de utilizare se reduce la o **abordare spaţială a FPGA-ului** pentru că nu se consideră aspectul temporal al utilizării resurselor implicate (LUT-uri, matrice de conectare, trasee de interconectare, elemente de memorare). O **abordare temporală a FPGA-ului** consideră o reutilizare în timp ale acestor resurse, adică cu ce frecvenţă maximă, f_{max}, aceste resurse pot fi reutilizate (în timp), dar asigurând o funcţionare corectă a funcţiei implementată. Frecvenţa maximă de reutilizare depinde de timpul minim de propagare, τ_{pmin}, prin lanţul de resurse implicate în implementarea funcţiei, $f_{max} \leq 1/\tau_{pmin}$. Utilizarea unui **FPGA în regim de timp multiplexat** se bazează tocmai pe această reutilizare a resurselor (schimbarea contextului) cu o frecvenţă de până la f_{max}; deci TM-FPGA utilizează atât abordarea spaţială cât şi cea temporală şi în acest fel utilizarea resurselor creşte până la capacitatea maximă (pentru circuitul implementat scade $D(n)$ dar creşte $T(n)$). Dificultatea care se ridică, la utilizarea unui circuit TM-FPGA, este găsirea partajării/distribuirii optime a funcţiei de implementat, fie în abordarea combinaţională fie în cea secvenţială, pe resursele utilizabile repetat (prin schimbarea contextelor).

Pentru o implementare ca circuit combinaţional funcţia de realizat se partajează în subfuncţii (subcircuite), fiecare subcircuit trebuie să fie implementat pe resursele existente ale FPGA-ului care sunt comandate de către un context; prin comutarea contextelor, al căror număr este egal cu numărul de subfuncţii ce procesează funcţia respectivă. De exemplu, în Figura 4.28-a este prezentată o funcţie sub forma unui grafic aciclic orientat compus din şase noduri (A, B, C, D, E, F) la care se aplică la intrare variabilele x_3, x_2, x_1, x_0 şi generează ieşirea y. Se poate partaja în patru subcircuite (1,2,3,4), nodurile (de procesare) din fiecare subcircuit se repartizează pe resursele circuitului TM-FPGA şi sunt controlate printr-un context. Modul cum cele patru contexte, în succesiunea 1,2,3,4, sunt repartizate celor patru subfuncţii este reprezentat pe organizarea din Figura 4.28-b; schimbarea contextelor corespunde liniilor vericale trasate întrerupt. Comunicarea adiacentă de semnale se realizează la schimbarea contextelor prin intermediul microregistrului. Pentru comunicarea directă de semnale, între contexte neadiacente, semnalele sunt memorate în celule buffer (care

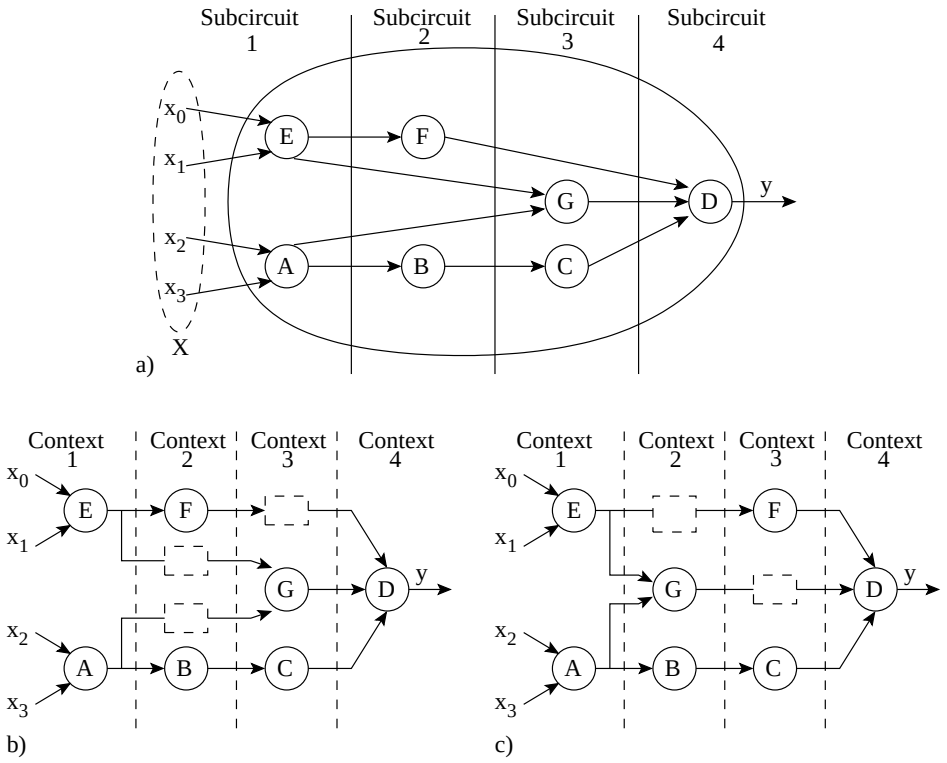


Figura 4.28 Partajarea grafului unei funcţii pentru implementarea pe un TM-FPGA: a) graful aciclic direcţionat al funcţiei şi partajarea în subfuncţii/subcircuite; maparea subcircuitelor pe contexte cu trei celule buffer (b) pentru comunicarea directă de semnale; cu două celule buffer (c).

sunt celuleale microregistrului). De exemplu, pentru comunicarea între nodurile A, E, din contextul 1, cu nodul G din contextul 3 semnalele care se transmit sunt păstrate în cele două celule buffer pe durata contextului 2 (celulele buffer sunt figurate prin dreptunghiuri desenate cu linie întreruptă), iar pentru comunicarea semnalului între nodul F din contextul 2 cu nodul D din contextul 4 este introdusă celula buffer din contextul 3, deci în total 3 celule buffer. Printr-o altă repartizare a subfuncţiilor pe contexte se obţine organizarea din Figura 4.28-c, care a redus numărul de celule buffer la doi. Numărul de celule buffer (care sunt celule ale microregistrelor) pe implementare poate constitui un criteriu de optimizare.

Pentru implementarea ca circuit secvenţial partajarea este, într-o primă abordare, aproape directă care rezultă din analiza grafului de tranziţie al stărilor/ieşirilor sau organigrama ASM. Într-o organigramă ASM pentru fiecare perioadă de ceas corespunde (se activează) doar un bloc de stare, Figura 3.15-d, deci fiecărui bloc de stare îi va corespunde în implementare un context (mapare unu-la-unu între blocuri de stare şi contexte); numărul contextelor fiind egal cu numărul stărilor din organigrama ASM (această abordare simplă poate fi utilă doar atunci când numarul stărilor este

mai mic decât numărul contextelor din TM-FPGA). La trecerea dintr-o stare într-o stare următoarea se comută pe contextul stării următoare, mai mult, se poate utiliza chiar codul stării următoare, calculat la sfârşitul stării prezente, pentru selectarea contextului următor. Comunicarea informaţiei între două stări (codul stării următoare care devine stare prezentă) se realizează prin intermediul microregistrelor. Această partajare simplă apare datorită faptului că funcţionarea unui automat se bazează pe calculul stării următoare care apoi se aplică la intrare ca stare prezentă, ceea ce corespunde cu operarea în timp multiplexat la FPGA. Există şi alte modalităţi de partajare, a unui circuit secvenţial, care nu sunt restricţionate de această mapare unu-la-unu [Chang '99].

4.9 COMPARAŢIE ÎNTRE DIFERITELE MODALITĂŢI DE PROGRAMARE

Progresele permanente în tehnologiile circuitelor integrate, evidenţiate prin legea lui Moore, a determinat în consecinţă şi obţinerea de performanţe la limita tehnologică, evident, aceste performanţe trebuind să justifice costul ridicat al noilor tehnologii (**Legea lui Gordon Moore(1965): "numărul de tranzistoare integrate pe unitatea de suprafaţă într-un circuit integrat se dublează în fiecare an"**; în ultimii ani această creştere exponenţială, după puterile lui doi, s-a diminuat ajungând la dublări la intervale de 1,5 ani sau chiar mai mari). O evoluţie a principalelor dimensiuni specifice tehnologiei CMOS este dată în tabelul din Figura 4.29-a.

a)

Evoluţia unor dimensiuni în tehnologia CMOS

Caracteristica	Anul					
	1999	2001	2003	2006	2009	2012
Lungimea de canal $[\mu m]$	0,14	0,12	0,10	0,07	0,05	0,03
Număr de tranzistoare $\left[\frac{\text{milioane}}{cm^2}\right]$	14	16	24	40	64	100
Suprafaţa cipului $[mm^2]$	800	850	900	1000	1100	1300

b)

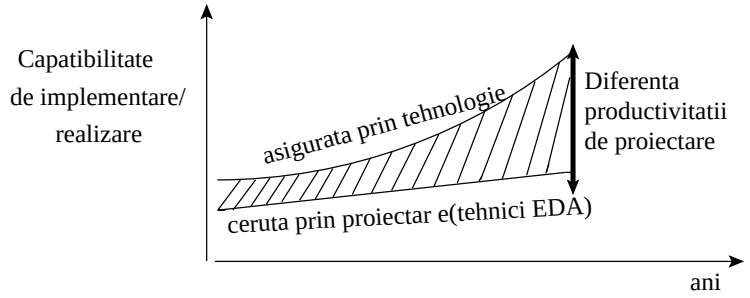


Figura 4.29 Evoluţii: a) a unor dimensiuni specifice tehnologiei de integrare CMOS; b) a capabilităţilor de implementare şi proiectare pentru aplicaţii

De asemenea, evoluţia de până acum a circuitelor integrate a arătat că posibilităţile de implementare disponibile într-o nouă tehnologie care acum se dezvoltă vor depăşi

cerinţele de implementare derminate de abordările prin proiectare (prin tehnicile EDA); această aserţiune referită prin –**diferenţa productivităţii de proiectare**– este redată în Figura 4.29-b. În această figură este dată evoluţia în timp atât a capabilităţii de implementare asigurată de tehnologie cât şi a capabilităţilor de realizare cerute de către proiectare/realizare.

În tehnologia CMOS, odată cu coborârea caracteristicii de proces sub $100nm$, devine din ce în ce mai scumpă şi numărul de defecte creşte. Creşterea numărului de defecte apare din ce în ce mai pregnant, pe lângă defectele de fabricaţie/de proces, prin defecte de tip parametric (zgomot, întârziere etc.). Creşterea costului se datorează pe lângă realizarea unui coeficient de recoltă scăzut (procentul de circuite funcţionale obţinute din totalul de circuite intrate în fabricaţie) şi datorită creşterii costului de proiectare (timp de proiectare, instrumente EDA mai costisitoare). Forţat de aceste considerente în realizarea de circuite suport pentru aplicaţii a apărut conceputul de structuri regulate. Aceste structuri crează circuite cu o anumită regularitate, pe partea logică şi pe partea de interconectare, întâlnită la circuitele standard, dar, în plus, oferă şi unele facilităţi de ASIC, în consecinţă rezultând un tip de ASIC cu preţ scăzut (vezi VPGA).

Circuistica suport în abordarea aplicaţiilor este destul de extinsă, de la circuitele logice discrete până la circuitele de tip full-custom. O structurare a acestei circuistici suport este prezentată în Figura 4.30-a. În secţiunea 4.1 s-a specificat că prin ASIC se înţelege un circuit dedicat la a cărui finalizare, pe lângă turnătoria de siliciu, intervine şi beneficiarul prin proiectare semicustom. În structurarea din această figură noţiunea de ASIC acoperă toate circuitele (full-custom, semicustom şi circuitele PLD (field programmed)) care realizează o aplicaţie specifică/dedicată; în literatură aria de acoperire a noţiunii de ASIC este destul de largă.

Un criteriu prin care să se decidă care mod de proiectare este optim pentru o

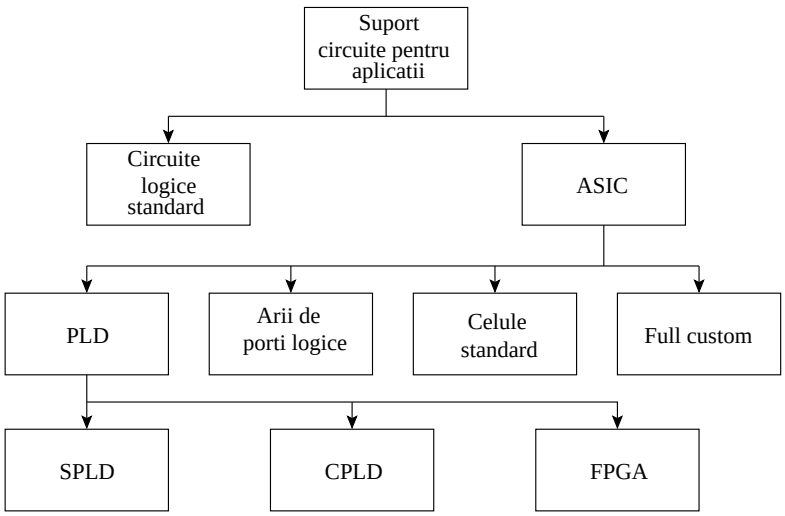


Figura 4.30 Reprezentarea suportului circuistic (tehnologia) pentru abordările de proiectare ale sistemelor logice.

aplicaţie este dificil de exprimat, în general decizia se ia în funcţie de un set de caracteristici asigurate de o anumită proiectare. În acest sens în Tabelul 4.3 se prezintă un set de caracteristici pentru fiecare dintre modurile de proiectare expus în acest capitol (abordarea cu circuite logice discrete a fost prezentată în secţiunea 2.3). De asemeanea, aceste caracteristici nu au fost date prin valori absolute ci prin valori relative; este mai importantă, în prima etapă de selectare a tehnologiei de implementare, o cunoaştere a ordonării relative a acestor caracteristici pentru fiecare tehnologie de implementare urmând apoi, după selectare, să se determine dacă valorile absolute satisfac.

Tabelul 4.3 Caracteristici ale diferitelor tipuri de proiectare

Caracteristici	Tehnologia(Tipul de circuit)					
	Componente discrete	Circuit full-custom	Arie de porţi	Celule standard	CPLD	FPGA
Dimensiunea celulei	-	variabilă	fixă	fixă (înălţimea)	fixă	fixă
Tipul de celulă	circuite diferite	variabilă	fixă	variabilă	programabilă	programabilă
Plasamentul celulelor	variabil	variabil	fix	pe linii	fix	fix
Interconexiunile	variabile	variabile	variabile	variabile	programabile	programabile
Număr de maşti	-	toate măştile	măştile de rutare	toate măştile	0	0
Aria	foarte mare	redusă (compact)	moderat	moderat (redusă)	moderat	mare
Densitate (porţi/chip)	medie	ridicată	medie	medie	scăzută	scăzută
Uşurinţă de (programabilitate)	bună	scăzută	redusă	redusă	bună	foarte bună (reconfigurabil)
Gradul de experienţă al proiectantului	scăzut	foarte ridicat	mediu	mediu	scăzut	scăzut
Timpul de proiectare	zile/ săptămâni	luni/ (an)	săptămâni/ luni	săptămâni/ luni	ore	ore/ zile
Investiţia iniţială	scăzută/ medie	mare	medie	medie	scăzută	scăzută
Performanţa (viteza)	medie	foarte bună	bună	bună	scăzută/ (medie)	scăzută
Recomandat pentru serii	mici/ medii	foarte mari	mari	mari	mici	mici (prototipuri)
Cost/ unitate*	mediu	foarte mic	mic/ (mediu)	mic	ridicat/ (mediu)	ridicat

*Depinde pronunţat de volumul de producţie (costul relativ indicat corespunde pentru valori de serie specificate în rândul anterior)

Capitolul 5

Bibliografie

[**Bryant ′92**] Bryant E. Randy: "Symbolic Boolean Manipulation with Ordered Binary-Decision Diagram". *ACM Computing Surveys*, Vol. 24, No. 3, september 1992.

[**Cocan ′01**] Cocan Moise, Pop Bogdana: *Bazele Matematice ale Sistemelor de Calcul*. Editura Albastră, Cluj, 2001.

[**Cârstea 2000**] Cârstea Horia, *Construcţia şi tehnologia Echipamentelor Electronice*. Editura Politehnică Timişoara, 2000.

[**Creangă ′73**] Creangă I., Reischer C., Simonovici D.: *Introducere Algebrică în Informatică. Teoria Automatelor*. Editura Junimea, Iaşi, 1973.

[**Chang ′99**] Douglas Chang, Marek-Sadowska Malgorzata: "Partitioning Sequetinal Circuit on Dynamically Reconfigurable FPGAs" in *Transactions on Computers*, vol. 48, No. 6, june 1999.

[**Chen ′03**] Chen Eai-Kai (editor), *Logic Design*. CRC Press, 2003.

[**Chinnery ′02**] Chinnery David, Keutzer Kurt: *Closing the Gap Between ASIC & Custom*. Kluwer Academic Publishers, Boston, 2002.

[**Floyd ′90**] Floyd Thomas, *Digital Fundamentals*. Fourth Edition, Merryl Publishing Company, Torronto, 1990.

[**Friedman ′01**] Friedman G., Eby: "Clock Distribution Networks in Synchronous Digital Integrated Circuits", in *Proceedings of the IEEE*, Vol. 89, No. 5, pp. 665-690, May 2001.

[**Green ′85**] Green David: *Modern Logic Design*. Addison-Westley Publishing Company, 1985.

[**Greenlaw ′98**] Greenlaw Raymond, Hoover H. James: *Fundamentals of the Theory of Computation-Principles and Practice*. Morgan Kaufmann Publishers Inc., San Francisco, 1998.

[**Gonţeanu ′96**] Gonţeanu Aurel, Băbăiţă Mircea: *Structuri Logice Programabile. Aplicaţii.* Editura de Vest, Timişoara, 1996.

[**Hennesy ′98**] Hennesy L.I., Petterson, A. D.: *Computer Organisation and Design - The Hardware/Software Interface.* 1998, Morgan Kaufmann Publishers Inc., San Fransico, California.

[**Kang ′96**] Kang S., Lebleblici Y., *CMOS Digital Integrated Circuits: Analysis and Design.* The McGraw-Hill Company Inc., 1996.

[**Lerouge ′04**] Lerouge Christoph: "L′International Technology Roadmap for Semiconductors" in *Sciences Physiques-Nanoscience, Microelectronique, Materiaux*, Jullet 2004, No. 12.

[**Lewin ′92**] Lewin D., Protheore D.: *Design of Logic Systems.* Champmann&Hall Publishing, London, 1982.

[**Matei ′93**] Matei S., Năslău P.: *Elemente de Logică Matematică şi Algebre Boolene.* Universitatea Tehnică Timişoara, 1993.

[**Mead ′80**] Mead C., Conway L., *Introduction to VLSI Systems.* Reading, MA, Addison-Wessley Publishing Company, 1980.

[**Maican ′99**] Maican Sanda: *Circuite Integrate Digitale.* Editura PRIMTECH, Bucureşti, 1999.

[**Mano ′02**] Mano, Morris: *Digital Design.* Prentice Hall International, London, 2002.

[**Mureşan ′02**] Mureşan T., Gonţean A., Băbăiţă M., Demian P.: *Circuite Integrate Numerice - Aplicaţii şi Proiectare.* Editura de Vest, Timişoara, 2002.

[**Nicula ′00**] Nicula Dan: *Proiectarea Sistemelor Digitale Implementate cu Dispozitive Programabile.* Editura Tehnică, Bucureşti, 2000.

[**Omandi ′94**] Omandi R. Amos: *Computer Arithmetic Systems - Algorithms, Architecture and Implementation.* Prentice Hall International (UK) Publishing, 1994.

[**Oberman ′78**] Oberman R. M.: *Numărătoare Electronice.* Editura Tehnică, Bucureşti, 1978.

[**Petterson ′96**] Petterson A. D., Hennesy L. I.: *Computer Architecture - A Quantitative Approach.* 1996, Morgan Kaufmann Publishers Inc., San Francisco, California.

[**Parker ′98**] Parker P. Kenneth: *The Boundary - Scan Handbook.* Kluwer Academic Publishers, 1998, London.

[**Sange ′02**] Sandige Richard: *Digital Design Essentials.* Prentice Hall, 2002.

[**Smith ′97**] Smith M. I. S.: *Application Specific Integrated Circuits.* 1997, Addison-Wesley Publishing Company.

[**Spira '71**] Spira P. M.: "On Time Hardware Complexity Tradeoff for Boolean Functions", in *Proceedings of Fourth Hawaii International Symposium on System Science*, pp. 525-527, 1971.

[**Sutherland '99**] Sutherland I., Sproull B., Harris D.: *Logical Effort.* Morgan Kaufmann Publishers, 1999.

[**Ştefan '91**] Ştefan Gheorghe: *Funcţii şi Structură în Sistemele Digitale.* Editura Academiei Române, Bucureşti, 1991.

[**Ştefan '93**] Ştefan Gheorghe: *Circuite Integrate Digitale.* Editura DENIX, Bucureşti, 1993.

[**Ştefan '97**] Ştefan Gheorghe: *Circuit Complexity, Recursion, Grammars and Information.* Universitatea Transilvania din Braşov, Braşov, 1997.

[**Ştefan '00**] Ştefan Gheorghe: *Circuite şi Sisteme Digitale.* Editura Tehnică, Bucureşti, 2000.

[**Ştefan '92**] Ştefan Gheorghe, Bistriceanu V.: *Circuite Integrate Digitale. Probleme. Proiectare.* Editura Didactică şi Pedagogică, Bucureşti, 1992.

[**Toacşe '96**] Toacşe G., Nicula D.: *Electronică Digitală.* Editura Teora, Bucureşti, 1996.

[**Vlăduţiu '89**] Vlăduţiu Mircea, Crişan Marius: *Tehnica Testării Echipamentelor Automate de Prelucrarea Datelor.* Editura Facla, Timişoara, 1989.

[**Wakerly '01**] Wakerly John: *Digital Design - Principle and Practice.* Third Edition, 2001, Prentice Hall.

[**Weste '01**] Weste N. H. E., Eshraghian K.: *Principle of CMOS VLSI Design.* Second Edition, 1993, Addison-Westley Publishing Company.

[**Yarbrough '97**] Yarbrough, John: *Digital Logic - Application and Design.* West Publishing Company, Minniapolis, 1997.