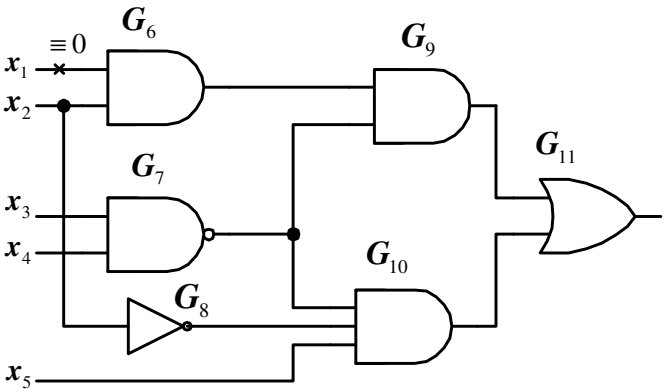
UNIVERSITATEA TEHNICĂ A MOLDOVEI

VIORICA SUDACEVSCHI

TESTAREA SISTEMELOR DE CALCUL

TESTAREA CIRCUITELOR NUMERICE

Prezentare teoretică şi aplicaţii



Chişinău
2010

UNIVERSITATEA TEHNICĂ A MOLDOVEI

FACULTATEA CALCULATOARE, INFORMATICĂ ŞI MICROELECTRONICĂ

CATEDRA CALCULATOARE

TESTAREA SISTEMELOR DE CALCUL

TESTAREA CIRCUITELOR NUMERICE

Prezentare teoretică şi aplicaţii

Chişinău
U.T.M.
2010

CUPRINS

PREFAŢĂ

Realizarea sistemelor de calcul de înaltă fiabilitate este o sarcină, ce presupune utilizarea unui ansamblu de măsuri constructiv-tehnologice complexe, atât la etapa de proiectare, cât şi la cea de exploatare. Aspectele referitoare la elaborarea unui sistem sigur în funcţionare pot fi abordate în două moduri de lucru complementare: detectarea defectării sistemului şi toleranţa la defectări a sistemului. Primul mod presupune depistarea şi localizarea defectelor folosind proceduri de diagnosticare prin testare. Cel de-al doilea mod utilizează redundanţa pentru a neutraliza efectul implicat de apariţia defectelor. La etapa actuală metodele şi tehnicile ce ţin de elaborarea sistemelor fiabile prezintă o importantă direcţie pentru cercetare, proiectare şi industrie. Prin urmare, fiecare specialist în electronică trebuie să aibă pregătirea teoretică respectivă şi să dispună de deprinderile practice necesare în domeniul proiectării şi aplicării în producţie a acestor metode şi tehnici.

Lucrarea de faţă conţine materialul referitor la mijloacele şi principalele aspecte ale testării circuitelor logice combinaţionale şi secvenţiale, precum şi analiza unor coduri detectoare şi corectoare de erori. În capitolul 1 sunt prezentate noţiunile şi principiile de bază ale testării circuitelor numerice, sunt analizate principalele modele de defecţiuni şi conceptele de elaborare a testelor. Capitolul 2 este destinat abordării principalelor aspecte ale testării circuitelor logice combinaţionale şi descrierii mai multor metode de generare a testelor. Pentru studierea lor practică sunt descrise aplicaţii pentru două lucrări de laborator. În capitolul 3 este expusă prezentarea teoretică şi sunt descrise aspectele de testare a circuitelor logice secvenţiale. Tot în acest capitol este prezentat materialul necesar îndeplinirii unei lucrări de laborator. În capitolul 4 sunt abordate unele aspecte din teoria codurilor şi analizate câteva coduri frecvent utilizate pentru detectarea şi corectarea erorilor, ce pot apărea la transmiterea datelor. Materialul teoretic prezentat este necesar pentru îndeplinirea celei de-a patra lucrări de laborator, descrise în acest capitol.

Toate lucrările de laborator vor fi îndeplinite utilizând sistemul de proiectare digitală **Logic Works**. Sistemul **Logic Works** este destinat proiectării şi simulării circuitelor digitale. Asamblarea circuitelor se efectuează cu ajutorul elementelor logice şi circuitelor combinaţionale şi secvenţiale, care se conţin în bibliotecile sistemului. Circuitul

asamblat se analizează cu ajutorul diagramelor de timp şi a elementelor de vizualizare a valorilor logice generate de circuit.

Îndeplinirea primelor trei lucrări de laborator presupune:

- asamblarea schemelor circuitelor numerice şi verificarea testelor prin simularea defectelor corespunzătoare;
- efectuarea analizei circuitelor asamblate şi colectarea datelor pentru darea de seamă a lucrării.

Îndeplinirea celei de a patra lucrări de laborator constă în asamblarea circuitelor pentru codorul şi decodorul Hamming şi verificarea acestor circuite pentru exemple concrete.

Pentru a fi admis la îndeplinirea lucrării de laborator fiecare student trebuie să îndeplinească următoarele condiţii:

- îndeplinirea sarcinii individuale pentru lucrarea de laborator conform variantei;
- cunoaşterea materialului teoretic necesar pentru îndeplinirea lucrării, demonstrat prin răspunsurile la întrebările de control puse de către profesor.

Lucrarea de laborator se consideră îndeplinită doar după ce studenţii demonstrează profesorului corectitudinea circuitelor şi a testelor elaborate.

Pentru fiecare lucrare de laborator studentul pregăteşte o dare de seamă, pe care o susţine în faţa profesorului. Darea de seamă include: foaia de titlu, tema, scopul lucrării, lucrul îndeplinit acasă, schemele tuturor circuitelor asamblate, tabelele de teste elaborate, diagramele de timp obţinute în rezultatul simulării testelor, concluzii şi interpretarea rezultatelor.

1. NOŢIUNI ŞI DEFINIŢII FUNDAMENTALE

1.1. Fiabilitatea

Fiabilitatea este un atribut al echipamentelor tehnice foarte important. Este de preferat să se ţină cont de aspectele legate de fiabilitate încă din faza de proiectare. Apariţia unei teorii a fiabilităţii a fost determinată de creşterea complexităţii echipamentelor şi de caracterul de masă al producţiei moderne.

Fiabilitatea este o disciplină din domeniul ingineriei, care utilizează cunoştinţe ştiinţifice pentru asigurarea unor performanţe

ridicate ale funcţiilor unui echipament, într-un anumit interval de timp şi condiţii de exploatare bine precizate. Aceasta include proiectarea, abilitatea de a întreţine, de a testa şi a menţine echipamentul la parametri acceptabili pe toată durata ciclului de viaţă. Fiabilitatea unui echipament este descrisă cel mai bine de păstrarea performanţelor acestuia în timp. Performanţele de fiabilitate ale unui echipament sunt concretizate în faza de proiectare prin alegerea arhitecturii echipamentului, a materialelor, a procesului de fabricaţie, a componentelor soft şi hard şi prin verificarea rezultatelor obţinute în urma simulărilor şi a testelor de laborator.

Principalele obiective ale fiabilităţii sunt:

a) studiul defecţiunilor echipamentelor (al cauzelor, al proceselor de apariţie şi dezvoltare şi al metodelor de combatere a defecţiunilor);

b) aprecierea cantitativă a comportării echipamentelor în timpul exploatării în condiţii normale, ţinând seama de influenţa pe care o exercită asupra acestora factorii interni şi externi;

c) determinarea modelelor şi metodelor de calcul şi prognoză ale fiabilităţii, pe baza încercărilor de laborator şi a urmăririi comportării în exploatare a echipamentelor;

d) analiza fizică a defecţiunilor;

e) stabilirea metodelor de proiectare, constructive, tehnologice şi de exploatare pentru asigurarea, menţinerea şi creşterea fiabilităţii echipamentelor, dispozitivelor şi elementelor componente;

f) stabilirea metodelor de selectare şi prelucrare a datelor privind analiza fiabilităţii echipamentelor.

Definită din punct de vedere calitativ, fiabilitatea reprezintă capacitatea unui sistem de a funcţiona fără defecţiuni, la parametri acceptabili, în decursul unui anumit interval de timp, în condiţii de exploatare bine precizate.

Din punct de vedere cantitativ, fiabilitatea unui sistem reprezintă probabilitatea ca acesta să-şi îndeplinească funcţiile sale cu anumite performanţe şi fără defecţiuni, într-un anumit interval de timp şi în condiţii de exploatare specificate.

Durata preconizată, de la momentul punerii în funcţiune si până la prima defecţiune, se numeşte **durata de viata** a sistemului. Fiabilitatea este estimată numeric prin probabilitatea că sistemul va funcţiona fără defecţiuni până la momentul $t > 0$, probabilitate calculată de proiectant sau executant pe baza caracteristicilor

componentelor sistemului si a rezultatelor controlului de calitate. Deci, fiabilitatea unui obiect (o componentă sau un sistem) este o funcţie de timp F(t), cuprinsă între valorile 0 şi 1. Nu există sisteme perfect fiabile, pentru care F(t) = 1 dacă t > 0.

Pentru a stabili fiabilitatea unui sistem şi durata sa de viaţă se analizează minuţios defectele ce pot surveni, erorile de funcţionare şi se introduc metode profilactice pentru a le preîntâmpina.

1.2. Diagnosticarea tehnică

Diagnosticarea tehnică este o ştiinţă aplicată, care include teoria şi metodele de organizare a proceselor de determinare a stării tehnice a circuitelor numerice la nivel de pastilă de siliciu, plachetă sau sistem.

Diagnosticarea tehnică rezolvă două sarcini de bază:

1) Detectarea defectelor hardware ale circuitelor numerice;

2) Localizarea acestor defecte în vederea înlăturării lor.

Diagnosticarea se poate realiza prin testare. Testarea poate fi efectuată cu ajutorul mijloacelor hard şi soft, încorporate în obiect sau separate de el.

1.3. Clasificarea defectelor hardware

Defectul sau **defecţiunea** este o imperfecţiune materială sau fizică cauzată de un eveniment de defectare şi care determină modificarea unei variabile logice sau parametru funcţional faţă de valoarea admisă iniţial.

Efectul apariţiei unui defect este **eroarea**, fiind de cele mai multe ori singura indicaţie despre existenţa defectului în sistem. Totuşi, un defect nu întotdeauna conduce la o eroare. În acest caz, defectul se consideră latent.

După durata de acţiune, defectele se clasifică în *defecte tranziente, defecte permanente şi defecte intermitente.*

Defectele tranziente se manifestă printr-o malfuncţie temporară a unei componente, dar nu prin defectarea ei definitivă. Un exemplu, ar putea fi o celulă de memorie al cărei conţinut este schimbat datorită unei interferenţe electromagnetice. Rescrierea ei cu conţinutul corect face ca eroarea să dispară. În sistemele de calcul contemporane, cea mai mare parte a erorilor sunt tranziente.

Defectele permanente se produc la un moment dat şi persistă până când sistemul este reparat. În această categorie includem şi

defectele de fabricaţie a componentelor fizice, factorii nefavorabili de exploatare, îmbătrânirea componentelor. Aceste defecte pot fi înlăturate doar prin efectuarea lucrărilor de reparaţie sau înlocuirea componentelor.

Există si defecte *intermitente*, caz în care defectul componentei pendulează între o stare activă si o stare benignă. Se poate exemplifica cu cazul unei conexiuni slăbite.

După natura sa, defectele se clasifică în *defecte logice* şi *defecte parametrice*. Un defect logic se manifestă prin faptul că valoarea logică a unui nod din circuit devine opusă valorii specificate. Un defect parametric se manifestă prin degradarea mărimilor specifice pentru curent, tensiune şi timp.

1.4. Defecte logice

Această clasă de defecte conduce la lipsa datelor sau date eronate şi se datorează întreruperilor, scurtcircuitelor, imperfecţiunilor fizice sau conceptuale în cadrul nivelelor de realizare a unui produs.

Principalele tipuri ale defectelor logice sunt următoarele:

- blocaje la 0 sau 1 logic la nivelul nodurilor din circuit;
- scurtcircuite cauzate de punţi nedorite, care apar de obicei în faza de execuţie a lipiturilor, între conductoarele imprimate ale plachetei;
- inversoare false la intrările sau ieşirile porţilor logice;
- impulsuri logice eronate;
- impulsuri parazite;
- oscilaţii.

1.4.1. Modele de defecţiuni

La elaborarea testelor se urmăreşte să se acopere o gamă cât mai largă de defecte având la bază modele de defecţiuni.

1) modelul de defecţiune **„blocaj la”** este cel mai răspândit şi a apărut odată cu primele familii de circuite logice, RTL (**R**esistor-**T**ransistor-**L**ogic) şi DTL (**D**iode-**T**ransistor-**L**ogic). Acest model se manifestă prin blocarea unuia sau mai multor noduri de conexiune la valoarea logică 0 - „blocaj la 0” sau la valoarea logică 1 - „blocaj la 1”.

Defectele de tip „blocaj la 0” şi „blocaj la 1” se notează ≡0 şi ≡1, respectiv.

Să presupunem că nodul x_1 al porţii logice ŞI-NU din figura 1.1 este „blocat la 1”. Indiferent de valoarea semnalului aplicat la intrarea x_1, poarta ŞI-NU va percepe acest nod fiind egal cu 1 logic. În acest caz, ieşirea *f* a porţii logice va fi egală cu 0 pentru valorile de intrare x_1=0 şi x_2=1. Astfel, setul x_1=0 şi x_2=1 poate fi considerat un test pentru intrarea x1≡1, deoarece există o diferenţă dintre valoarea ieşirii *f* în absenţa defectului şi în prezenţa lui.

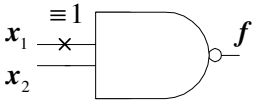


Figura 1.1. Poarta logică ŞI-NU cu intrarea x_1 „blocată la 1”

Modelul „blocaj la” reprezintă un model clasic de defecţiune. El şi-a păstrat utilitatea, chiar la creşterea complexităţii tehnologiei de integrare a componentelor şi oferă o interpretare a celor mai des întâlnite tipuri de defecţiuni (*întreruperea* traseelor din circuit, *scurtcircuitarea* ieşirii la masă sau la borna „+” a tensiunii de alimentare în cazul tehnologiilor CMOS – **C**omplementary-Symmetry **M**etal **O**xid **S**emiconductor).

Figura 1.2 ilustrează realizarea porţii logice ŞI-NU cu două intrări în baza tehnologiei CMOS.

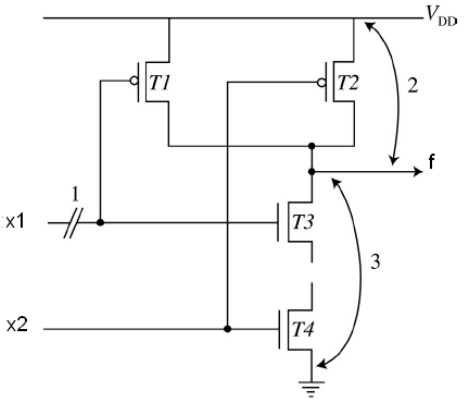
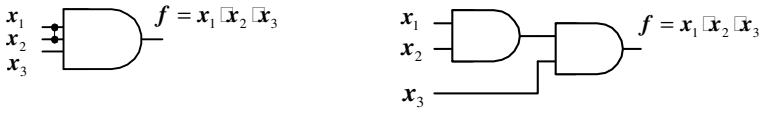


Figura 1.2. Poarta logică ŞI-NU cu două intrări în baza tehnologiei CMOS

Defectul 1 este cauzat de întreruperea traseului. Acest defect va deconecta intrarea x_1 de la tranzistoarele T_1 şi T_2. În acest caz, un tranzistor este blocat, iar celălalt este în conducţie. Astfel, defectul poate fi reprezentat de blocajul valorii semnalului x_1. Dacă considerăm că $x_1 \equiv 0$, tranzistorul T_1 este în conducţie, iar T_2 este blocat şi dacă considerăm că $x_1 \equiv 1$, atunci T_2 este în conducţie, iar T_1 este blocat. Defectul 2 este determinat de o agăţare a ieşirii porţii logice la borna „+" a tensiunii de alimentare (V_{DD}) în urma străpungerii tranzistorului T_2 şi deci reprezintă un defect de tip „blocaj la 1". Defectul 3 este determinat de o agăţare a ieşirii porţii logice la masă în urma străpungerii tranzistorului T_3 şi deci reprezintă un defect de tip „blocaj la 0".

2) modelul **„scurtcircuit"** acoperă defecţiunile frecvent întâlnite atât la nivelul circuitelor integrate cât şi al plachetelor. Defecţiunile de acest tip, care apar în procesul de fabricaţie al plachetelor, sânt în majoritatea cazurilor datorate unor punţi accidentale ale aliajului de lipire, formate între traseele imprimate alăturate. În cazul circuitelor integrate, acelaşi defect este determinat de imperfecţiuni ale izolaţiei între regiuni ale materialului semiconductor, trasee de metalizare, etc.

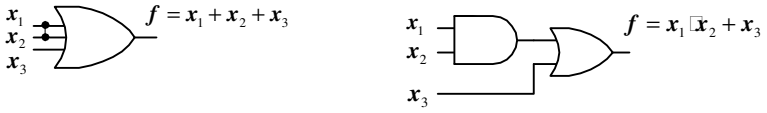
Defectele de tip „scurtcircuit" pot fi de două tipuri: defecte cauzate de apariţia punţilor dintre două sau mai multe linii de intrare şi defecte cauzate de apariţia punţilor dintre linia de ieşire şi cea de intrare a circuitului. Nu toate defectele de acest tip duc la erori în circuitele logice. De exemplu, apariţia unei punţi dintre liniile de intrare x_1 şi x_2 a porţii logice ŞI cu trei intrări (figura 1.3, a) este echivalentă cu introducerea unei porţi logice fictive ŞI (figura 1.3, b), ceea ce nu schimbă funcţia logică realizată de poarta dată.



(a) (b)

Figura 1.3. Exemple de defecte de tip „scurtcircuit" ale liniilor de intrare pentru poarta logică ŞI (a) şi afectarea funcţiei logice (b).

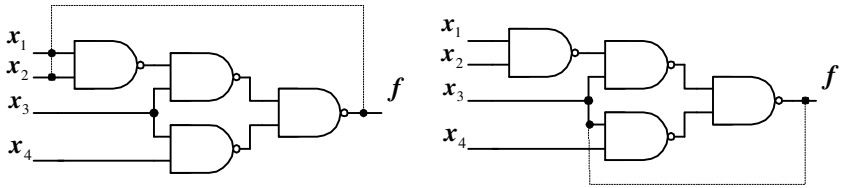
În acelaşi timp, apariţia unei punţi dintre liniile de intrare x_1 şi x_2 a porţii logice SAU cu trei intrări (figura 1.4, a) va schimba funcţia logică realizată de această poartă (figura 1.4, b).



(a) (b)

Figura 1.4. Exemple de defecte de tip „scurtcircuit" ale liniilor de intrare pentru poarta logică SAU (a) şi afectarea funcţiei logice (b).

În cazul apariţiei unei punţi nedorite dintre ieşirea unui circuit logic şi una sau mai multe intrări, acesta trece în regim de oscilare sau se transformă într-un circuit secvenţial asincron. Spre exemplu, pentru circuitul prezentat în figura 1.5, a, scurtcircuitul dintre ieşirea *f* şi intrările x_1 şi x_2 va duce la generarea oscilaţiilor de frecvenţă înaltă, în cazul când $x_1=x_2=x_3=1$ şi $x_4=0$. În acelaşi timp, scurtcircuitul dintre ieşirea *f* şi intrările x_3 şi x_4 va transforma acelaşi circuit combinaţional în unul secvenţial asincron pentru $x_3=x_4=1$ (figura 1.5, b).



(a) (b)

Figura 1.5. Exemple de defecte de tip „scurtcircuit", care duc la transformarea circuitului în generator de oscilaţii de frecvenţă înaltă (a) şi circuit secvenţial asincron (b)

3) modelul de defecţiune **de timp** este necesar în situaţia, în care comportarea dinamică a schemei este dependentă strict de valorile de timp. În principiu, evitarea utilizării circuitelor logice asincrone conduce la minimizarea riscului de apariţie a erorilor datorate parametrilor de timp. Înserarea acestui tip de defecţiune într-un circuit funcţional corect sau simularea devine extrem de dificilă.

1.5. Concepte de bază ale testării circuitelor numerice

1.5.1. Teste

În linii generale, un sistem numeric poate fi reprezentat printr-o mulţime de circuite logice: porţi logice, bistabile, registre, numărătoare, memorii ROM şi RAM, microprocesoare s. a. Pentru efectuarea testării sistemului pot fi accesate doar unele conexiuni.

Intrările, valorile logice ale cărora pot fi nemijlocit controlate, se numesc **intrări primare**.

Ieşirile, valorile logice ale cărora pot fi nemijlocit observate, se numesc **ieşiri primare**.

Detectarea defectelor poate fi efectuată prin aplicarea unei succesiuni de teste şi observarea valorilor de ieşire într-un circuit logic. Practic, majoritatea testelor sunt generate, presupunând că în circuit are loc un singur defect (*defect singular*).

Testul T_k reprezintă mulţimea semnalelor logice aplicate la intrările primare x_i şi mulţimea reacţiilor de la ieşirile primare y_j pentru un circuit logic corect: $T_k = (x_1, ..., x_n; y_1, ..., y_m)$.

Dacă reacţia circuitului este diferită faţă de cea aşteptată, în acest circuit este prezent un defect singular.

Scopul testării la nivelul porţilor logice din circuit este verificarea corectitudinii funcţionării componentelor şi a interconexiunilor dintre ele. Pentru aceasta e necesar a genera teste, care să detecteze defectele singulare pentru toate nodurile din circuit.

Un circuit logic combinaţional (CLC) cu n intrări poate fi testat prin aplicarea a 2^n combinaţii de intrare. Evident, numărul de teste va creste exponenţial, odată cu creşterea numărului variabilelor de intrare.

Pentru un circuit logic secvenţial (CLS) cu n intrări şi cu m bistabile, numărul total de combinaţii aplicate la intrare pentru efectuarea unei testări complete este de $2^n \cdot 2^m = 2^{n+m}$. De exemplu, pentru n=20 şi m=40 vom avea 2^{60} teste. La o rată de 10 000 teste pe secundă, timpul total de testare va fi de aproximativ 3,65 milioane ani.

Pentru a efectua testarea unui circuit numeric nu este necesară aplicarea tuturor combinaţiilor de intrare, ci doar a celor care permit detectarea defectelor singulare. Mulţimea acestor combinaţii de intrare şi valorile aşteptate ale funcţiei de ieşire, se numeşte **mulţimea de teste** pentru circuitul logic.

Mulţimea de teste este completă, dacă ea garantează verificarea prezenţei oricărui defect singular din circuitul logic testat.

Mulţimea de teste este minimală, dacă ea conţine cel mai mic număr de teste.

1.5.2. Principii de elaborare a testelor

După cum a fost menţionat anterior, un test va detecta un defect doar în cazul când valoarea ieşirii primare în prezenţa defectului va fi diferită faţă de valoarea ieşirii primare în absenţa defectului. De exemplu, vom analiza poarta logică ŞI-NU din figura 1.1, a cărei intrare x_1 este blocată la 1 ($x_1 \equiv 1$). În tabelul 1.1 sunt arătate valorile ieşirii porţii logice ŞI-NU în cazul prezenţei şi în cazul absenţei defectului. După cum se observă, doar pentru valorile de intrare $x_1=0$ şi $x_2=1$ valoarea de la ieşirea porţii logice va fi diferită în cazul prezenţei şi în cazul absenţei defectului $x_1 \equiv 1$, deci doar această combinaţie poate fi utilizată pentru testare.

Tabelul 1.1. Reacţia ieşirii porţii logice ŞI-NU la $x_1 \equiv 1$

Intrări		Ieşire	
x_1	x_2	Absenţa defectului	Prezenţa defectului
0	0	1	1
0	**1**	**1**	**0**
1	0	1	1
1	1	0	0

Pentru a detecta un defect într-un circuit, acesta trebuie pentru început *excitat* sau *manifestat*. Procedura constă în aplicarea unei astfel de combinaţii la intrarea circuitului, care să asigure pe nodul testat valoarea logică opusă valorii defectului. Apoi, defectul trebuie *sensibilizat*. Această procedură constă în propagarea univocă a semnalului de la nodul testat spre ieşirea primară a circuitului.

Vom analiza circuitul logic din figura 1.6 cu nodul G_2 blocat la 1 ($G_2 \equiv 1$). Pentru a asigura manifestarea defectului, la intrările primare ale circuitului trebuie aplicate următoarele valori: $x_1=x_2=x_3=1$, deoarece doar în acest caz valoarea nodului G_2 va primi valoarea opusă defectului analizat ($G_2=0$). Petru a propaga defectul prin poarta G_3, vom considera $x_4=1$. În cazul absenţei defectului $F=0$, iar în cazul prezenţei defectului $F=1$. Deci, testul pentru $G_2 \equiv 1$ este:

$T_{G2\equiv1}$ =(x_1, x_2, x_3, x_4; F)=(1, 1, 1, 1; 0).

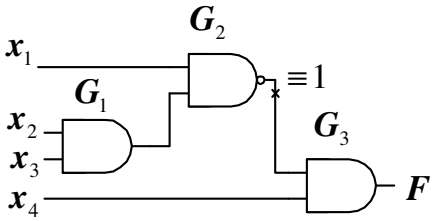


Figura 1.6. CLC cu defectul $G_2\equiv1$

1.5.3. Defecte nedetectabile

Un defect se consideră nedetectabil, dacă este imposibil a atribui asemenea valori la intrările primare ale circuitului pentru a asigura manifestarea lui sau de a propaga univoc valoarea semnalului de la nodul testat spre ieşirea primară a circuitului, pentru sensibilizarea lui. Cu alte cuvinte, un defect este nedetectabil, dacă nu există un test pentru verificarea lui.

Pentru a ilustra cele spuse mai sus, vom analiza circuitul logic din figura 1.7. Pentru a verifica defectul $G_2\equiv0$ este necesar a seta nodul G_2 în 1 logic, ceea ce nu este posibil. Deci, defectul $G_2\equiv0$ nu poate fi manifestat şi se consideră nedetectabil.

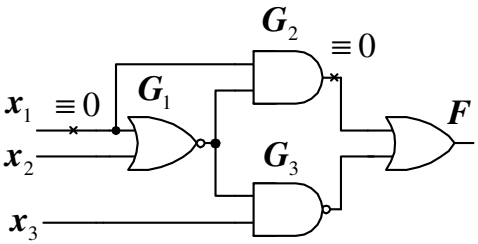


Figura 1.7. Circuit logic cu defecte nedetectabile

Defectul $x_1\equiv0$ poate fi manifestat aplicând x_1=1 şi x_2=0. Dar nu există nici o cale de propagare univocă a acestui defect spre ieşirea primară F, aceasta fiind egală cu 1 logic atât în prezenţa cât şi în absenţa defectului. Nodurile x_1 şi G_2 nu pot fi testate, deoarece circuitul

este redundant, iar porţile şi conexiunile suplimentare creează contradicţii de propagare univocă a semnalelor.

Un circuit logic se numeşte *redundant* dacă el conţine conexiuni sau porţi, eliminarea cărora nu implică schimbarea funcţiei logice a circuitului. Orice circuit redundant va avea defecte nedetectabile.

Elaborarea unei mulţimi de teste pentru un circuit se bazează pe presupunerea că doar un singur defect este prezent în circuit în momentul aplicării testului. Astfel, prezenţa simultană a defectelor detectabile şi nedetectabile nu este în acord cu această presupunere. Mai mult decât atât, prezenţa defectelor nedetectabile poate denatura procesul de localizare a defectelor detectabile.

1.5.4. Testabilitatea, controlabilitatea şi observabilitatea

Testabilitatea este o măsură a capabilităţilor circuitului de a putea fi testat pentru verificarea funcţionalităţii sale. Cele două aspecte majore ale testabilităţii sunt:

Controlabilitatea - măsura capabilităţii circuitului de a stabili valoarea logica a unei variabile interne sau a unei ieşiri.

Observabilitatea - măsura capabilităţii circuitului de a propaga valoarea logica a unei intrări sau a unei variabile interne până la ieşire.

Controlabilitatea, respectiv observabilitatea, este foarte bună dacă un singur vector de intrare permite stabilirea sau semnalizarea unei valori logice într-un punct din circuit şi este foarte slabă dacă este nevoie de o secvenţă de vectori de intrare.

Controlabilitatea unui nod dintr-un CLC, denumită şi controlabilitate combinaţională (CC), este echivalată cu dimensiunea subcircuitului minimal, care determină o anumită valoare logică a nodului respectiv. Subcircuitul care forţează un nod în valoarea 1 poate fi diferit de subcircuitul care forţează valoarea aceluiaşi nod în 0, prin urmare avem controlabilitate combinaţională în 0 (CC0) şi controlabilitate combinaţională în 1 (CC1).

Observabilitatea unui nod dintr-un CLC, denumită observabilitate combinaţională (CO), este echivalată cu dimensiunea subcircuitului minimal, care permite propagarea până la ieşire a valorii logice a nodului respectiv, oricare ar fi aceasta. Acest subcircuit conţine porţile logice ale traseului de la acel nod până la ieşire cât şi subcircuitele necesare configurării corespunzătoare a porţilor logice de pe traseu.

2. TESTAREA CIRCUITELOR LOGICE COMBINAŢIONALE

2.1. Clasificarea metodelor de generare a secvenţelor de test

Metodele de generare a secvenţelor de test, după principiul utilizat, pot fi clasificate în metode *deterministe* şi metode *probabilistice*.

Metodele deterministe reprezintă anumiţi algoritmi formali de generare a secvenţelor de test prin analiza funcţiilor logice şi/sau a structurii circuitului. Aceste metode pot fi clasificate în felul următor:

- *metode structurale*: generarea secvenţelor de test are loc în urma analizei structurii circuitului;
- *metode analitice*: generarea secvenţelor de test are loc în urma analizei funcţiei logice;
- *metode structural-analitice*: generarea secvenţelor de test are loc în urma analizei atât a structurii circuitului logic cât şi a funcţiei logice.

Exemple de asemenea metode sunt: metoda activării unei căi, metoda algoritmului D, metoda derivatelor booleene, metoda formei echivalente normale ş. a.

Metodele probabilistice se bazează pe generarea aleatorie sau pseudoaleatorie a testelor şi se folosesc în cazul circuitelor mari.

2.2. Metoda activării unei căi

Metoda activării unei căi este una dintre primele abordări de generare a testelor de detectare a defectelor singulare în CLC iredundante. Ea a fost propusă de colaboratorul firmei „IBM" Stiglets şi colaboratorul firmei „Bell Telephone Laboratories" Armstrong.

Această metodă structurală este bazată pe alegerea unei căi de propagare a defectului de la un punct de manifestare spre ieşirea primară a circuitului logic.

Procedura de elaborare a testelor constă din următoarele etape:

1) Se asigură obţinerea pe conexiunea defectă a nivelului logic opus presupusei erori (condiţia manifestării defectului);

2) Se selectează în mod arbitrar o cale de la locul de manifestare a defectului la una din ieşirile primare ale circuitului;

3) Se activează calea selectată, asigurând astfel condiţia de observabilitate a defectului prin propagarea univocă a lui până la ieşirea primară a circuitului (procedura constă în *sensibilizarea* porţilor logice din calea selectată);

4) Se determină unul sau mai multe teste pentru detectarea defectului analizat, atribuind valori intrărilor primare, astfel încât să se producă semnalele dorite la ieşirile diverselor porţi logice din circuit;

5) Dacă nu s-a epuizat mulţimea căilor de propagare a tuturor defectelor analizate spre ieşirea primară a circuitului, se reia cu etapa 2, dacă da, atunci generarea testelor s-a încheiat.

Etapele 1-3 reprezintă *faza de trecere înainte,* iar etapa 4 – *faza de consistenţă* sau *faza de trecere înapoi.*

Vom analiza mai detaliat etapa 3 a metodei. Pentru a sensibiliza o poartă logică cu o intrare presupusă defectă (*a*), e necesar a atribui celorlalte intrări asemenea valori, încât valoarea ieşirii să depindă doar de valoarea lui *a.* Spre exemplu, în cazul porţii logice ŞI cu două intrări valoarea de sensibilizare este egală cu 1 logic (figura 2.1). Într-adevăr, pentru *a=1* valoarea de la ieşire va fi egală cu 1 logic, iar pentru *a=0* această valoare va fi egală cu 0 logic. În cazul când vom atribui celei de-a doua intrări valoarea logică 0, ieşirea porţii ŞI va fi egală cu 0 atât pentru *a=1* cât şi pentru *a=0*, deci poarta nu este sensibilizată.

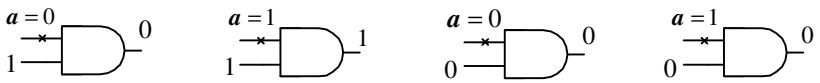


Figura 2.1. Sensibilizarea porţii logice ŞI

Generalizând cele expuse, se poate uşor observa că pentru porţile logice ŞI şi ŞI-NU valorile de intrare pentru sensibilizare sunt egale cu 1 logic, iar pentru porţile SAU şi SAU-NU – cu 0 logic. În cazul porţilor logice SAU Exclusiv (XOR) şi SAU-NU Exclusiv (XNOR) valoarea de intrare pentru sensibilizare poate fi atât 0 cât şi 1 logic. În figura 2.2 sunt prezentate porţile logice cu o intrare presupusă defectă (*a*) şi cu valori de intrare pentru sensibilizare.

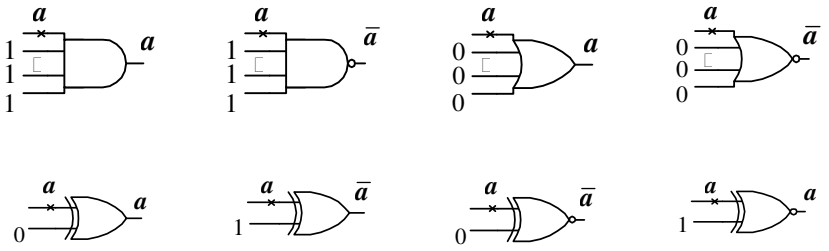


Figura 2.2. Sensibilizarea porţilor logice

În continuare vom analiza procedura de activare a unei căi de propagare a defectului $G_5 \equiv 1$ (figura 2.3).

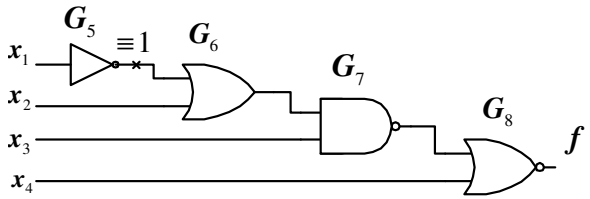


Figura 2.3. O cale de propagare a defectului $G_5 \equiv 1$

Pentru a asigura condiţia de manifestare a defectului $G_5 \equiv 1$ e necesar a obţine $G_5=0$. Pentru aceasta vom considera $x_1=1$. Condiţia de sensibilizare pentru poarta logică G_6 este $x_2=0$, pentru poarta logică G_7: $x_3=1$ şi pentru poarta logică G_8: $x_4=0$.

În urma activării căii (6, 7, 8) am obţinut următorul test:

$$T_{G5\equiv1}=(x_1, x_2, x_3, x_4; f)=(1, 0, 1, 0; 0).$$

Deci, în lipsa defectului $G_5 \equiv 1$, valoarea ieşirii primare *f* va fi egală cu 0 logic ($f=0$), iar în prezenţa defectului: $f=1$. Astfel, defectul $G_5 \equiv 1$ este detectabil.

Să analizăm mai detaliat procedura de generare a testului pentru detectarea defectului $x_1 \equiv 0$ pe calea (5, 7, 9) pentru CLC arbitrar din figura 2.4. Un circuit logic se numeşte *arbitrar* dacă conţine ramificări atât ale intrărilor primare cât şi a conexiunilor interne.

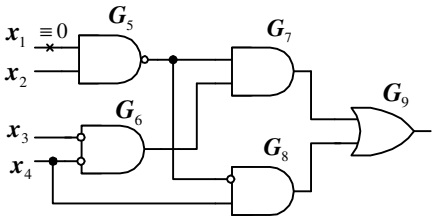


Figura 2.4. Exemplu de circuit arbitrar

Paşii de generare a testului sunt arătaţi în tabelul 2.1. Pentru început se asigură condiţia de manifestare a defectului $x_1 \equiv 0$ prin setarea intrării primare x_1 în 1 logic (pasul 1). Apoi se sensibilizează pe rând porţile G_5, G_7 şi G_9 pentru a activa calea selectată (paşii 2, 3 şi 4). În continuare se trece la faza de consistenţă şi anume se asigură asemenea valori pentru x_3 şi x_4, astfel încât să se producă semnalele dorite la ieşirile porţilor G_8 şi G_6 (paşii 5 şi 6).

Tabelul 2.1. Paşii de generare a unui test

Nr	Intrări primare				Conexiuni interne				Ieşire primară	Comentariu
	x_1	x_2	x_3	x_4	G_5	G_6	G_7	G_8	G_9	
1	**1**									x_1=1
2	1	**1**			0					Sensib. G_5
3					0	**1**	0			Sensib. G_7
4							0	**0**	**0**	Sensib. G_9
5				**0**	0			0		x_4=0
6			**0**	0		1				x_3=0
	1	**1**	**0**	**0**	0	1	0	0	**0**	Testul obţinut

Metoda activării unei căi nu conduce întotdeauna pentru orice tip de circuit la un test de diagnostic. De exemplu, dacă un circuit conţine porţi logice redundante, acestea nu pot fi testate.

Generarea testelor prin metoda activării unei căi este o procedură simplă, care nu necesită calcule voluminoase. Datorită acestui fapt ea poate fi utilizată şi la generarea testelor pentru circuite secvenţiale. Totodată, există şi un dezavantaj semnificativ: activarea unei singure căi nu conduce întotdeauna la elaborarea unui test, care ar putea fi găsit prin activarea simultană a mai multor căi. Un exemplu clasic, în acest sens, este circuitul propus de Schneider (figura 2.5).

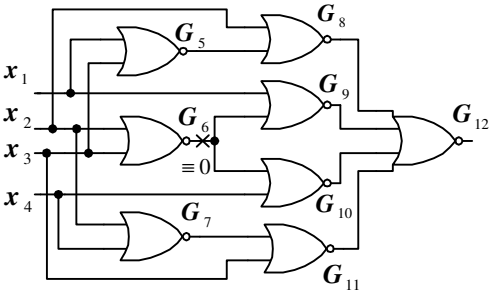


Figura 2.5. Exemplul lui Schneider

Condiţia de manifestare a defectului $G_6 \equiv 0$ este: $x_2=x_3=0$. Condiţia de observabilitate a defectului prin activarea căii (10, 12) este: $x_4=0$ şi $G_8=G_9=G_{12}=0$. Pentru a obţine $G_9=0$ e necesar a seta x_1 în 1 logic, ceea ce va duce la $G_5=0$ şi, deoarece $x_2=0$, vom obţine $G_8=1$. Aceasta este în contradicţie cu condiţia de sensibilizare a porţii logice G_8, şi anume $G_8=0$. La fel, este imposibil a activa calea (9, 12). Totuşi, este evident că atribuind $x_1=x_4=0$ vom putea activa simultan două căi de propagare a defectului $G_6 \equiv 0$, iar testul obţinut va fi T=(0,0,0,0;1).

2.2.1. Exemplu de generare a testelor pentru un CLC arbitrar

Fie dată funcţia booleană $f=\sum(7,15,24,25,26,28,29,30)$.

Pentru început se efectuează minimizarea acestei funcţii utilizând diagrama Karnaugh (figura 2.6).

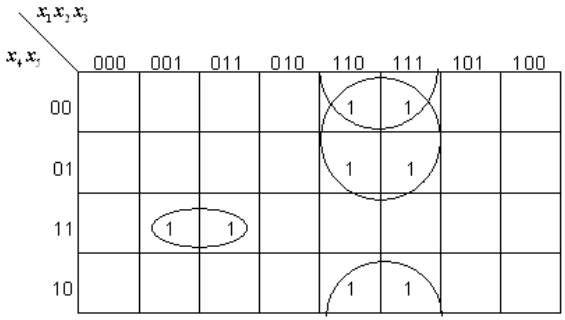


Figura 2.6. Minimizarea funcţiei logice prin diagrama Karnaugh

Pentru a realiza un circuit arbitrar, expresia logică obţinută după minimizare se transcrie în felul următor:

$$F = x_1x_2\bar{x}_4 + x_1x_2\bar{x}_5 + \bar{x}_1x_3x_4x_5 = x_1x_2(\overline{\bar{x}_4 + \bar{x}_5}) + \bar{x}_1x_3x_4x_5 =$$

$$= x_1x_2(\overline{x_4x_5}) + \bar{x}_1x_3x_4x_5$$

Circuitul realizat conform acestei expresii logice, adaptat pentru simularea testelor în sistemul de proiectare digitală Logic Works, este prezentat în figura 2.7.

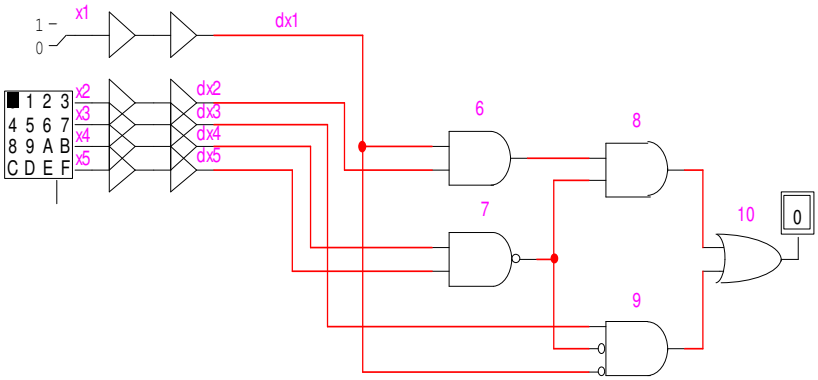


Figura 2.7. CLC arbitrar realizat în Logic Works

Pentru aplicarea testelor se utilizează elementele „Hex Keyboard" şi „Binary Switch". Pentru simularea defectelor se folosesc liniile dintre două elemente de tip „buffer", care pot fi fixate în 1 sau 0 logic.

Testele elaborate sunt prezentate în tabelul 2.2. La elaborarea testelor au fost folosite următoarele notaţii:

- ***** - semnifică faptul că nodul respectiv are o valoare indiferentă (0 sau 1 logic);
- **0/*** şi ***/0** - determină trei combinaţii posibile pe două noduri (01, 10 şi 00).

Un test, în general, poate detecta mai mult decât un singur defect, iar mai multe teste pot detecta acelaşi defect. Astfel, obiectivul major la generarea testelor este minimizarea lor prin determinarea testelor echivalente.

Două teste sunt echivalente dacă:

1) coincid toate valorile definite pentru intrările şi ieşirile primare;

2) coincid toate valorile nedefinite pentru intrările primare (în acest caz în testul rezultant se va alege una din valorile 0 sau 1 pentru valorile nedefinite);

3) valorile definite pentru intrările primare în unul din teste corespund unor valori nedefinite în celălalt test (în acest caz, în testul rezultant se va alege valoarea definită).

Tabelul 2.2. Tabelul inițial al testelor

Nr.	Def.	Intrări primare					Conex. Interne				Ieş. Pr.	Calea
		x_1	x_2	x_3	x_4	x_5	6	7	8	9	10	
1	$x_1 \equiv 0$	**1**	1	*	0/*	*/0	1	1	1	0	1	6,8,10
2	$x_1 \equiv 0$	**1**	*	1	1	1	*	0	0	0	0	9,10
3	$x_1 \equiv 1$	**0**	1	*	0/*	*/0	0	1	0	0	0	6,8,10
4	$x_1 \equiv 1$	**0**	*	1	1	1	0	0	0	1	1	9,10
5	$x_2 \equiv 0$	1	**1**	*	0/*	*/0	1	1	1	0	1	6,8,10
6	$x_2 \equiv 1$	1	**0**	*	0/*	*/0	0	1	0	0	0	6,8,10
7	$x_3 \equiv 0$	0	*	**1**	1	1	0	0	0	1	1	9,10
8	$x_3 \equiv 1$	0	*	**0**	1	1	0	0	0	0	0	9,10
9	$x_4 \equiv 0$	1	1	*	**1**	1	1	0	0	0	0	7,8,10
10	$x_4 \equiv 0$	0	*	1	**1**	1	0	0	0	1	1	7,9,10
11	$x_4 \equiv 1$	1	1	*	**0**	1	1	1	1	0	1	7,8,10
12	$x_4 \equiv 1$	0	*	1	**0**	1	0	1	0	0	0	7,9,10
13	$x_5 \equiv 0$	1	1	*	1	**1**	1	0	0	0	0	7,8,10
14	$x_5 \equiv 0$	0	*	1	1	**1**	0	0	0	1	1	7,9,10
15	$x_5 \equiv 1$	1	1	*	1	**0**	1	1	1	0	1	7,8,10
16	$x_5 \equiv 1$	0	*	1	1	**0**	0	1	0	0	0	7,9,10

Testele minimizate, în care au fost înlocuite toate valorile nedefinite, sunt prezentate în tabelul 2.3.

Corectitudinea testelor obţinute poate fi determinată prin aplicarea lor pentru circuitul realizat în sistemul Logic Works. În figura 2.8 este prezentată diagrama de timp, obţinută în urma aplicării primelor trei teste în cazul absenţei defectelor analizate şi în cazul prezenţei acestor defecte.

Tabelul 2.3. Testele minimizate

Nr.	Nr. testelor iniţiale	Teste cu valori nedefinite	Teste cu toate valorile definite	Defectele detectate
		$x_1,x_2,x_3,x_4,x_5;10$	$x_1,x_2,x_3,x_4,x_5;10$	
1	1,5,11	1, 1, *, 0, 1; 1	1, 1, 1, 0, 1; 1	$x_1\equiv0$, $x_2\equiv0$, $x_4\equiv1$
2	1,5,15	1, 1, *, 1, 0; 1	1, 1, 1, 1, 0; 1	$x_1\equiv0$, $x_2\equiv0$, $x_5\equiv1$
3	2,9,13	1, 1, *, 1, 1; 0	1, 1, 1, 1, 1; 0	$x_1\equiv0$, $x_4\equiv0$, $x_5\equiv0$
4	3,12	0, 1, 1, 0, 1; 0	0, 1, 1, 0, 1; 0	$x_1\equiv1$, $x_4\equiv1$
5	3,16	0, 1, 1, 1, 0; 0	0, 1, 1, 1, 0; 0	$x_1\equiv1$, $x_5\equiv1$
6	4,7,10. 14	0, *, 1, 1, 1; 1	0, 1, 1, 1, 1; 1	$x_1\equiv1$, $x_3\equiv0$, $x_4\equiv0$, $x_5\equiv0$
7	6	1,0,*, 0/*,*/0;0	1, 0, 0, 0, 0; 0	$x_2\equiv1$
8	8	0, *, 0, 1, 1; 0	0, 0, 0, 1, 1; 0	$x_3\equiv1$

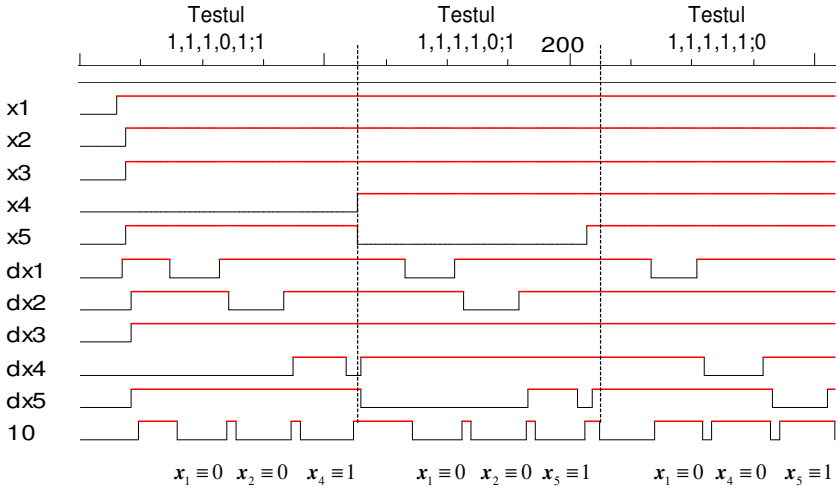


Figura 2.8. Diagrama de timp, obţinută în urma aplicării testelor

Din diagrama de timp se observă că, în prezenţa defectelor ieşirea primară 10 îşi inversează valoarea, deci testele elaborate permit detectarea defectelor intrărilor primare.

2.3. Lucrarea de laborator Nr. 1

Tema: Generarea testelor prin metoda activării unei căi

Scopul lucrării: Însuşirea deprinderilor practice de generare a testelor pentru circuite logice combinaţionale arbitrare prin metoda activării unei căi şi verificarea corectitudinii lor utilizând mijloacele de simulare şi verificare ale sistemului de proiectare digitală Logic Works.

Tema pentru acasă

1. Să se efectueze minimizarea funcţiei logice conform variantei din tabelul 2.4 (la indicaţia profesorului) şi să se transcrie expresia minimizată pentru a realiza un circuit arbitrar.
2. Să se efectueze sinteza circuitului logic arbitrar conform expresiei logice obţinute.
3. Să se elaboreze tabelul iniţial al testelor pentru detectarea defectelor tuturor intrărilor primare pe toate căile posibile prin metoda activării unei căi.
4. Să se determine toate testele echivalente şi să se elaboreze tabelul testelor minimizate, excluzând valorile nedefinite ale intrărilor primare.

Desfăşurarea lucrării

1. Se asamblează circuitul logic, adaptat pentru simularea testelor, în sistemul de proiectare digitală Logic Works.
2. Se verifică corectitudinea circuitului asamblat pentru toate combinaţiile de intrare, obţinându-se diagrama de timp.
3. Se aplică seturile din tabelul testelor minimizate, atât în absenţa cât şi în prezenţa defectelor, obţinându-se diagramele de timp corespunzătoare. Verificarea fiecărui test se va efectua prin inserarea unui singur defect în schema logică la momentul aplicării testului.

Tabelul 2.4. Variantele individuale

Nr.	Funcţia logică
1.	F=∑(8, 9, 10, 11, 12, 13, 22, 30)
2.	F=∑(11, 20, 21, 22, 23, 27, 28, 29)
3.	F=∑(12, 13, 14, 15, 19, 23, 28, 29)
4.	F=∑(5, 13, 24, 25, 26, 27, 28, 30)
5.	F=∑(9, 11, 13, 15, 18, 25, 26, 29)
6.	F=∑(10, 11, 14, 15, 21, 23, 26, 30)
7.	F=∑(1, 5, 9, 13, 17, 21, 30, 31)
8.	F=∑(14, 15, 17, 19, 21, 25, 27, 29)
9.	F=∑(12, 13, 14, 15, 21, 28, 29, 30)
10.	F=∑(8, 9, 12, 13, 23, 24, 28, 31)
11.	F=∑(10, 11, 17, 19, 21, 23, 25, 29)
12.	F=∑(9, 11, 18, 19, 22, 23, 26, 30)
13.	F=∑(4, 5, 6, 7, 12, 13, 27, 31)
14.	F=∑(10, 11, 16, 18, 20, 22, 24, 28)
15.	F=∑(13, 15, 20, 21, 22, 23, 28, 30)
16.	F=∑(1, 5, 9, 13, 17, 21, 28, 29)
17.	F=∑(7, 15, 16, 17, 18, 19, 20, 21)
18.	F=∑(2, 3, 6, 7, 10, 14, 29, 31)
19.	F=∑(4, 5, 6, 7, 12, 14, 25, 29)
20.	F=∑(0, 1, 2, 3, 4, 5, 23, 31)
21.	F=∑(8, 10, 12, 14, 22, 23, 24, 26)
22.	F=∑(4, 6, 12, 14, 20, 22, 26, 27)
23.	F=∑(2, 3, 6, 7, 18, 19, 21, 23)
24.	F=∑(11, 15, 24, 25, 26, 28, 29, 30)
25.	F=∑(0, 2, 4, 6, 16, 18, 22, 23)
26.	F=∑(2, 3, 6, 7, 10, 11, 29, 31)

Întrebări

1. Care sunt cele două condiţii pentru elaborarea testelor de detectare a defectelor singulare în CLC?
2. Enumeraţi etapele de generare a testelor prin metoda activării unei căi.
3. În ce constă sensibilizarea unei porţi logice?

4. Care sunt considerentele de alegere a valorilor de intrare pentru sensibilizarea porţilor logice?
5. Daţi definiţia circuitului arbitrar.
6. Explicaţi de ce nodurile redundante nu pot fi testate.
7. Este posibilă testarea tuturor nodurilor unui circuit realizat direct din FCD (Forma canonică disjunctivă)? Argumentaţi răspunsul.
8. În ce condiţii două teste se consideră echivalente?
9. De ce este necesară minimizarea testelor iniţiale?
10. Câte defecte singulare ale porţii logice ŞI-NU cu patru intrări pot fi detectate prin aplicarea testului (1,1,1,1;0), dar a testului (1,0,1,1;1)?
11. Determinaţi toate testele echivalente pentru detectarea defectelor singulare a intrărilor porţii logice SAU Exclusiv.

2.5. Algoritmul *D*

Algoritmul ***D*** este o metodă structurală de elaborare a testelor, care conduce la obţinerea unui test de diagnosticare a unui defect în termenii intrărilor şi ieşirii porţii defecte, generând simultan toate căile posibile de propagare a defectului la toate ieşirile primare ale circuitului. La fiecare pas al algoritmului se verifică convergenţa căilor, renunţându-se la caile care nu sunt divergente. În final, se urmăresc până la intrările primare toate căile de propagare generate, căutându-se în mod automat valorile logice ale semnalelor de intrare, care evidenţiază manifestarea defectului la ieşirile primare.

Algoritmul ***D*** utilizează noţiunile de *cub singular* sau *cub de definiţie* şi *cub* ***D*** pentru descrierea porţilor logice din circuit.

2.5.1. Cuburi singulare

Funcţionarea porţilor logice poate fi descrisă prin tabele de adevăr. Dacă în aceste tabele valorile de intrare, ce nu influenţează valoarea semnalului de ieşire, se vor considera valori indiferente (notate prin simbolul *), vom obţine un nou tabel. Acest tabel va fi format din cuburi singulare, totalitatea cărora va forma *acoperirea singulară a funcţiei logice,* care descrie comportamentul porţii respective.

De exemplu, pentru determinarea cuburilor singulare a porţii logice ŞI cu două intrări se ţine seama de următoarele considerente:

- este de ajuns ca o singură intrare să fie egală cu 0 pentru a avea la ieşire valoarea 0;

- este necesar ca ambele intrări să fie egale cu 1 pentru a avea la ieşire valoarea 1.

Obţinerea cuburilor singulare din tabelul de adevăr al porţii logice ŞI cu două intrări este prezentată în figura 2.9.

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

$\Rightarrow$

x_1	x_2	y	
0	*	0	CS_1
*	0	0	CS_2 $\leftarrow$ cuburi singulare
1	1	1	CS_3

Figura 2.9. Tabelul de adevăr şi tabelul cuburilor singulare al porţii logice ŞI cu două intrări

Elementele cubului singular sunt denumite coordonate sau noduri. De exemplu, cubul singular 0*0 are nodurile 000 şi 010 cu coordonatele x1, x2 şi y.

Cuburile singulare ale porţilor logice ŞI, SAU, ŞI-NU şi SAU-NU cu două intrări sunt prezentate în tabelul 2.5.

Tabelul 2.5. Tabelul cuburilor singulare pentru porţile logice

	ŞI			ŞI-NU			SAU			SAU-NU		
x_1	0	*	1	0	*	1	1	*	0	1	*	0
x_2	*	0	1	*	0	1	*	1	0	*	1	0
y	0	0	1	1	1	0	1	1	0	0	0	1

2.5.2. Cuburi D de propagare

Pentru a obţine un cub $\boldsymbol{D}$ pentru o poartă logică se intersectează două cuburi singulare cu valori diferite ale ieşirii conform următoarelor reguli:

$$\begin{aligned} &0\,I\;0 = 0\,I\;* = *\,I\;0 = 0 \\ &1\,I\;1 = 1\,I\;* = *\,I\;1 = 1 \\ &*\,I\;* = * \\ &1\,I\;0 = \boldsymbol{D} \\ &0\,I\;1 = \bar{\boldsymbol{D}} \end{aligned} \qquad (2.1)$$

De exemplu, pentru a obţine cuburile $\boldsymbol{D}$ ale porţii logice ŞI (vezi figura 2.9) putem intersecta CS_3 cu CS_1 şi CS_3 cu CS_2:

$$\begin{array}{r} \boldsymbol{CS}_3 = 1\,1\,1 \\ \boldsymbol{CS}_1 = 0 * 0 \\ \boldsymbol{CS}_3 \text{ I } \boldsymbol{CS}_2 = \boldsymbol{D}1\boldsymbol{D} \end{array} \qquad \begin{array}{r} \boldsymbol{CS}_3 = 1\,1\,1 \\ \boldsymbol{CS}_2 = *0\,0 \\ \boldsymbol{CS}_3 \text{ I } \boldsymbol{CS}_2 = 1\boldsymbol{DD} \end{array}$$

Cubul $\boldsymbol{D}$ exprimă dependenţa semnalului de la ieşirea porţii logice faţă de cea aplicată la una din intrările ei. $\boldsymbol{D}$ poate avea două valori: 0 sau 1. De exemplu, cubul $\boldsymbol{D}\,1\,\boldsymbol{D}$ are nodurile 010 şi 111. Aceasta mărturiseşte despre faptul că, dacă poarta e corectă, atunci semnalul de la ieşirea *y* este determinat doar de semnalul de la intrarea x_1, x_2 fiind fixat în 1 logic. Astfel, apariţia defectului $x_1 \equiv 1$ ($x_1 \equiv 0$) este detectată la ieşirea porţii respective.

Dacă semnalul oricărei coordonate a cubului $\boldsymbol{D}$, notat prin $\boldsymbol{D}$, are valoarea 1(0), atunci toate celelalte coordonate, notate prin $\boldsymbol{D}$, vor fi egale cu 1(0), iar semnalele coordonatelor, notate prin $\overline{\boldsymbol{D}}$, vor fi egale cu 0(1).

Se deosebesc *cuburi* $\boldsymbol{D}$ *singulare* şi *cuburi* $\boldsymbol{D}$ *multiple*. În cuburile $\boldsymbol{D}$ singulare doar o singură intrare e notată prin $\boldsymbol{D}$ sau $\overline{\boldsymbol{D}}$. În cuburile $\boldsymbol{D}$ multiple două sau mai multe intrări sunt notate prin simbolurile $\boldsymbol{D}$ sau $\overline{\boldsymbol{D}}$. Necesitatea utilizării cuburilor $\boldsymbol{D}$ multiple se explică prin faptul că la intrările unei porţi logice se pot întâlni mai multe semnale $\boldsymbol{D}$, atunci când în circuit sunt prezente ramificaţii.

Cuburile $\boldsymbol{D}$ ale porţilor logice se mai numesc *cuburi* $\boldsymbol{D}$ *de propagare* (CDP). În tabelul 2.6 sunt prezentate CDP ale porţilor logice ŞI, SAU, ŞI-NU şi SAU-NU cu două intrări.

Tabelul 2.6. Cuburile $\boldsymbol{D}$ de propagare ale porţilor logice

	ŞI				ŞI-NU				SAU				SAU-NU			
x_1	$\boldsymbol{D}$	1	$\boldsymbol{D}$	$\boldsymbol{D}$	$\boldsymbol{D}$	1	$\boldsymbol{D}$	$\boldsymbol{D}$	$\boldsymbol{D}$	0	$\boldsymbol{D}$	$\boldsymbol{D}$	$\boldsymbol{D}$	0	$\boldsymbol{D}$	$\boldsymbol{D}$
x_2	1	$\boldsymbol{D}$	$\boldsymbol{D}$	$\overline{\boldsymbol{D}}$	1	$\boldsymbol{D}$	$\boldsymbol{D}$	$\overline{\boldsymbol{D}}$	0	$\boldsymbol{D}$	$\boldsymbol{D}$	$\overline{\boldsymbol{D}}$	0	$\boldsymbol{D}$	$\boldsymbol{D}$	$\overline{\boldsymbol{D}}$
y	$\boldsymbol{D}$	$\boldsymbol{D}$	$\boldsymbol{D}$	0	$\overline{\boldsymbol{D}}$	$\overline{\boldsymbol{D}}$	$\overline{\boldsymbol{D}}$	1	$\boldsymbol{D}$	$\boldsymbol{D}$	$\boldsymbol{D}$	1	$\overline{\boldsymbol{D}}$	$\overline{\boldsymbol{D}}$	$\overline{\boldsymbol{D}}$	0

2.5.3. Cuburi D ale defectelor (CDD)

Cuburile D ale defectelor permit exprimarea testelor în termenii intrării şi ieşirii porţii defecte. Ele se utilizează atunci când este necesar a testa nodurile interne ale circuitului. În CDD simbolul D e interpretat ca semnal 1 logic pentru starea corectă şi ca semnal 0 logic pentru starea defectă. Simbolul $\bar{D}$ e interpretat invers – 0 logic pentru starea corectă şi 1 logic pentru starea defectă.

CDD pentru un nod blocat la 0 poate fi obţinut prin intersecţia fiecărui cub singular cu valoarea ieşirii egală cu 1 logic pentru poarta corectă cu fiecare cub singular al cărei ieşire este egală cu 0 logic pentru o poartă defectă. În mod similar, CDD pentru un nod blocat la 1 poate fi obţinut prin intersecţia fiecărui cub singular cu valoarea ieşirii egală cu 0 logic pentru poarta corectă cu fiecare cub singular al cărei ieşire este egală cu 1 logic pentru o poartă defectă.

În tabelul 2.7 sunt prezentate CDD ale porţilor logice ŞI, SAU, ŞI-NU şi SAU-NU.

Tabelul 2.7. Cuburile D ale defectelor

Defecte	ŞI			ŞI-NU			SAU			SAU-NU		
	x_1	x_2	y	x_1	x_2	y	x_1	x_2	y	x_1	x_2	y
≡ 0	**1**	**1**	D	**0**	*	D	**1**	*	D	**0**	**0**	D
				*	**0**	D	*	**1**	D			
≡ 1	**0**	*	$\bar{D}$	**0**	**0**	$\bar{D}$	**1**	**1**	$\bar{D}$	**1**	*	$\bar{D}$
	*	**0**	$\bar{D}$							*	**1**	$\bar{D}$

2.5.4. Intersecţia D

Generarea unei căi de propagare a defectului spre ieşirea primară a circuitului este realizată prin intermediul intersecţiei D.

Fie date două cuburi D:

$A = (a_1, a_2, \text{K}\ , a_n)$

$B = (b_1, b_2, \text{K}\ , b_n)$,

unde $a_i, b_i \in \{1, 0, *, D, \bar{D}\}, \quad i = \overline{1, n}$.

Intersecţia $\boldsymbol{D}$ se efectuează doar pentru coordonate identice conform următoarelor reguli:

1) $*\underset{D}{\mathrm{I}}\, \boldsymbol{a_i} = \boldsymbol{a_i}$

$*\underset{D}{\mathrm{I}}\, \boldsymbol{b_i} = \boldsymbol{b_i}$

2) Dacă $\boldsymbol{a_i} \neq *$ şi $\boldsymbol{b_i} \neq *$, atunci

$$\boldsymbol{a_i}\ \underset{D}{\mathrm{I}}\ \boldsymbol{b_i} = \begin{Bmatrix} \boldsymbol{a_i}, & \boldsymbol{pentru} \quad \boldsymbol{a_i} = \boldsymbol{b_i} \\ \varnothing, & \boldsymbol{pentru} \quad \boldsymbol{a_i} \neq \boldsymbol{b_i} \end{Bmatrix}.$$

Intersecţia $\boldsymbol{D}$ reprezintă o formă de descriere a propagării defectului de la nodul analizat spre ieşirea primară a circuitului.

Să analizăm o cale arbitrară, selectată pentru propagarea defectului $x_1 \equiv 0$ (figura 2.10).

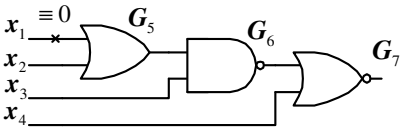


Figura 2.10. O cale de propagare a defectului $x_1 \equiv 0$

Utilizând intersecţia $\boldsymbol{D}$ dintre cuburile $\boldsymbol{D}$ ale porţilor logice din calea respectivă vom obţine cubul $\boldsymbol{D}$ al circuitului (tabelul 2.8).

Tabelul 2.8. Formarea cubului $\boldsymbol{D}$ al circuitului

Cuburi	Coordonate						
	x_1	x_2	x_3	x_4	G_5	G_6	G_7
CDP5	$\boldsymbol{D}$	0	*	*	$\boldsymbol{D}$	*	*
CDP6	*	*	1	*	$\boldsymbol{D}$	$\overline{\boldsymbol{D}}$	*
CDP7	*	*	*	0	*	$\overline{\boldsymbol{D}}$	$\boldsymbol{D}$
CD al circuitului	$\boldsymbol{D}$	0	1	0	$\boldsymbol{D}$	$\overline{\boldsymbol{D}}$	$\boldsymbol{D}$

Cubul $\boldsymbol{D}$ obţinut ($\boldsymbol{D}$, 0, 1, 0, $\boldsymbol{D}$, $\overline{\boldsymbol{D}}$, $\boldsymbol{D}$) verifică conexiunile x_1, G_5, G_6 în baza ieşirii G_7, fiind fixate valorile semnalelor intrărilor primare x_2, x_3, x_4.

Deoarece $\boldsymbol{D}$ poate lua două valori logice – 0 şi 1, cubul $\boldsymbol{D}$ al circuitului se foloseşte la detectarea a două defecte: $x_1 \equiv 0$ şi $x_1 \equiv 1$.

Pentru $x_1 \equiv 0$, considerăm $\boldsymbol{D}=1$ şi obţinem testul:

$$T_{x1\equiv 0}=(x_1, x_2, x_3, x_4; G_7)=(1, 0, 1, 0; 1).$$

Pentru $x_1 \equiv 1$, considerăm $\boldsymbol{D}=0$ şi obţinem testul:

$$T_{x1\equiv 1}=(x_1, x_2, x_3, x_4; G_7)=(0, 0, 1, 0; 0).$$

2.5.5. Etapele algoritmului $\boldsymbol{D}$

Pentru început se determină cuburile singulare (CS) şi cuburile $\boldsymbol{D}$ de propagare (CDP) a fiecărei porţi logice din circuitul logic combinaţional (CLC).

Generarea testelor prin metoda algoritmului $\boldsymbol{D}$ constă din următoarele etape:

1) Construirea cubului $\boldsymbol{D}$ al defectului (CDD);
2) Propagarea defectului prin efectuarea intersecţiei CDD cu CDP a porţilor logice de pe calea aleasă până la ieşirea primară;
3) Verificarea consistenţei (trecerea înapoi) prin intersecţia cubului rezultant cu CS ale porţilor logice, care nu au fost utilizate la prima intersecţie;
4) Repetarea etapelor 1-3 până când se obţin testele pentru toate defectele analizate pe toate căile singulare şi multiple;
5) Minimizarea testelor.

La etapa de verificare a consistenţei se pot obţine elemente vide pe anumite coordonate. În acest caz întreaga intersecţie se consideră vidă şi se renunţă la calea dată, deoarece ea este inconsistentă.

Să analizăm mai detaliat primele trei etape de generare a testelor pentru circuitul logic prezentat în figura 2.11.

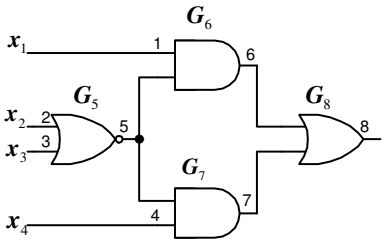


Figura 2.11. Circuit logic

Testul pentru detectarea defectelor intrării primare $x_2 \equiv 0$ şi $x_2 \equiv 1$ pe calea (5,6,7) este prezentat în tabelul 2.9. Pentru a obţine CDD al x_2 vom nota nodul respectiv prin $\boldsymbol{D}$, iar testul obţinut va fi utilizat la detectarea defectului $x_2 \equiv 0$ pentru $\boldsymbol{D}=1$ şi a defectului $x_2 \equiv 1$ pentru $\boldsymbol{D}=0$.

Tabelul 2.9. Detectarea defectelor intrării primare x_2

Et.	Explicaţii	Cub	Coordonate		
			1 2 3 4	5 6 7	8
Et. 1	CDD	C_1	$*\,\boldsymbol{D}\,*\,*$	$*\,*\,*$	$*$
Et. 2	Intersectăm C_1 cu CDP al G_5	C_2	$*\,\boldsymbol{D}\,0\,*$	$\bar{\boldsymbol{D}}\,*\,*$	$*$
	Intersectăm C_2 cu CDP al G_6	C_3	$1\,\boldsymbol{D}\,0\,*$	$\bar{\boldsymbol{D}}\bar{\boldsymbol{D}}\,*$	$*$
	Intersectăm C_3 cu CDP al G_8	C_4	$1\,\boldsymbol{D}\,0\,*$	$\bar{\boldsymbol{D}}\bar{\boldsymbol{D}}\,0$	$\bar{\boldsymbol{D}}$
Et. 3	Intersectăm C_4 cu CS al G_7	C_5	$1\,\boldsymbol{D}\,0\,0$	$\bar{\boldsymbol{D}}\bar{\boldsymbol{D}}\,0$	$\bar{\boldsymbol{D}}$

Din cubul rezultant C_5 se obţin două teste:

$T_{x2\equiv0}=(x_1,x_2,x_3,x_4;8)=(1,1,0,0;0)$ şi

$T_{x2\equiv1}=(x_1,x_2,x_3,x_4;8)=(1,0,0,0;1)$.

Testul pentru detectarea defectului conexiunii interne $G_5 \equiv 0$ pe calea (7,8) este prezentat în tabelul 2.10. Deoarece în CDD ale porţilor logice simbolul $\boldsymbol{D}$ poate lua doar valoarea 1, iar $\bar{\boldsymbol{D}}$ doar valoarea 0, nu este posibil a detecta printr-un singur test defectele $G_5 \equiv 0$ şi $G_5 \equiv 1$, fiind necesare două CDD diferite.

Tabelul 2.10. Detectarea defectului conexiunii interne $G_5 \equiv 0$

Et.	Explicaţii	Cub	Coordonate		
			1 2 3 4	5 6 7	8
Et. 1	CDD	C_1	$*\,0\,0\,*$	$\boldsymbol{D}\,*\,*$	$*$
Et. 2	Intersectăm C_1 cu CDP al G_7	C_2	$*\,0\,0\,1$	$\boldsymbol{D}\,*\,\boldsymbol{D}$	$*$
	Intersectăm C_2 cu CDP al G_8	C_3	$*\,0\,0\,1$	$\boldsymbol{D}\,0\,\boldsymbol{D}$	$\boldsymbol{D}$
Et. 3	Intersectăm C_3 cu CS al G_6	C_4	$0\,0\,0\,1$	$\boldsymbol{D}\,0\,\boldsymbol{D}$	$\boldsymbol{D}$

Din cubul rezultant C_4 se obţine următorul test:

$T_{G5\equiv0}$=(x_1,x_2,x_3,x_4;8)= (0,0,0,1;1).

Pentru a obţine testul detectării defectului $G_5\equiv1$ vom considera CDD=(x_2,x_3,5)=(1,*, $\bar{D}$) sau CDD=(x_2,x_3,5)=(*,1, $\bar{D}$).

Algoritmul D este o metodă bine formalizată şi poate fi uşor programată, ceea ce permite automatizarea procesului generării testelor. În prezent există mai multe modificaţii ale algoritmului D, care permit accelerarea procesului de generare a testelor. De exemplu, procedura PODEM, în care fiecare pas de propagare este urmat de paşii de trecere înapoi până la intrările primare ale circuitului, depistându-se mai repede căile inconsistente. Pe lângă aceasta, spre deosebire de metoda activării unei căi, algoritmul D garantează obţinerea testului, dacă acesta există, datorită faptului că sunt analizate toate căile de propagare a defectului, inclusiv şi cele multiple.

2.5.6. Exemplu de generare a testelor prin metoda algoritmului D pentru un CLC arbitrar

Circuitul logic arbitrar, adaptat pentru simularea testelor în sistemul de proiectare digitală Logic Works, este prezentat în figura 2.12.

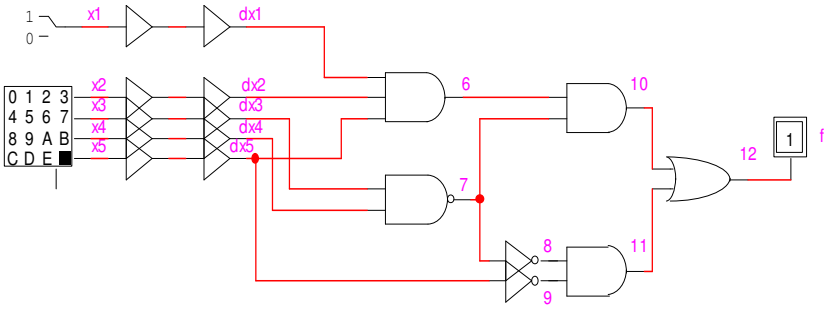


Figura 2.12. CLC arbitrar realizat în Logic Works

Ca şi în cazul metodei activării unei căi, vom elabora teste pentru detectarea defectelor intrărilor primare. În tabelul 2.11 sunt

prezentate cuburile $\boldsymbol{D}$ ale circuitului, obţinute pentru generarea testelor. Două căi de propagare a defectelor din acest tabel vor fi eliminate în cadrul analizei ulterioare, deoarece ele sunt convergente (semnalul $\boldsymbol{D}$ este pierdut în procesul de propagare) şi o cale va fi eliminată din cauza că pe coordonata 7 s-a obţinut un element vid.

Tabelul 2.11. Tabelul cuburilor $\boldsymbol{D}$

Nr	Def.	Intrări primare					Conexiuni interne						Ieş.	Calea
		x1	x2	x3	x4	x5	6	7	8	9	10	11	12	
1	x_1	$\boldsymbol{D}$	1	0/*	*/0	1	$\boldsymbol{D}$	1	0	0	$\boldsymbol{D}$	0	$\boldsymbol{D}$	6,10,12
2	x_2	1	$\boldsymbol{D}$	0/*	*/0	1	$\boldsymbol{D}$	1	0	0	$\boldsymbol{D}$	0	$\boldsymbol{D}$	6,10,12
3	x_3	1	1	$\boldsymbol{D}$	1	1	1	$\overline{\boldsymbol{D}}$	$\boldsymbol{D}$	0	$\overline{\boldsymbol{D}}$	0	$\overline{\boldsymbol{D}}$	7,10,12
4	x_3	0/*	*/0	$\boldsymbol{D}$	1	0	0	$\overline{\boldsymbol{D}}$	$\boldsymbol{D}$	1	0	$\boldsymbol{D}$	$\boldsymbol{D}$	7,8,11,12
5	x_3			$\boldsymbol{D}$	1	0	1	$\overline{\boldsymbol{D}}$	$\boldsymbol{D}$	1	$\overline{\boldsymbol{D}}$	$\boldsymbol{D}$	1	7,8,10-11,12
							Cale convergentă							
6	x_4	1	1	1	$\boldsymbol{D}$	1	1	$\overline{\boldsymbol{D}}$	$\boldsymbol{D}$	0	$\overline{\boldsymbol{D}}$	0	$\overline{\boldsymbol{D}}$	7,10,12
7	x_4	0/*	*/0	1	$\boldsymbol{D}$	0	0	$\overline{\boldsymbol{D}}$	$\boldsymbol{D}$	1	0	$\boldsymbol{D}$	$\boldsymbol{D}$	7,8,11,12
8	x_4			1	$\boldsymbol{D}$	0	1	$\overline{\boldsymbol{D}}$	$\boldsymbol{D}$	1	$\overline{\boldsymbol{D}}$	$\boldsymbol{D}$	1	7,8,10-11,12
							Cale convergentă							
9	x_5	1	1	0/*	*/0	$\boldsymbol{D}$	$\boldsymbol{D}$	1	0	$\overline{\boldsymbol{D}}$	$\boldsymbol{D}$	0	$\boldsymbol{D}$	6,10,12
10	x_5	*	*	1	1	$\boldsymbol{D}$	*	0	1	$\overline{\boldsymbol{D}}$	0	$\boldsymbol{D}$	$\overline{\boldsymbol{D}}$	9,11,12
11	x_5	1	1			$\boldsymbol{D}$	$\boldsymbol{D}$	$\varnothing$	1	$\overline{\boldsymbol{D}}$	0	$\boldsymbol{D}$	$\overline{\boldsymbol{D}}$	6,9,10-11,12
							Cale inconsistentă							

Din cele 8 cuburi rămase, pot fi obţinute 16 teste, înlocuind simbolul $\boldsymbol{D}$ cu valorile 0 şi 1, în dependenţă de defectul analizat (tabelul 2.12).

Tabelul 2.12. Tabelul iniţial al testelor

Nr.	Def	Testul
		x_1, x_2, x_3, x_4, x_5; 12
1	$x_1 \equiv 0$	1, 1, 0/*, */0, 1; 1
2	$x_1 \equiv 1$	0, 1, 0/*, */0, 1; 0
3	$x_2 \equiv 0$	1, 1, 0/*, */0, 1; 1
4	$x_2 \equiv 1$	1, 0, 0/*, */0, 1; 0
5	$x_3 \equiv 0$	1, 1, 1, 1, 1; 0
6	$x_3 \equiv 1$	1, 1, 0, 1, 1; 1

Nr.	Def	Testul
		x_1, x_2, x_3, x_4, x_5; 12
9	$x_4 \equiv 0$	1, 1, 1, 1, 1; 0
10	$x_4 \equiv 1$	1, 1, 1, 0, 1; 1
11	$x_4 \equiv 0$	0/*, */0, 1, 1, 0; 1
12	$x_4 \equiv 1$	0/*, */0, 1, 0, 0; 0
13	$x_5 \equiv 0$	1, 1, 0/*, */0, 1; 1
14	$x_5 \equiv 1$	1, 1, 0/*, */0, 0; 0

7	$x_3 \equiv 0$	0/*, */0, 1, 1, 0; 1	15	$x_5 \equiv 0$	*, *, 1, 1, 1; 0
8	$x_3 \equiv 1$	0/*, */0, 0, 1, 0; 0	16	$x_5 \equiv 1$	*, *, 1, 1, 0; 1

Aceste teste, la rândul său, pot fi minimizate, utilizând regulile descrise în p. 2.2.1, la care se adaugă condiţia de coincidere a simbolurilor $\boldsymbol{D}$ şi $\bar{\boldsymbol{D}}$ pe aceleaşi coordonate. Testele minimizate, în care au fost înlocuite toate valorile nedefinite, sunt prezentate în tabelul 2.13.

Tabelul 2.13. Tabelul testelor minimizate

Nr	Nr. testelor iniţiale	Teste cu toate valorile definite x_1, x_2, x_3, x_4, x_5; 10	Defectele detectate
1	1,3,6,13	1, 1, 0, 1, 1; 1	$x_1 \equiv 0$, $x_2 \equiv 0$, $x_3 \equiv 1$, $x_5 \equiv 0$
2	1,3,10,13	1, 1, 1, 0, 1; 1	$x_1 \equiv 0$, $x_2 \equiv 0$, $x_4 \equiv 1$, $x_5 \equiv 0$
3	2	0, 1, 0, 0, 1; 0	$x_1 \equiv 1$
4	4	1, 0, 0, 0, 1; 0	$x_2 \equiv 1$
5	5,9,15	1, 1, 1, 1, 1; 0	$x_3 \equiv 0$, $x_4 \equiv 0$, $x_5 \equiv 0$
6	7,11,16	0, 0, 1, 1, 0; 1	$x_3 \equiv 0$, $x_4 \equiv 0$, $x_5 \equiv 1$
7	8	0, 0, 0, 1, 0; 0	$x_3 \equiv 1$
8	12	0, 0, 1, 0, 0; 0	$x_4 \equiv 1$
9	14	1, 1, 0, 0, 0; 0	$x_5 \equiv 1$

Corectitudinea testelor obţinute poate fi determinată prin aplicarea lor pentru circuitul realizat în sistemul Logic Works. În figura 2.13 este prezentată diagrama de timp, obţinută în urma aplicării testelor 1, 3, 4 şi 5 în cazul absenţei defectelor analizate şi în cazul prezenţei acestor defecte.

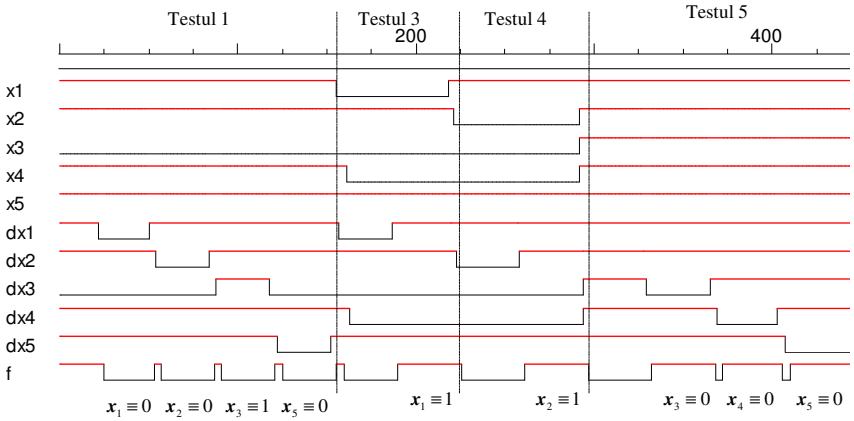


Figura 2.13. Diagrama de timp, obţinută în urma aplicării testelor

2.3. Lucrarea de laborator Nr. 2

Tema: Generarea testelor prin metoda algoritmului *D*

Scopul lucrării: Însuşirea deprinderilor practice de generare a testelor pentru circuite logice combinaţionale arbitrare prin metoda algoritmului ***D*** şi verificarea corectitudinii lor utilizând mijloacele de simulare şi verificare ale sistemului de proiectare digitală Logic Works.

Tema pentru acasă

1. Să se efectueze minimizarea funcţiei logice conform variantei din tabelul 2.4 (la indicaţia profesorului) şi să se transcrie expresia minimizată pentru a realiza un circuit arbitrar.
2. Să se efectueze sinteza circuitului logic arbitrar conform expresiei logice obţinute.
3. Să se elaboreze tabelul cuburilor ***D*** ale circuitului pentru toate căile de propagare.
4. Să se elaboreze tabelul iniţial al testelor pentru detectarea defectelor tuturor intrărilor primare pe toate căile posibile.

5. Să se determine toate testele echivalente şi să se elaboreze tabelul testelor minimizate, excluzând valorile nedefinite ale intrărilor primare.

Desfăşurarea lucrării

1. Se asamblează circuitul logic, adaptat pentru simularea testelor, în sistemul de proiectare digitală Logic Works.
2. Se verifică corectitudinea circuitului asamblat pentru toate combinaţiile de intrare, obţinându-se diagrama de timp.
3. Se aplică seturile din tabelul testelor minimizate, atât în absenţa cât şi în prezenţa defectelor, obţinându-se diagramele de timp corespunzătoare. Verificarea fiecărui test se va efectua prin inserarea unui singur defect în schema logică la momentul aplicării testului.

Întrebări

1. Care sunt deosebirile principale dintre metoda activării unei căi şi metoda algoritmului $\boldsymbol{D}$?
2. Ce reprezintă un cub singular al unei porţi logice şi cum poate fi el obţinut?
3. Explicaţi procedura de obţinere a cubului $\boldsymbol{D}$ pentru o poartă logică.
4. De ce cuburile $\boldsymbol{D}$ se mai numesc cuburi $\boldsymbol{D}$ de propagare?
5. Prin ce se deosebesc cuburile $\boldsymbol{D}$ singulare de cele multiple?
6. În ce constă intersecţia $\boldsymbol{D}$?
7. Ce reprezintă cuburile $\boldsymbol{D}$ ale defectelor şi care sunt valorile atribuite simbolurilor $\boldsymbol{D}$ şi $\overline{\boldsymbol{D}}$ pentru aceste cuburi?
8. Enumeraţi etapele algoritmului $\boldsymbol{D}$. Prin ce se aseamănă şi prin ce se deosebesc ele faţă de etapele metodei activării unei căi?
9. Când o cale se consideră convergentă, dar inconsistentă?

10. Este posibilă detectarea defectelor de pe nodurile redundante ale circuitului prin metoda algoritmului ***D*** ? Argumentaţi răspunsul.
11. Care sunt avantajele algoritmului ***D*** ?
12. Formaţi toate cuburile ***D*** singulare şi multiple pentru poarta logică SAU EXCLUSIV.
13. Formaţi toate cuburile ***D*** singulare şi multiple pentru poarta logică SAU-NU cu trei intrări.
14. Cum este interpretat cubul ***D*** al circuitului în cazul elaborării testelor pentru detectarea defectelor intrărilor primare şi a defectelor conexiunilor interne?
15. Generaţi testele de detectare a defectului 7≡1 din figura 2.12 pe toate căile de propagare.
16. Generaţi testele de detectare a defectului G_6≡0 din figura 2.5 pe toate căile de propagare.
17. Cum poate fi accelerată procedura de generare automată a testelor prin metoda algoritmului ***D*** ?

3. TESTAREA CIRCUITELOR LOGICE SECVENŢIALE

3.1. Noţiuni de bază şi definiţii

Generarea testelor pentru circuitele logice secvenţiale (CLS) este extrem de dificilă, în comparaţie cu generarea testelor pentru CLC, deoarece funcţionarea CLS depinde atât de mulţimea curentă a valorilor de intrare, cât şi de mulţimile anterioare.

Un circuit logic combinaţional realizează o dependenţă a valorilor binare de ieşire numai de variabilele de intrare curente. În CLC nu se ia în considerare variabila timp, cel puţin teoretic, deoarece ea nu există în suportul formal şi anume în algebra booleană.

Din punct de vedere formal, un CLC reprezintă un 3-tuplu: {X, Y, F}, în care semnificaţia obiectelor matematice este următoarea:

- *X* – mulţimea variabilelor de intrare;
- *Y* – mulţimea variabilelor de ieşire;
- F – funcţia de ieşire.

Modelul matematic al CLS reprezintă un automat cu stări finite şi este definit de următorul 5-tuplu: {X, Y, S, F, G}, în care semnificaţia obiectelor matematice este următoarea:

- X – mulţimea variabilelor de intrare;
- Y – mulţimea variabilelor de ieşire;
- S – mulţimea stărilor interne;
- F – funcţia de ieşire, care exprimă procesul de modificare a ieşirilor în dependenţă de variabilele de intrare şi starea internă în momentul de timp curent: $\boldsymbol{Y(t) = F[S(t), X(t)]}$;
- G – funcţia de tranziţie a stărilor, care exprimă procesul de modificare a stărilor în următorul moment de timp în funcţie de variabilele de intrare şi de starea internă în momentul de timp curent: $\boldsymbol{S(t+1) = G[S(t), X(t)]}$.

În figura 3.1 este prezentat modelul general al unui circuit secvenţial. Conform acestui model, un circuit secvenţial este compus din circuite combinaţionale şi bistabile pentru memorarea stării interne. Partea combinaţională a circuitului are două mulţimi de variabile de intrare: primare $\boldsymbol{X(t)}$ (aplicate din exterior) şi secundare (aplicate de la ieşirile bistabilelor). Variabilele secundare de intrare se numesc *variabile de stare*, iar mulţimea variabilelor de stare la momentul de timp t formează *starea curentă* a circuitului $\boldsymbol{S(t)}$.

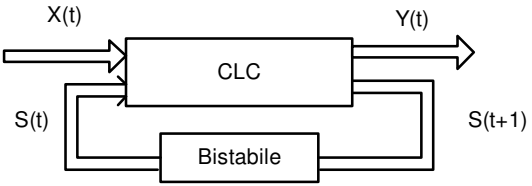


Figura 3.1. Modelul CLS

Un circuit format din m bistabile va avea 2^m stări curente. Ieşirile părţii combinaţionale ale circuitului sunt formate din două mulţimi. Ieşirile primare $\boldsymbol{Y(t)}$, accesibile exteriorului sunt utilizate pentru gestiunea operaţiilor din circuit. Ieşirile secundare se folosesc pentru specificarea stării următoare a circuitului $\boldsymbol{S(t+1)}$. Modelul descris poate fi reprezentat în forma mai multor copii ale aceluiaşi CLS (figura 3.2), astfel încât starea primei copii să servească drept intrare secundară pentru copia a doua, starea celei de-a doua copii să servească drept intrare secundară pentru copia a treia, ş.a.m.d.

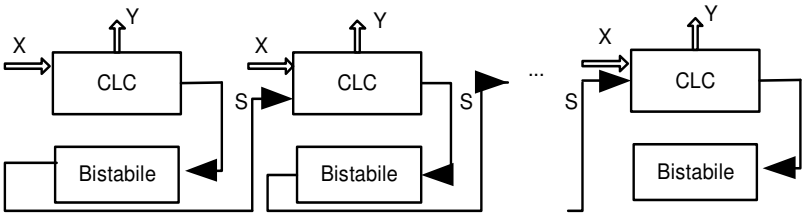


Figura 3.2. Modelul iterativ al CLS

În rezultatul acestei interpretări un CLS poate fi transformat în k CLC, unde k este numărul total de stări ($k=2^m$). CLS, reprezentat astfel poate fi testat, utilizând metodele descrise pentru CLC. Dar, în acest caz, un defect singular va fi prezent în toate cele k copii şi va deveni un defect k-multiplu. Astfel, odată cu creşterea complexităţii CLS, testarea devine practic imposibilă.

O altă abordare pentru efectuarea testării CLS este metoda verificării funcţionale în baza tabelelor de tranziţie a stărilor. În acest caz, sarcina de bază este găsirea unei perechi {X(k), Y(k)}, astfel încât, secvenţa ieşirilor să corespundă Y(k) pentru secvenţa intrărilor X(k), doar în absenţa defectelor.

Atât în cazul primei abordări, cât şi în cazul celei de-a doua, apare problema setării iniţiale a CLS în mod univoc în 1 sau 0 logic, fiind necunoscută valoarea iniţială de la ieşirea circuitului. Pe lângă aceasta, pe parcursul elaborării testelor este necesară aplicarea unor seturi suplimentare de instalare a circuitului într-o valoare sau alta, pentru a fi posibilă detectarea tuturor defectelor. Aceasta măreşte şi mai mult numărul seturilor aplicate la intrarea circuitului, făcând imposibilă testarea circuitelor mari.

Pentru a creşte testabilitatea CLS de tip LSI (Large Scale Integration) şi VLSI (Very Large Scale Integration) sunt prevăzute un şir de măsuri: proiectarea circuitelor astfel, încât să fie posibilă separarea părţii combinaţionale de elementele de memorie şi testarea separată a lor; proiectarea circuitelor autotestabile, în care sunt prevăzute mijloace încorporate de generare a testelor şi tehnici de detectare a defectelor.

3.2. Testarea CLS asincrone cu o buclă de reacţie

Cel mai simplu circuit secvenţial asincron poate fi realizat prin introducerea unei bucle de reacţie între ieşirea circuitului combinaţional şi una dintre intrări (figura 3.3).

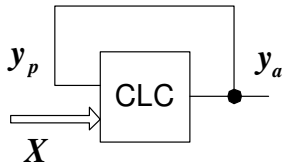


Figura 3.3. CLS asincron cu o buclă

Testarea acestui tip de circuit poate fi efectuată utilizând metodele cunoscute pentru CLC în cazul întreruperii imaginare a buclei de reacţie. Pentru aceasta vom lua în consideraţie valorile variabilei de stare la începutul şi la sfârşitul acestei bucle în momente succesive de timp. În figura 3.3 variabila de stare în momentul de timp actual este notată prin y_a, iar variabila de stare în momentul de timp precedent - prin y_p. Evident, în acest caz, în expresia logică care descrie funcţionarea circuitului va fi prezentă şi variabila y_p.

Etapele de elaborare a testelor pentru CLS cu o buclă de reacţie sunt următoarele:

1) Reprezentarea expresiei logice care descrie funcţionarea circuitului în forma disjunctivă normală (FDN).

2) Aplicarea setului iniţial de instalare univocă a y_a în 0 sau 1 logic, y_p fiind necunoscut.

3) Analizând FDN, se caută un termen pentru care e posibilă instalarea tuturor variabilelor în 1 logic. În acelaşi timp se asigură în ceilalţi termeni cel puţin câte un 0 logic. Aceasta va permite detectarea defectelor „blocaj la 0" pentru variabilele ce sunt prezente în formă directă în termenul ales şi, respectiv, detectarea defectelor „blocaj la 1" pentru variabilele ce sunt prezente în formă inversă.

În absenţa defectelor variabilelor din termenul ales, valoarea y_a va fi egală cu 1 logic, în prezenţa defectelor, valoarea y_a va fi egală cu 0 logic.

Procedura se repetă pentru fiecare termen.

4) Analizând FDN, se caută acei termeni pentru care e posibilă instalarea în 0 logic a unei singure variabile, în ceilalţi termeni

fiind prezente două sau mai multe variabile egale cu 0. Aceasta va permite detectarea defectelor „blocaj la 1” pentru variabilele ce sunt prezente în formă directă în termenii aleşi şi, respectiv, detectarea defectelor „blocaj la 0” pentru variabilele ce sunt prezente în formă inversă.

În absenţa defectelor variabilelor din termenii aleşi, valoarea y_a va fi egală cu 0 logic, în prezenţa defectelor, valoarea y_a va fi egală cu 1 logic.

Procedura se repetă până când toate variabilele funcţiei logice vor fi testate astfel.

La elaborarea testelor se vor respecta următoarele reguli:

1) Valoarea y_p din testul curent va coincide cu valoarea y_a din testul precedent.

2) Atunci când valoarea y_p din setul curent nu permite asigurarea condiţiilor pentru etapele 3 sau 4, se adaugă seturi de instalare a y_a în valoarea necesară.

Spre deosebire de CLC, în cazul testării CLS este importantă ordinea aplicării testelor.

3.2.1. Exemplu de elaborare a testelor pentru CLS cu o buclă de reacţie

Fie dată următoarea expresie logică:

$$y_a = \overline{\overline{x_1 y_p} \cdot \overline{\bar{x}_2 x_3}} \qquad (3.1)$$

CLS cu o buclă de reacţie, elaborat conform acestei expresii şi adaptat pentru efectuarea testării în sistemul Logic Works, este prezentat în figura 3.4.

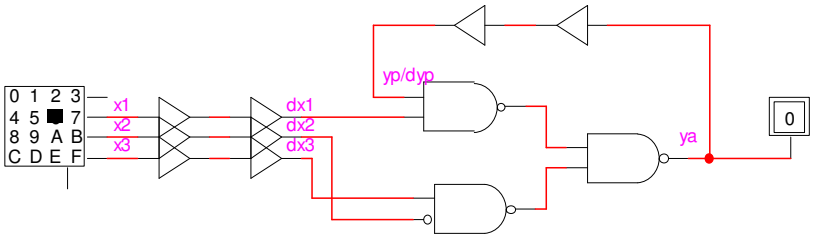


Figura 3.4. CLS cu o buclă de reacţie realizat în Logic Works

Vom reprezenta expresia logică care descrie funcţionarea circuitului în forma disjunctivă normală:

$$y_a = \overline{\overline{x_1 y_p} \cdot \overline{\bar{x}_2 x_3}} = x_1 y_p + \bar{x}_2 x_3 \qquad (3.2)$$

Elaborarea testelor pentru detectarea defectelor variabilelor de intrare şi a variabilei de stare y_p este prezentată în tabelul 3.1.

Tabelul 3.1. Elaborarea testelor pentru CLS cu o buclă de reacţie

Nr	Teste					FDN					Defecte			
	x_1	x_2	x_3	y_p	y_a	$x_1 y_p$		+	$\bar{x}_2$	x_3	x_1	x_2	x_3	y_p
1	0	0	1	*	1	0	*		1	1		≡1	≡0	
2	1	1	*	1	1	1	1		0	*	≡0			≡0
3	0	1	1	1	0	0	1		0	1	≡1	≡0		
4	1	0	0	0	0	1	0		1	0			≡1	≡1

În exemplul analizat setul iniţial instalează semnalul $y_a = 1$. Prin alegerea semnalului $x_1 = 0$, acest set poate servi şi drept test pentru detectarea defectelor $x_2 \equiv 1$ şi $x_3 \equiv 0$.

Testele obţinute în urma înlocuirii tuturor valorilor nedefinite sunt prezentate în tabelul 3.2.

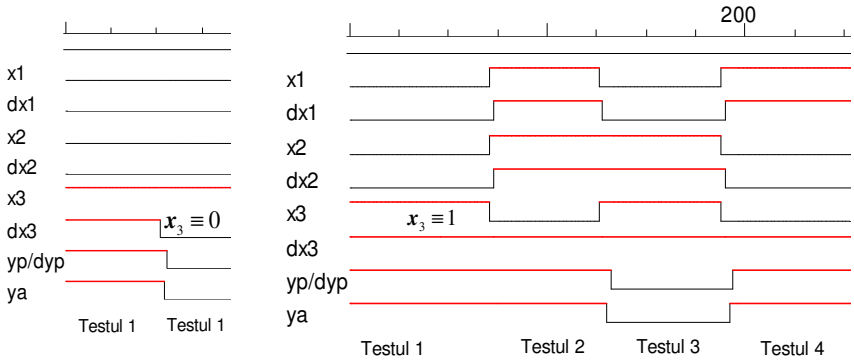
Tabelul 3.2. Testele pentru CLS cu o buclă de reacţie

Nr	Teste					Defecte
	x_1	x_2	x_3	y_p	y_a	
1	0	0	1	*	1	$x_2 \equiv 1$, $x_3 \equiv 0$
2	1	1	0	*1*	1	$x_1 \equiv 0$, $y_p \equiv 0$
3	0	1	1	*1*	0	$x_1 \equiv 1$, $x_2 \equiv 0$
4	1	0	0	*0*	0	$x_3 \equiv 1$, $y_p \equiv 1$

Verificarea testelor poate fi efectuată prin vizualizarea valorii semnalului de stare y_a şi prin obţinerea diagramelor de timp în cazurile absenţei şi prezenţei defectelor considerate. Pentru detectarea fiecărui defect, acesta va fi înserat în circuit după aplicarea setului iniţial. Apoi vor fi aplicate celelalte teste, urmând ordinea din tabelul 3.2, până la testul pentru care valoarea semnalului y_a nu va

corespunde valorii din tabel. În final defectul înserat va fi eliminat. Procedura se va repeta pentru toate defectele analizate.

În figura 3.5 sunt prezentate diagramele de timp obţinute în urma aplicării testelor de detectare a defectelor $x_3 \equiv 0$ (figura 3.5, a) şi $x_3 \equiv 1$ (figura 3.5, b).



(a) (b)

Figura 3.5. Diagramele de timp obţinute în urma aplicării testelor de detectare a defectului $x_3 \equiv 0$ (a) şi a defectului $x_3 \equiv 1$ (b)

După cum se poate observa, pentru detectarea defectului $x_3 \equiv 0$ a fost necesară aplicarea unui singur test, iar pentru detectarea defectului $x_3 \equiv 1$ – a tuturor celor patru teste.

3.3. Testarea CLS asincrone cu mai multe bucle de reacţie

În CLS asincrone cu mai multe bucle de reacţie se pot evidenţia câteva subcircuite combinaţionale, iar numărul variabilelor de stare depinde de numărul buclelor formate. În figura 3.6. este prezentat un CLS asincron cu două bucle de reacţie. Vom utiliza următoarele notaţii: y_{a1} - variabila de stare internă în momentul de timp actual; y_{a2} - variabila de stare externă în momentul de timp actual; y_{p1} - variabila de stare internă în momentul de timp precedent; y_{p2} - variabila de stare externă în momentul de timp precedent.

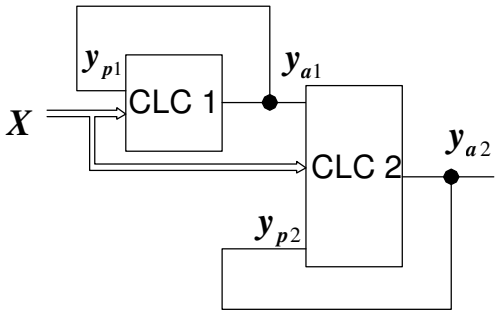


Figura 3.6. CLS cu două bucle de reacţie

Pentru a efectua testarea CLS cu două bucle de reacţie se obţine FDN a expresiei logice pentru variabila de stare externă y_{a2} şi FDN pentru variabila de stare internă y_{a1}. În continuare se analizează condiţiile instalării iniţiale a circuitului, reieşind din faptul că semnalele y_{p1} şi y_{p2} sunt nedefinite. Pentru început se examinează logica funcţionării şi posibilitatea instalării în 0 sau 1 logic a subcircuitului intern, apoi a celui extern. Pentru detectarea defectelor de tip „blocaj la 0" şi „blocaj la 1" a intrărilor circuitului se aplică aceeaşi logică ca şi în cazul CLS cu o buclă de reacţie (etapele 3 şi 4 din subcapitolul 3.2). Pentru a fi posibilă testarea tuturor variabilelor de intrare şi a variabilelor de stare y_{p1} şi y_{p2}, este necesară aplicarea mai multor seturi de setare sau resetare a semnalelor y_{a1} şi y_{a2}.

Una din inconvenientele principale ale CLS asincrone este faptul că ele, în anumite condiţii, pot intra în starea de generare a semnalelor. Aceasta se poate întâmpla în cazul când numărul de inversări în porţile logice din interiorul buclei este impar şi aceste porţi logice sunt sensibilizate (figura 3.7).

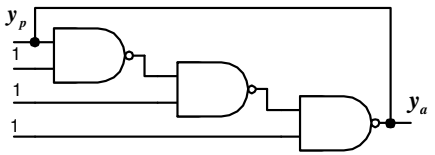


Figura 3.7. CLS în starea de generare a semnalelor

Dacă în rezultatul elaborării testelor se obţine un set ce duce la transformarea circuitului în generator de semnale, se încearcă o altă combinaţie a valorilor de intrare şi în caz de nereuşită se renunţă la detectarea defectului dat.

3.3.1. Exemplu de elaborare a testelor pentru CLS cu două bucle de reacţie

Fie dată următoarea expresie logică:

$$y_{a2} = \overline{\overline{\overline{x}_1 y_{p1}} \cdot x_2 \cdot \overline{x_3 x_4 y_{p2}}} \tag{3.3}$$

Evidenţiem în această expresie bucla de reacţie internă:

$$y_{a1} = \overline{\overline{\overline{x}_1 y_{p1}} \cdot x_2} \tag{3.4}$$

CLS cu două bucle de reacţie, elaborat conform acestei expresii şi adaptat pentru efectuarea testării în sistemul Logic Works, este prezentat în figura 3.8.

Vom reprezenta expresiile logice pentru buclele de reacţie externă (3.3) şi internă (3.4) în forma disjunctivă normală:

$$\begin{aligned} y_{a2} &= \overline{\overline{\overline{x}_1 y_{p1}} \cdot x_2 \cdot \overline{x_3 x_4 y_{p2}}} = \overline{\overline{x}_1 y_{p1}} \cdot x_2 + x_3 x_4 y_{p2} = \\ &= (x_1 + \overline{y}_{p1}) x_2 + x_3 x_4 y_{p2} = x_1 x_2 + x_2 \overline{y}_{p1} + x_3 x_4 y_{p2} \end{aligned} \tag{3.5}$$

$$y_{a1} = \overline{\overline{\overline{x}_1 y_{p1}} \cdot x_2} = \overline{x}_1 y_{p1} + \overline{x}_2 \tag{3.6}$$

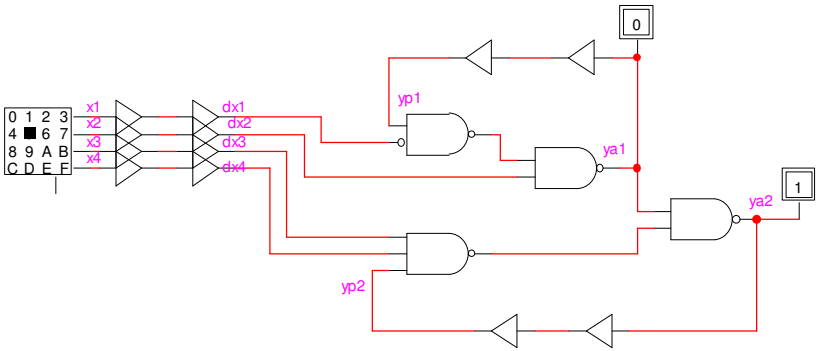


Figura 3.8. CLS cu două bucle de reacţie realizat în Logic Works

Elaborarea testelor pentru detectarea defectelor variabilelor de intrare şi a variabilelor de stare y_{p1} şi y_{p2} este prezentată în tabelul 3.3. Pentru comoditate în tabel este introdusă doar FDN pentru y_{a2}, FDN pentru y_{a1} fiind utilizată implicit pentru calculul valorii variabilei de stare internă.

Tabelul 3.3. Elaborarea testelor pentru CLS cu două bucle de reacţie

Teste								FDN	Defecte					
x_1	x_2	x_3	x_4	y_{p1}	y_{a1}	y_{p2}	y_{a2}	$x_1x_2 + x_2y_{p1} + x_3x_4y_{p2}$	x_1	x_2	x_3	x_4	y_{p1}	y_{p2}
*	0	0	*	*	1	*	0	* 0 0 * 0 * *	y_{a2}=0					
1	1	*	*	1	0	0	1	1 1 1 0 * * 0	≡0	≡0				
0	1	0	*	0	0	1	1	0 1 1 1 0 * 1		≡0			≡1	
*	0	1	1	0	1	1	1	* 0 0 1 1 1 1			≡0	≡0		≡0
0	1	0	1	1	1	1	0	0 1 1 0 0 1 1	≡1		≡1		≡0	
1	0	1	1	1	1	0	0	1 0 0 0 1 1 0		≡1				
1	1	*	*	1	0	0	1	1 1 1 0 * * 0	y_{a2}=1					
*	0	1	0	0	1	1	0	* 0 0 1 1 0 1				≡1		

În exemplul analizat setul iniţial instalează semnalele $y_{a2} = 0$ şi $y_{a1} = 1$. În continuare sunt testate concomitent variabilele pentru care se îndeplinesc condiţiile descrise în subcapitolul 3.2 (etapele 3 şi 4). Pentru a fi posibilă testarea defectului $x_4 \equiv 1$ se introduce un set suplimentar de instalare a semnalului y_{a2} în 1 logic.

Testele obţinute în urma înlocuirii tuturor valorilor nedefinite sunt prezentate în tabelul 3.4.

Nr	Teste						Defecte
	x_1	x_2	x_3	x_4	y_{a1}	y_{a2}	
1	0	0	0	0	1	0	$y_{a2} \equiv 0$
2	1	1	0	0	0	1	$x_1 \equiv 0$, $x_2 \equiv 0$
3	0	1	0	0	0	1	$x_2 \equiv 0$, $y_{p1} \equiv 1$
4	0	0	1	1	1	1	$x_3 \equiv 0$. $x_4 \equiv 0$, $y_{p2} \equiv 0$
5	0	1	0	1	1	0	$x_1 \equiv 1$, $x_3 \equiv 1$, $y_{p1} \equiv 0$

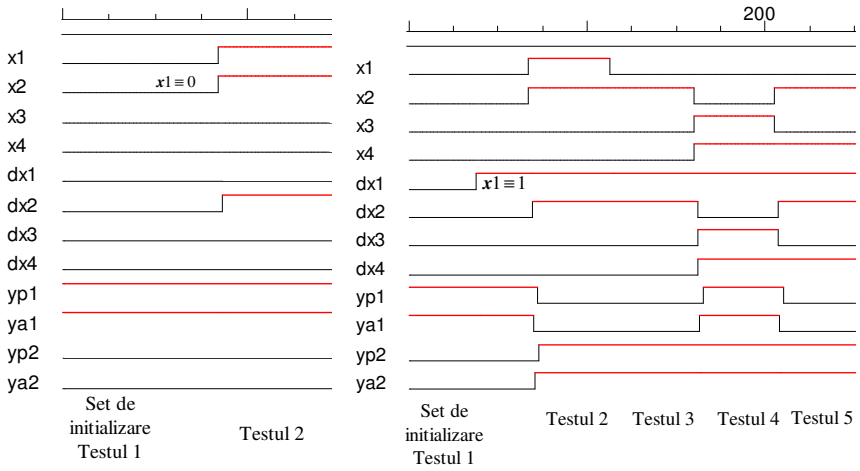
6	1	0	1	1	1	0	$x_2 \equiv 1$, $y_{p2} \equiv 1$
7	1	1	0	0	0	1	$y_{a2}=1$
8	0	0	1	0	1	0	$x_4 \equiv 1$

Tabelul 3.4. Testele pentru CLS cu două bucle de reacţie

Verificarea testelor poate fi efectuată prin vizualizarea valorilor semnalelor de stare y_{a1} şi y_{a2} şi prin obţinerea diagramelor de timp în cazurile absenţei şi prezenţei defectelor considerate. Pentru detectarea fiecărui defect, acesta va fi înserat în circuit după aplicarea setului iniţial. Apoi vor fi aplicate celelalte teste, urmând ordinea din tabelul 3.4, până la testul pentru care valoarea semnalului y_{a2} nu va corespunde valorii din tabel. În final defectul înserat va fi eliminat. Procedura se va repeta pentru toate defectele analizate.

În figura 3.9 sunt prezentate diagramele de timp obţinute în urma aplicării testelor de detectare a defectelor $x_1 \equiv 0$ (figura 3.9, a) şi $x_1 \equiv 1$ (figura 3.9, b). După cum se poate observa, detectarea defectului $x_1 \equiv 0$ a avut loc după aplicarea testului 2, deoarece valoarea semnalului y_{a2} nu a coincis cu valoarea specificată în tabelul 3.4 pentru acest test. Din aceleaşi considerente, detectarea defectului $x_1 \equiv 1$ a avut loc după aplicarea testului 5.

La fel pot fi obţinute şi celelalte teste pentru detectarea defectelor variabilelor x_2, x_3, x_4, y_{p1} şi y_{p2}.



(a) (b)

Figura 3.9. Diagramele de timp obţinute în urma aplicării testelor de detectare a defectului $x_1 \equiv 0$ (a) şi a defectului $x_1 \equiv 1$ (b)

Din cele expuse reiese că metoda elaborării testelor pentru CLS cu bucle de reacţie este bazată pe utilizarea metodelor pentru CLC. Se prevede doar întreruperea buclelor de reacţie şi adăugarea la variabilele de intrare a variabilelor de stare precedente. Dacă nu am întrerupe buclele de reacţie, ar fi necesară aplicarea la intrările circuitului nu doar a tuturor seturilor posibile, ci şi a tuturor succesiunilor de aplicare ale acestor seturi.

3.4. Proiectarea circuitelor testabile

Dificultatea de bază în cazul testării circuitelor secvenţiale mari este setarea şi analiza stărilor bistabilelor, numărul cărora este semnificativ în asemenea circuite. De aceea, luând în considerare imposibilitatea practică de elaborare a testelor pentru CLS mari, se impune modificarea circuitelor în scopul simplificării procedurii de testare. Una dintre primele abordări, cunoscută sub denumirea de metoda căii scanate (Scan Path), constă în separarea părţii combinaţionale de elementele de memorie şi testarea lor separată. Testarea circuitului combinaţional poate fi efectuată prin metode deterministe sau nedeterministe. Pentru testarea elementelor de memorie (bistabilelor), acestea se unesc într-un registru unic de deplasare care, în regim de testare, primeşte anumiţi vectori-test standard de tipul: 100...0, 011...1 ş.a. Astfel, fiecare element de memorie primeşte toate combinaţiile posibile ale variabilei de stare în momentele de timp precedent şi actual.

În prezent există mai multe tehnici de acest gen, toate având drept scop creşterea testabilităţii circuitelor proiectate. Testabilitatea unui circuit constituie aptitudinea acestuia de detectare şi localizare uşoară a defectelor posibile. Testabilitatea se realizează în faza de proiectare, regulile de proiectare urmărind să crească observabilitatea şi controlabilitatea circuitului respectiv.

Proiectarea circuitelor testabile se bazează pe două metode:

1) Proiectarea circuitelor astfel încât să fie uşor testabile. Această metodă poate introduce întârzieri mari în propagarea semnalelor şi o creştere a numărului de porţi logice.

2) Proiectarea circuitelor autotestabile prin introducerea elementelor logice adiţionale. Aceasta conduce la o creştere a complexităţii circuitului, dar reduce numărul de teste necesare pentru diagnoză. Metoda se poate aplica la nivelul de plăci de circuite integrate prin introducerea unor puncte de test suplimentare.

Prima metodă presupune utilizarea tehnicii de proiectare LSSD (Level Sensitive Scan Design) şi este o continuare a metodei căii scanate. Ea este folosită pentru circuitele electronice integrate de tip LSI sau VLSI. Implementarea acestei metode se bazează pe următoarele concepte:

- Toate schimbările de stare a circuitului sunt determinate de nivelul logic al semnalului de test şi nu de fronturile acestuia (Level Sensitive);
- În regim de testare toate elementele de memorie formează un registru unic de deplasare, în care poate fi înscrisă orice stare a circuitului şi care poate fi utilizat pentru testarea părţii combinaţionale a lui (Scan Design).

În acest scop se utilizează o structură logică special proiectată de tip SRL (Shift Register Latch), a cărui schemă bloc este prezentată în figura 3.10. Bistabilul B_1 este utilizat în regim de funcţionare normală şi în regim de testare, bistabilul B_2 este utilizat doar în regim de testare. Bistabilul B_1 în regim de funcţionare normală are intrarea de date pe *D,* semnalul de tact al echipamentului pe intrarea *C* şi ieşirea pe L_1. Pe perioada funcţionării normale semnalele de tact *A* şi *B* sunt fixate în 0 logic. Memorarea datelor din echipament are loc în momentul trecerii semnalului *C* din 1 în 0 logic. În regim de testare semnalul de tact *C* este blocat în 0 logic. Pentru încărcarea datelor de pe intrarea *SDI* (Scan Data Input) în bistabilul B_1, semnalul de tact *A* este instalat în 1 logic. În momentul trecerii acestui semnal în 0, datele se fixează la ieşirea L_1. În acelaşi timp, semnalul de tact *B* se instalează în 1 logic pentru a efectua transferul informaţiei de pe L_1 în bistabilul B_2. Pentru fixarea datelor, la ieşirea L_2 se efectuează trecerea semnalului de tact *B* în 0 logic. Formarea registrului de deplasare în regim de testare are loc prin conexiunea ieşirii L_2 a unei structuri SRL cu intrarea *SDI* a următoarei structuri SRL.

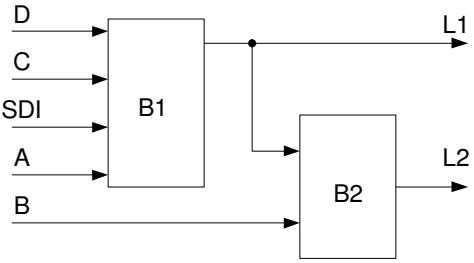


Figura 3.10. Structura SRL

Realizarea unei testări automate în tehnica LSSD presupune următoarele etape:

a) se testează registrele SRL, transferându-se în mod serial o secvenţă logică cunoscută;

b) se testează blocurile combinaţionale, transferându-se la intrarea lor vectorii de test necesari prin intermediul unui registru de deplasare;

c) se citeşte vectorul semnalelor rezultate la ieşirea blocului combinaţional, extrăgându-se informaţia în mod serial prin intermediul registrului de deplasare;

d) se repetă acţiunile de la punctele *b* şi *c* până la expirarea programului de test.

Proiectarea circuitelor LSI şi VLSI autotestabile se bazează pe următoarele principii: secvenţele de test sunt generate nemijlocit în circuit; reacţiile la secvenţele de test se păstrează la fel în circuit; pentru realizarea autotestării e necesar doar a iniţializa procedurile de testare şi a analiza rezultatele.

Generarea pseudoaleatoare a vectorilor de test poate fi realizată pe un registru de deplasare cu bucle de reacţie, ce extrage informaţia din diferite puncte ale registrului şi o transportă la intrarea acestuia. Structura unui astfel de registru, numit LFSR (Linear Feedback Shift Register), este prezentată în figura 3.11.

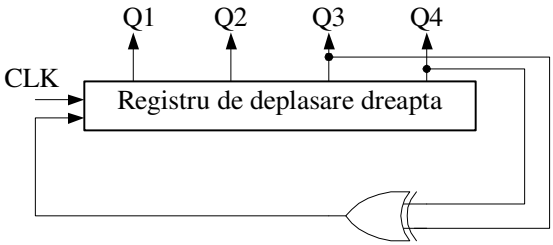


Figura 3.11. Generarea vectorilor de test pe un registru de deplasare

Metoda este simplă, uşor de realizat şi permite obţinerea unui număr de 2^{n-1} vectori de test, unde *n* este numărul de bistabile din registru.

Metodele de analiză a rezultatelor aplicării secvenţei de test constau fie în numărarea tranziţiilor (numărarea trecerii semnalului din 0 în 1 logic şi invers într-un anumit interval de timp), fie în analiza de semnătură (reacţia unică a nodului din circuit la un vector de testare cunoscut). Scopul de bază a acestor metode este comprimarea informaţiei obţinute în urma aplicării secvenţelor de test.

În figura 3.12 este prezentat analizorul de semnătură realizat în baza registrului de deplasare LFSR pe 16 biţi.

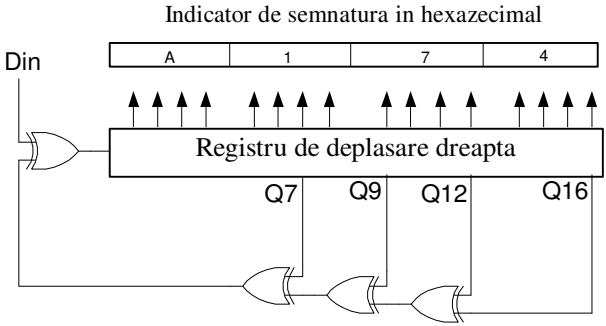


Figura 3.12. Analizor de semnătură

La intrarea registrului de deplasare LFSR se aplică printr-o poartă SAU-EXCLUSIV două surse de informaţie: şirul de *m* biţi rezultaţi în urma aplicării secvenţei de test pentru un nod anumit din circuitul testat şi şirul de semnale de pe bucla de reacţie. La terminarea secvenţei de test, LSFR conţine un cuvânt de 16 biţi, care poartă denumirea de *semnătură* a nodului analizat. Orice defect al nodului va

cauza o eroare la ieşirea lui, care va determina schimbarea conţinutului registrului, iar semnătura rezultată va fi diferită faţă de cea etalon.

Una dintre cele mai răspândite structuri, care permite realizarea unui şir de funcţii necesare pentru implementarea autotestării, este BILBO (Built-In Logic Block Observation). Structura BILBO este realizată în baza unui registru de deplasare cu buclă de reacţie şi mai multe porţi logice. Prin intermediul a patru semnale de tact structura poate trece în următoarele regimuri: resetarea registrului, funcţionare normală, generator de teste şi analizor de semnătură.

3.5. Lucrarea de laborator Nr. 3

Tema: Generarea testelor pentru CLS asincrone cu bucle de reacţie

Scopul lucrării: Însuşirea deprinderilor practice de generare a testelor pentru circuite logice secvenţiale asincrone cu una şi cu două bucle de reacţie şi verificarea corectitudinii lor utilizând mijloacele de simulare şi verificare ale sistemului de proiectare digitală Logic Works.

Tema pentru acasă

1. Să se transforme expresia logică (conform variantei din tabelul 3.5) care descrie funcţionarea CLS asincron cu o buclă de reacţie în forma disjunctivă normală.
2. Să se elaboreze secvenţa de teste pentru detectarea defectelor variabilelor de intrare şi a variabilei de stare.
3. Să se transforme expresia logică care descrie funcţionarea CLS asincron cu două bucle de reacţie în forma disjunctivă normală.
4. Să se elaboreze secvenţa de teste pentru detectarea defectelor variabilelor de intrare şi a variabilelor de stare internă şi externă.

Tabelul 3.5. Variantele individuale

Nr	CLS cu o buclă	CLS cu două bucle

1	$y_a = \overline{\overline{y_p x_1} \cdot x_2}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot x_2} \cdot \overline{x_3 x_4 y_{p2}}}$
2	$y_a = \overline{\overline{y_p x_1} \cdot x_2 x_3}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot x_2 x_3} \cdot \overline{x_4 y_{p2}}}$
3	$y_a = \overline{\overline{y_p x_1} \cdot \overline{x}_2 x_3}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot \overline{x}_2 x_3} \cdot \overline{x_4 y_{p2}}}$
4	$y_a = \overline{\overline{y_p x_1} \cdot x_2 \overline{x}_3}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot x_2 \overline{x}_3} \cdot \overline{\overline{x}_4 y_{p2}}}$
5	$y_a = \overline{\overline{y_p x_1} \cdot x_2 x_3 x_4}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot x_2 x_3 x_4} \cdot \overline{x_5 y_{p2}}}$
6	$y_a = \overline{\overline{y_p x_1} \cdot \overline{x}_2 x_3 x_4}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot \overline{x}_2 x_3 x_4} \cdot \overline{\overline{x}_5 y_{p2}}}$
7	$y_a = \overline{\overline{y_p x_1} \cdot x_2 \overline{x}_3 x_4}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot x_2 \overline{x}_3 x_4} \cdot \overline{\overline{x}_5 y_{p2}}}$
8	$y_a = \overline{\overline{y_p x_1} \cdot x_2 x_3 \overline{x}_4}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot x_2 x_3 \overline{x}_4} \cdot \overline{x_5 y_{p2}}}$
9	$y_a = \overline{\overline{y_p x_1} \cdot \overline{x}_2 x_3}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot \overline{x}_2 x_3} \cdot \overline{x_4 x_5 y_{p2}}}$
10	$y_a = \overline{\overline{y_p x_1} \cdot x_2 x_3}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot x_2 x_3} \cdot \overline{x_4 \overline{x}_5 y_{p2}}}$
11	$y_a = \overline{\overline{y_p x_1} \cdot x_2 \overline{x}_3}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot x_2 \overline{x}_3} \cdot \overline{\overline{x}_4 y_{p2}}}$
12	$y_a = \overline{\overline{y_p x_1} \cdot x_2 x_3 x_4}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot x_2 x_3 x_4} \cdot \overline{\overline{x}_5 y_{p2}}}$
13	$y_a = \overline{\overline{y_p x_1} \cdot \overline{x}_2 x_3 x_4}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot \overline{x}_2 x_3 x_4} \cdot \overline{x_5 y_{p2}}}$
14	$y_a = \overline{\overline{y_p x_1} \cdot x_2 \overline{x}_3 x_4}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot x_2 \overline{x}_3 x_4} \cdot \overline{\overline{x}_5 y_{p2}}}$
15	$y_a = \overline{\overline{y_p x_1} \cdot x_2 x_3 \overline{x}_4}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_1} \cdot x_2 x_3 \overline{x}_4} \cdot \overline{x_5 y_{p2}}}$
16	$y_a = \overline{\overline{y_p x_3} \cdot x_1 x_2}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_3} \cdot x_1 x_2} \cdot \overline{\overline{x}_4 \overline{x}_5 y_{p2}}}$
17	$y_a = \overline{\overline{y_p x_2} \cdot \overline{\overline{x}_1 x_3}}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_2} \cdot \overline{\overline{x}_1 x_3} \cdot \overline{x_4 y_{p2}}}}$
18	$y_a = \overline{\overline{y_p x_4} \cdot \overline{\overline{x}_1 x_2 \overline{x}_3}}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_4} \cdot \overline{\overline{x}_1 x_2 \overline{x}_3} \cdot \overline{x_5 y_{p2}}}}$

Tabelul 3.5. Variantele individuale (Continuare)

Nr	CLS cu o buclă	CLS cu două bucle

19	$y_a = \overline{\overline{y_p \overline{x}_2} \cdot \overline{x}_1}$	$y_{a2} = \overline{\overline{\overline{y_{p1} \overline{x}_2} \cdot \overline{x}_1 \cdot x_3 \overline{x}_4} \, \overline{y_{p2}}}$
20	$y_a = \overline{\overline{y_p x_3} \cdot \overline{x}_1 x_2}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_3} \cdot \overline{x}_1 x_2} \cdot \overline{\overline{x}_4 y_{p2}}}$
21	$y_a = \overline{\overline{y_p x_2} \cdot x_1 \overline{x}_3}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_2} \cdot x_1 \overline{x}_3} \cdot \overline{\overline{x}_4 x_5 y_{p2}}}$
22	$y_a = \overline{\overline{y_p \overline{x}_4} \cdot x_1 x_2 x_3}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_4} \cdot x_1 x_2 x_3} \cdot \overline{x_5 y_{p2}}}$
23	$y_a = \overline{\overline{y_p x_3} \cdot \overline{x}_1 \overline{x}_2 x_4}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_3} \cdot \overline{x}_1 \overline{x}_2 x_4} \cdot \overline{\overline{x}_5 y_{p2}}}$
24	$y_a = \overline{\overline{y_p \overline{x}_2} \cdot x_1 \overline{x}_3 x_4}$	$y_{a2} = \overline{\overline{\overline{y_{p1} \overline{x}_2} \cdot x_1 \overline{x}_3 x_4} \cdot \overline{\overline{x}_5 y_{p2}}}$
25	$y_a = \overline{\overline{y_p x_4} \cdot x_1 \overline{x}_2 \overline{x}_3}$	$y_{a2} = \overline{\overline{\overline{y_{p1} x_4} \cdot x_1 \overline{x}_2 \overline{x}_3} \cdot \overline{x_5 y_{p2}}}$
26	$y_a = \overline{\overline{y_p \overline{x}_2} \cdot \overline{x_1 \overline{x}_3}}$	$y_{a2} = \overline{\overline{\overline{y_{p1} \overline{x}_2} \cdot \overline{x_1 \overline{x}_3}} \cdot \overline{x_4 x_5 y_{p2}}}$

Desfăşurarea lucrării

1. Se asamblează CLS asincron cu o buclă de reacţie în setul ŞI-NU, adaptat pentru simularea testelor, în sistemul de proiectare digitală Logic Works.
2. Se verifică corectitudinea secvenţei de teste elaborate în cazul absenţei defectelor, obţinându-se diagrama de timp.
3. Se aplică secvenţa de teste, înserându-se pe rând toate defectele considerate, obţinându-se diagramele de timp corespunzătoare.
4. Se asamblează CLS asincron cu două bucle de reacţie în setul ŞI-NU, adaptat pentru simularea testelor, în sistemul de proiectare digitală Logic Works.
5. Se verifică corectitudinea secvenţei de teste elaborate în cazul absenţei defectelor, obţinându-se diagrama de timp.
6. Se aplică secvenţa de teste, înserându-se pe rând toate defectele considerate, obţinându-se diagramele de timp corespunzătoare.

Întrebări

1. Prin ce se deosebesc circuitele logice combinaţionale de cele secvenţiale?
2. Ce caracteristică a CLS determină dependenţa acestor circuite de variabila timp?
3. În ce caz testarea CLS poate fi efectuată prin aplicarea metodelor de testare a CLC?
4. Cum poate fi obţinut modelul iterativ al CLS?
5. De ce un defect singular devine un defect *k*-multiplu în modelul iterativ al CLS?
6. Care sunt abordările de bază pentru efectuarea testării CLS?
7. Care sunt dificultăţile principale în cazul testării CLS mari?
8. Ce măsuri pot fi întreprinse pentru a creşte testabilitatea CLS?
9. Numiţi etapele testării CLS cu o buclă de reacţie.
10. Când este necesară adăugarea seturilor de instalare a variabilei de stare?
11. Explicaţi de ce în cazul testării CLS este importantă ordinea aplicării testelor.
12. În ce condiţii un CLS asincron poate intra în starea de generare a semnalelor?
13. Numiţi metodele de proiectare a circuitelor testabile.
14. Ce concepte stau la baza implementării metodei LSSD?
15. Ce reprezintă structura logică SRL şi unde este ea utilizată?
16. Cum are lor formarea registrului de deplasare în baza structurilor SRL?
17. Numiţi etapele testării automate în baza tehnicii LSSD.
18. Care sunt principiile de proiectare a circuitelor LSI şi VLSI autotestabile?
19. Generaţi toţi vectorii de test în baza registrului LFSR din figura 3.11, dacă starea iniţială a registrului este 0001.
20. Generaţi toţi vectorii de test în baza registrului LFSR din figura 3.11, dacă starea iniţială a registrului este 1000.
21. Explicaţi funcţionarea analizorului de semnătură realizat în baza registrului de deplasare LFSR pe 16 biţi din figura 3.12.

4. CODURI DETECTOARE ŞI CORECTOARE DE ERORI

4.1. Redundanţa

În sistemele de calcul contemporane cea mai mare parte a erorilor sunt tranziente. Principalele cauze ale erorilor tranziente sunt perturbaţiile: zgomotul, interferenţa semnalelor şi fluctuaţia tactului de sondare în transmisiunile de date. Aceste erori, spre deosebire de erorile permanente, sunt cu mult mai greu de depistat din cauza caracterului temporar. Coeficientul de eroare pentru un circuit dat depinde de mai mulţi factori, cum ar fi: tipul circuitului, debitul datelor. În mod uzual coeficientul de eroare variază între 10^{-4} si 10^{-6}.

Sistemele fiabile, tolerante la astfel de erori, pot fi construite doar folosind **redundanţa**. Redundanţa se poate defini ca rezultatul încorporării în sistem, prin proiect, a unor module suplimentare, în aşa mod încât funcţia sistemului să nu fie periclitată de apariţia unei disfuncţii. Se deosebesc următoarele tipuri de redundanţe:

1) *Redundanţă hardware* - foloseşte mai multe componente decât strictul necesar pentru a implementa un anumit sistem. Resursele adiţionale fac calcule suplimentare şi rezultatele sunt comparate între ele. În general, cu cît redundanţa unui sistem este mai mare, cu atât se pot detecta sau tolera mai multe erori.

2) *Redundanţă software* - se asigură prin echipe de programare multiple. Se scriu versiuni diferite de software pentru aceeaşi funcţie, pentru aceeaşi aplicaţie.

2) *Redundanţă temporală* – utilizează un singur dispozitiv pentru a calcula acelaşi lucru în mod repetat, după care rezultatele se compară între ele.

3) *Redundanţă informaţională* - se realizează prin adăugarea de biţi suplimentari la cei originali. Erorile în biţi pot fi detectate şi chiar corectate.

Costul redundanţei hardware este substanţial, deoarece este necesară replicarea identică a unui întreg sistem. De exemplu, redundanţa modulară triplă, propusă de John von Neumann în 1956, are o eficienţă de 33%. Redundanţa temporală duce, cel puţin, la dublarea timpului necesar pentru efectuarea calculelor sau pentru transmiterea datelor.

Redundanţa informaţională oferă metode de protecţie a datelor împotriva erorilor, folosind mai puţine resurse suplimentare şi implică utilizarea codurilor detectoare şi corectoare de erori. Deschizători de

drumuri, în această privinţă, au fost Claude Shannon şi Richard Hamming, spre sfârşitul anilor 40.

4.2. Metode de protecţie a datelor împotriva erorilor

Metodele de protecţie a datelor împotriva erorilor introduse de canalul de transmisiune implică folosirea unui codor în transmiţător, a unui decodor în receptor şi a unei strategii de control al erorii. Strategia de utilizare a codorului şi decodorului depinde de ansamblul sistemului de comunicaţii de date considerat. Această strategie poate fi o *simplă detecţie a erorilor*, entitatea care primeşte datele fiind informată despre blocurile de date recepţionate cu erori. Alte strategii urmăresc *corectarea erorilor* si, pentru acestea, se disting doua cazuri:

1) detectarea blocurilor de date recepţionate eronat şi *corectarea erorilor prin retransmiterea* acestor blocuri;

2) *corectarea directă*, la recepţie, a erorilor.

Strategia corectării directe a erorilor, notată şi FEC după denumirea în limba engleză (Forward Error Correction), necesită utilizarea unor coduri corectoare de erori. Celelalte strategii, de detecţie simplă a erorilor sau de corectare prin retransmitere, necesită utilizarea unor coduri detectoare de erori.

Strategia de corectare prin retransmitere este notată ARQ (Automatic Repeat Request). Corectarea erorilor prin retransmitere este mai simplă în ceea ce priveşte complexitatea decodorului, dar necesită doua căi de transmisiune: una pentru a transmite mesajele de informaţie (blocurile de date) şi alta, în sens invers, pentru a transmite confirmările de recepţie (pozitivă, fără erori şi negativă, cu erori). În plus, corectarea erorilor se face cu o anumită întârziere.

4.3. Noţiuni de bază din teoria codurilor

Fie dat un cod binar cu lungimea de n simboluri. Numărul maxim de combinaţii de simboluri care pot fi obţinute este de 2^n. Dacă toate aceste combinaţii sunt cuvinte de cod valide, formate doar din simboluri informaţionale, nu va exista posibilitatea de a detecta sau corecta erorile, ce apar în procesul de prelucrare sau transmitere a datelor. Practic, dacă un cuvânt de cod se modifică, datorită perturbaţiilor, se va obţine tot un cuvânt de cod valid.

Un asemenea cod, unde toate simbolurile sunt informaţionale, se numeşte **cod simplu** sau **cod iredundant.**

Pentru a avea posibilitatea de a detecta prezenţa erorilor în secvenţele de cod recepţionate, mulţimea cuvintelor se va divide în două submulţimi: cuvintele de cod valide şi combinaţiile nevalide. Pentru aceasta se adaugă o anumită informaţie redundantă, de obicei prin introducerea unor simboluri suplimentare, numite *simboluri de control*. Rolul acestor simboluri de control este a indica utilizatorului prezenţa erorilor şi, în unele cazuri, de a le corecta.

Codurile, în care în afară de n simboluri informaţionale se folosesc şi k simboluri de control, se numesc **coduri redundante**. În asemenea coduri pot fi obţinute 2^{n+k} combinaţii de simboluri, dintre care doar 2^n reprezintă cuvinte de cod valide. Dacă cele $n+k$ simboluri primite nu constituie un cuvânt de cod valid, înseamnă că s-au detectat erori. Pentru anumiţi algoritmi de codare, unele tipuri de erori pot fi nu numai detectate, ci pot fi chiar corectate la destinaţie, de aceea aceste coduri se mai numesc **coduri detectoare şi corectoare de erori.**

Se poate face o clasificare a codurilor detectoare şi corectoare de erori după modul de prelucrare al simbolurilor. Dacă prelucrările necesare obţinerii proprietăţilor de detecţie sau de corecţie se fac în blocuri de n simboluri, avem *coduri bloc*. Dacă prelucrarea simbolurilor generate de sursă se realizează în mod continuu, avem de-a face cu *coduri convoluţionale (recurente)*.

Din categoria codurilor bloc se disting:

1) *codurile grup (coduri liniare)*, secvenţele de cod sunt considerate ca fiind elemente dintr-un spaţiu vectorial;

2) *codurile ciclice,* secvenţele de cod sunt considerate ca fiind elemente într-o algebră.

O noţiune importantă în teoria codurilor o reprezintă **distanţa Hamming.** Distanţa Hamming între două cuvinte de cod este dată de numărul de poziţii binare, prin care cele două cuvinte diferă. Evident distanţa Hamming se poate defini numai pentru cuvinte de lungimi identice. În figura 4.1 distanţa Hamming dintre două cuvinte, care sunt separate de o muchie, este 1. Două cuvinte separate prin două muchii (ca cel mai scurt drum parcurs de la unul la altul) sunt distanţate Hamming cu 2. Cea mai mare distanţă Hamming, 3, este între cuvintele situate în extremităţile unei diagonale a cubului.

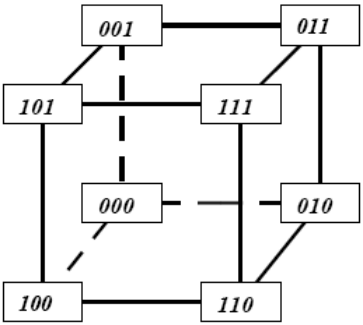


Figura 4.1. Reprezentarea geometrică a unui cod de 3 simboluri

Distanţa Hamming dintre două cuvinte binare se calculează prin efectuarea adunării modulo 2 dintre biţii acestor cuvinte. Spre exemplu, distanţa Hamming dintre cuvintele binare 011011 şi 110111 este egală cu trei:

$$\begin{array}{r} 011011 \\ \oplus\ 110111 \\ \hline 101100 \end{array}$$

Se numeşte **distantă minimă ($\mathbf{d_{min}}$)** a unui cod cea mai mică distantă Hamming între două cuvinte distincte ale codului.

Un cod se zice că poate detecta toate combinaţiile de t erori, dacă pentru fiecare cuvânt transmis a şi fiecare cuvânt recepţionat b, obţinut prin alterarea a t biţi în a, b nu este un cuvânt de cod.

Proprietatea 1: Un cod detectează toate cuvintele în care t biţi sunt eronaţi dacă distanţa lui minimă este superioară lui t, $d_{min} > t$.

Proprietatea 2: Un cod poate corecta la receptie t biti eronaţi, dacă distanţa lui minimă este mai mare ca $2t$, $d_{min} > 2t$.

În general, capacitatea codului a detecta şi a corecta erorile este determinată de următoarea relaţie:

$$\mathbf{d_{min}=r+s+1,}$$

unde **r** este numărul erorilor detectate şi **s** este numărul erorilor corectate.

Deci, un cod poate *detecta* o eroare singulară pentru $d_{min}=2$. Un cod poate *detecta şi corecta* o eroare singulară pentru $d_{min}=3$. Pentru codurile simple $d_{min}=1$.

4.4. Coduri cu paritate

Codurile cu paritate se utilizează la detectarea erorilor la transmiterea datelor (protocolul COM, RS) , în circuitele de memorare (memorie cu paritate), în operaţiile aritmetice, ş. a.

Ele sunt bazate pe o metodă foarte simplă de codare, care constă în completarea cuvintelor de cod cu un simbol de control, numit bit de paritate. Pentru un cod cu paritate cu soţ (sau fără soţ), bitul suplimentar este fixat astfel, ca numărul total de unităţi binare din cuvântul de cod să fie totdeauna par (sau impar).

Un cod cu paritate are $d_{min}=2$ şi este capabil a detecta erorile care apar în număr impar într-un cuvânt de cod. Dacă un bit se schimbă din 0 în 1 sau invers, paritatea se schimbă şi eroarea poate fi detectată prin verificarea parităţii. Această paritate simplă nu poate însă corecta bitul eronat.

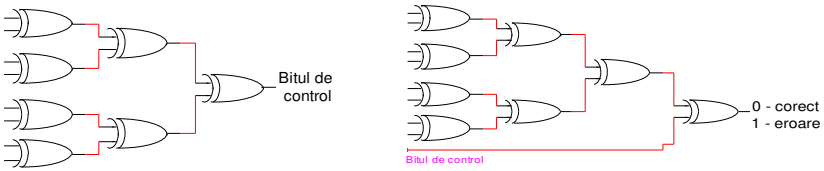
Lungimea cuvintelor de cod este de $n=n_0+1$, unde n_0 este numărul simbolurilor informaţionale.

4.4.1. Circuite de codare si de decodare pentru coduri cu paritate

Codorul este un sumator de biţi modulo-2. Acesta generează un 0 dacă numărul de unităţi binare din cuvântul format din simboluri informaţionale este cu soţ şi un 1 dacă numărul de unităţi binare a acestui cuvânt este fără soţ. Ieşirea sumatorului este semnalul de paritate. În figura 4.2 (a) este prezentat un codor pentru un cod cu paritate cu soţ pentru $n_0=8$.

Decodorul regenerează bitul de paritate din biţii care constituie partea de date în cuvântul recepţionat si îl compară cu bitul de paritate primit. În figura 4.2 (b) este prezentat un decodor pentru un cod cu paritate cu soţ cu $n_0=8$. Dacă există potrivire între cei doi biţi de paritate, cel evaluat şi cel primit, ieşirea porţii SAU-EXCLUSUV din ultimul nivel este un 0, ceea ce indică absenţa erorii. Lipsa coincidenţei produce în acelaşi punct al schemei un 1 ceea ce indică o eroare.

Erorile pe doi biţi nu pot fi detectate de un cod cu paritate, deoarece paritatea nu este modificată prin inversarea a doi biţi. Toate erorile de trei biţi într-un cuvânt, desigur o situaţie mult mai rară, sunt detectate fără a putea discerne dacă este vorba de una sau trei erori în biţii cuvântului de cod.



(a) (b)

Figura 4.2. Codorul (a) şi decodorul (b) pentru un cod cu paritate cu soţ

După cum s-a amintit mai devreme, paritatea utilizată poate fi cu soţ sau fără soţ. Când se alege una şi când se alege cealaltă? Uzual este preferată paritatea cu soţ, dar decizia depinde uneori de tipul de erori pe toţi biţii, care este mai probabil. Pentru paritatea pară, bitul de paritate pentru toţi biţii nuli este 0 şi o cădere de tipul *toţi-biţii*-0 trece neobservată, deoarece apare deghizată ca un cuvânt de cod valid. Prin alegerea tipului de paritate impar, eroarea de tipul *toţ-biţii*-0 devine detectabilă. Dacă eroarea de tipul *toţi-biţii*-1 este mai probabilă, atunci, după caz, dacă numărul total de biţi (n_0 + 1) este par trebuie selectată paritatea impară, iar dacă (n_0 + 1) este un număr impar trebuie reţinută şi utilizată paritatea pară.

4.4.2. Paritate încrucişată

O creştere a capacităţii de detectare se poate obţine prin **metoda parităţii încrucişate,** care se aplică unor blocuri de date. În schema cea mai simplă, informaţia este organizată într-o matrice bidimensională. Biţii de la finele unei linii sunt biţi de paritate pe acea linie (paritate transversală), biţii de la baza unei coloane sunt biţi de paritate pe acea coloană (paritate longitudinală). O eroare pe un bit, situată oriunde în matricea utilizată, afectează corectitudinea parităţii atât pe o linie cât şi pe o coloană. Identificarea liniei şi coloanei cu bitul eronat nu este o problemă, iar corectarea lui este imediată.

Controlul erorilor pe principiul parităţii încrucişate poate conduce la:

1) depistarea erorilor atât prin paritate transversală cât şi prin paritate longitudinală;

2) depistarea numai prin paritate transversală sau longitudinală;

3) erorile să nu fie depistate.

În tabelul 4.1 este prezentat un exemplu de utilizare a metodei de paritate încrucişată pentru codarea unui bloc de date.

Tabelul 4.1. Codarea unui bloc de date prin metoda parităţii încrucişate

Secvenţa	Biţi informaţionali								Control coloană
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	l_i
s_1	0	1	1	0	1	1	1	0	1
s_2	1	0	1	0	1	0	0	1	0
s_3	1	1	1	0	1	0	1	0	1
s_4	0	0	0	0	0	1	1	1	1
Control linie c_j	0	0	1	0	1	0	1	0	0

La recepţie se calculează încă o dată paritatea transversală şi longitudinală a informaţiei transmise. Comparând parităţile recepţionate cu cele calculate, se poate afirma că blocul de informaţie a fost transmis: *fără erori*, dacă parităţile recepţionate şi calculate coincid; *cu erori*, dacă cel puţin pe o coloană şi/sau pe o linie parităţile recepţionate şi calculate nu coincid.

În tabelul 4.2 este prezentat modul de verificare a informaţiei transmise prin metoda parităţii încrucişate.

Tabelul 4.2. Verificarea informaţiei prin metoda parităţii încrucişate

Secvenţa	Informaţia recepţionată									l_i calculat
	x_1'	x_2'	x_3'	x_4'	x_5'	x_6'	x_7'	x_8'	l_i'	
s_1	0	1	1	0	1	1	1	0	1	1
s_2	1	0	1	0	1	0	0	1	0	0
s_3	1	1	1	0	1	1	1	0	1	0
s_4	0	0	0	0	0	1	1	1	1	1
c_j'	0	0	1	0	1	0	1	0	0	0
c_j calculat	0	0	1	0	1	1	1	0	0	

Din tabel se observă că $l_3' \neq l_3$ calculat şi $c_6' \neq c_6$ calculat. Deci, eroarea este în secvenţa s_3 pe poziţia x_6'.

4.5. Coduri Hamming

Codurile Hamming alcătuiesc o clasă importantă de coduri grup pentru detectarea şi corectarea unei erori.

Un cod Hamming *(n,n₀)* este format din cuvinte de cod cu lungimea de *n* simboluri, dintre care n_0 simboluri sunt informaţionale şi $k=n-n_0$ simboluri sunt de control.

Codul Hamming este singurul cod grup perfect corector de o eroare. Un *cod perfect* are cea mai mică d_{min}, necesară pentru a corecta *t* biţi eronaţi şi garantează viteza informaţională maximă de corectare (cuvinte mai puţine şi mai scurte). Deci, pentru codul Hamming corector de o eroare $d_{min}=3$.

Dacă vom considera că la transmiterea unui cuvânt de *n* simboluri poate apărea o singură eroare, atunci numărul variantelor posibile va fi egal cu *n*+1, în care o combinaţie va reprezenta cuvântul de cod transmis fără erori, iar celelalte *n* combinaţii - toate variantele posibile de erori singulare. Deci, pentru a corecta presupusa eroare, utilizând *k* simboluri de control, este necesar a descrie *n+1* evenimente distincte:

$$\boldsymbol{n}+1=\boldsymbol{n}_0+\boldsymbol{k}+1\le 2^{\boldsymbol{k}} \qquad (4.1)$$

Din relaţia (4.1) rezultă:

1) Lungimea cuvântului de cod: $\boldsymbol{n}\le 2^{\boldsymbol{k}}-1$
2) Numărul simbolurilor informaţionale: $\boldsymbol{n}_0\le 2^{\boldsymbol{k}}-\boldsymbol{k}-1$

Redundanţa codului Hamming se calculează conform formulei:

$$\boldsymbol{R}=\frac{(\boldsymbol{n}-\boldsymbol{n}_0)}{\boldsymbol{n}} \qquad (4.2)$$

Utilizând relaţia (4.1), vom calcula numărul simbolurilor de control *k*, fiind cunoscut numărul simbolurilor informaţionale n_0 (tabelul 4.3).

Tabelul 4.3. Simbolurile informaţionale şi de control

n_0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
k	2	3	3	3	4	4	4	4	4	4	4	5	5	5

În tabelul 4.4 sunt prezentaţi parametrii pentru mai multe coduri Hamming cu lungimi diferite ale cuvintelor de cod, calculaţi conform relaţiei (4.1). Se observă că redundanţa descreşte rapid cu creşterea numărului de simboluri informaţionale, dar şi protecţia la

erori devine din ce în ce mai slabă (codurile pot corecta un singur simbol din 7, din 15, din 31, ş.a.m.d.).

Tabelul 4.4. Parametrii codurilor Hamming

Nr.	k	n	n_0	R	Codul Hamming
1.	2	3	1	0.6666	(3,1)
2.	3	7	4	0.4285	(7,4)
3.	4	15	11	0.2666	(15,11)
4.	5	31	26	0.1613	(31,26)
5.	6	63	57	0.0952	(63,57)
6.	7	127	120	0.0551	(127, 120)

Detectarea erorilor în codurile Hamming se bazează pe acelaşi principiu ca şi în codurile cu paritate. Deosebirea constă în următoarele:

- poziţiile de control sunt date de puteri ale lui 2;
- fiecare simbol de control răspunde pentru paritatea anumitor simboluri informaţionale.

Simbolurile de control se plasează pe poziţiile 2^0, 2^1, 2^2, ..., 2^{k-1} (1, 2, 4, 8, 16, ...) deoarece la reprezentarea binară a acestor numere este prezentă doar o singură unitate şi astfel se simplifică procedura de calcul a acestor simboluri. Fiecare simbol de control reprezintă paritatea unui set de simboluri informaţionale din cuvântul de cod. Poziţia simbolului de control determină secvenţa biţilor, care *verifică* şi *sare* alternativ. Poziţia 1 verifică un bit şi sare 1 bit (1, 3, 5, 7, 9 etc.). Poziţia 2 verifică 2 biţi şi sare 2 biţi (2, 3, 6, 7, 10, 11 etc.). Poziţia 4 verifică 4 biţi şi sare 4 biţi (4, 5, 6, 7, 12, 13, 14, 15 etc.). Poziţia 8 verifică 8 biţi şi sare 8 biţi (8-15, 24-31 etc.).

După transmiterea cuvintelor de cod, se calculează încă o dată paritatea seturilor de biţi, formându-se *sindromul erorii*. Dacă cuvintele au fost transmise fără erori, sindromul este nul. În cazul apariţiei unei erori, sindromul va indica poziţia ei în cuvântul transmis, iar simbolul eronat va fi corectat prin inversare. În final, din cuvântul de cod vor fi extrase doar simbolurile informaţionale.

Vom examina codul Hamming (7,4). Un cuvânt de cod are $n=7$ simboluri, dintre care $n_0=4$ simboluri informaţionale şi $k=n-n_0=3$ simboluri de control.

Vom nota prin d_i simbolurile informaţionale şi prin c_i simbolurile de control. Indicele i va corespunde poziţiei ocupate de simbolul respectiv.

Se va genera un cuvânt de cod $\boldsymbol{V}$, format din 7 biţi, prin plasarea simbolurilor informaţionale şi de control în felul următor:

$$\boldsymbol{V} = \{\boldsymbol{d}_7, \boldsymbol{d}_6, \boldsymbol{d}_5, \boldsymbol{c}_4, \boldsymbol{d}_3, \boldsymbol{c}_2, \boldsymbol{c}_1\}$$

Ţinând cont de cele expuse anterior referitor la calculul simbolurilor de control, vom obţine următoarele relaţii:

$$\begin{aligned} \boldsymbol{c}_1 &= \boldsymbol{d}_3 \oplus \boldsymbol{d}_5 \oplus \boldsymbol{d}_7 \\ \boldsymbol{c}_2 &= \boldsymbol{d}_3 \oplus \boldsymbol{d}_6 \oplus \boldsymbol{d}_7 \\ \boldsymbol{c}_4 &= \boldsymbol{d}_5 \oplus \boldsymbol{d}_6 \oplus \boldsymbol{d}_7 \end{aligned} \qquad (4.3)$$

La recepţionarea cuvântului de cod $\boldsymbol{V}' = \{\boldsymbol{d}_7', \boldsymbol{d}_6', \boldsymbol{d}_5', \boldsymbol{c}_4', \boldsymbol{d}_3', \boldsymbol{c}_2', \boldsymbol{c}_1'\}$, se va calcula sindromul erorii conform următoarelor relaţii:

$$\begin{aligned} \boldsymbol{s}_1 &= \boldsymbol{c}_1' \oplus \boldsymbol{d}_3' \oplus \boldsymbol{d}_5' \oplus \boldsymbol{d}_7' \\ \boldsymbol{s}_2 &= \boldsymbol{c}_2' \oplus \boldsymbol{d}_3' \oplus \boldsymbol{d}_6' \oplus \boldsymbol{d}_7' \\ \boldsymbol{s}_4 &= \boldsymbol{c}_4' \oplus \boldsymbol{d}_5' \oplus \boldsymbol{d}_6' \oplus \boldsymbol{d}_7' \end{aligned} \qquad (4.4)$$

Sindromul erorii $(\boldsymbol{s}_4, \boldsymbol{s}_2, \boldsymbol{s}_1)$ va fi nul, dacă cuvântul a fost transmis fără erori. În cazul apariţiei unei erori, sindromul va indica poziţia simbolului eronat.

Să analizăm un exemplu concret. Fie că este necesar a forma un cuvânt de cod pentru simbolurile informaţionale {1,0,1,1}. Pentru aceasta vom calcula simbolurile de control:

$$\begin{aligned} \boldsymbol{c}_1 &= \boldsymbol{d}_3 \oplus \boldsymbol{d}_5 \oplus \boldsymbol{d}_7 = 1 \oplus 1 \oplus 1 = 1 \\ \boldsymbol{c}_2 &= \boldsymbol{d}_3 \oplus \boldsymbol{d}_6 \oplus \boldsymbol{d}_7 = 1 \oplus 0 \oplus 1 = 0 \\ \boldsymbol{c}_4 &= \boldsymbol{d}_5 \oplus \boldsymbol{d}_6 \oplus \boldsymbol{d}_7 = 1 \oplus 0 \oplus 1 = 0 \end{aligned}$$

Se va obţine următorul cuvânt de cod:

Poz.	d_7	d_6	d_5	c_4	d_3	c_2	c_1
$\boldsymbol{V}$	1	0	1	0	1	0	1

Să presupunem, că la transmitere a fost recepţionat cuvântul de cod $\boldsymbol{V}'$={1, 1, 1, 0, 1, 0, 1}:

Poz.	d_7'	d_6'	d_5'	c_4'	d_3'	c_2'	c_1'
$\boldsymbol{V}'$	1	1	1	0	1	0	1

Pentru a determina dacă cuvântul de cod a fost transmis corect, se va calcula sindromul erorii conform relaţiilor (4.4).

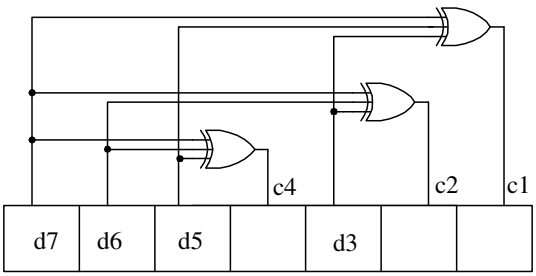
$$\boldsymbol{s}_1 = \boldsymbol{c}_1' \oplus \boldsymbol{d}_3' \oplus \boldsymbol{d}_5' \oplus \boldsymbol{d}_7' = 1 \oplus 1 \oplus 1 \oplus 1 = 0$$
$$\boldsymbol{s}_2 = \boldsymbol{c}_2' \oplus \boldsymbol{d}_3' \oplus \boldsymbol{d}_6' \oplus \boldsymbol{d}_7' = 0 \oplus 1 \oplus 1 \oplus 1 = 1$$
$$\boldsymbol{s}_4 = \boldsymbol{c}_4' \oplus \boldsymbol{d}_5' \oplus \boldsymbol{d}_6' \oplus \boldsymbol{d}_7' = 0 \oplus 1 \oplus 1 \oplus 1 = 1$$

Sindromul $(\boldsymbol{s}_4, \boldsymbol{s}_2, \boldsymbol{s}_1)$=(1,1,0) indică că eroarea a avut loc în poziţia 6 (bitul d_6). Acest bit poate fi corectat prin inversare.

4.5.1. Codorul şi decodorul Hamming detector şi corector de o eroare

Circuitul de la emisie, şi anume, codorul, care are misiunea de a calcula simbolurile de control, cunoscute fiind simbolurile informaţionale, va fi format dintr-un registru, care conţine *n* circuite basculante bistabile, în care se stochează simbolurile informaţionale în poziţiile corespunzătoare din cuvântul de cod şi o serie de sumatoare modulo 2, care, conform relaţiilor (4.3), calculează simbolurile de control. Simbolurile de control, astfel calculate, se stochează apoi în poziţiile, în care acestea intervin în cuvântul de cod.

Codorul pentru codul Hamming (7,4), realizat în baza relaţiilor (4.3) va avea structura din figura 4.3.



4.3. Codorul Hamming

Circuitul de la recepţie, şi anume, decodorul Hamming, va fi format dintr-un registru cu *n* circuite basculante bistabile, în care va fi

memorat cuvântul recepţionat, *k* sumatoare modulo 2, care calculează componentele sindromului şi un decodificator binar cu *k* intrări şi *n+1* ieşiri. La intrările decodificatorului vor fi aplicaţi cei *k* biţi ai sindromului. În dependenţă de semnalul activ de la ieşirea decodificatorului se va stabili dacă cuvântul de cod a fost transmis fără erori sau, în cazul unei erori, se va determina poziţia acesteia. Corectarea prin inversare a simbolului eronat se va efectua prin intermediul a *n* sumatoare modulo 2.

Structura decodorului pentru codul Hamming (7,4), în care sindromul se calculează conform relaţiilor (4.4), este prezentată în figura 4.4.

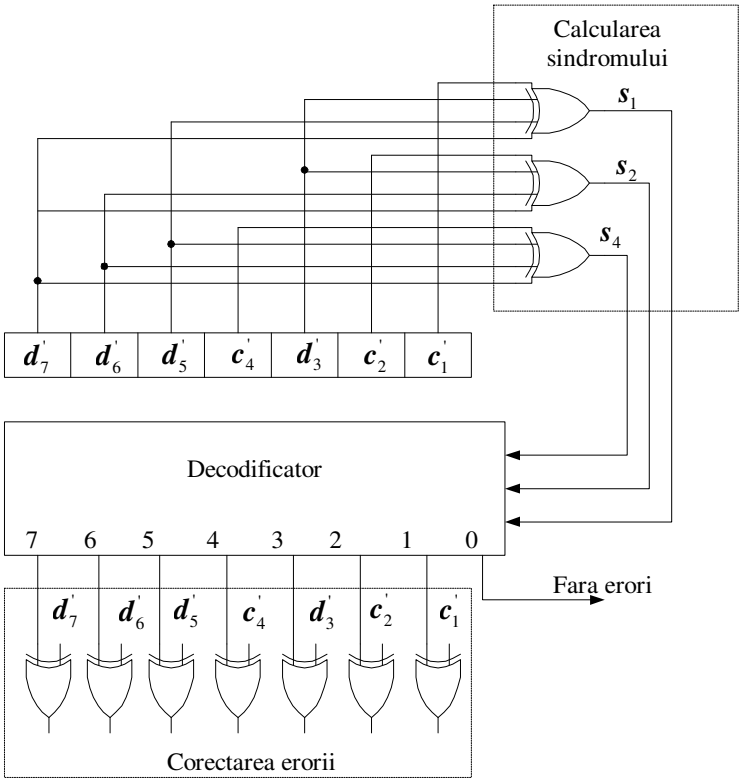


Figura 4.4. Decodorul Hamming

4.5.2. Codul Hamming detector de două erori şi corector de o eroare

În scopul corecţiei unei erori şi a detectării erorilor duble trebuie adăugat la cele n_0 simboluri informaţionale, pe lângă simbolurile de control necesare corecţiei unei erori, un simbol suplimentar, numit de *control al parităţii*, prin care se decide dacă în cuvântul recepţionat sunt două erori sau o eroare.

Structura cuvântului de cod în acest caz este de forma:

$$\boldsymbol{V} = \{\boldsymbol{d}_7, \boldsymbol{d}_6, \boldsymbol{d}_5, \boldsymbol{c}_4, \boldsymbol{d}_3, \boldsymbol{c}_2, \boldsymbol{c}_1, \boldsymbol{c}_0\}$$

Simbolul suplimentar de control al parităţii c_0 determină paritatea tuturor simbolurilor din cuvântul de cod şi este calculat conform relaţiei:

$$\boldsymbol{c}_0 = \boldsymbol{d}_7 \oplus \boldsymbol{d}_6 \oplus \boldsymbol{d}_5 \oplus \boldsymbol{c}_4 \oplus \boldsymbol{d}_3 \oplus \boldsymbol{c}_2 \oplus \boldsymbol{c}_1 \qquad (4.5)$$

Fie $\boldsymbol{V}' = \{\boldsymbol{d}_7', \boldsymbol{d}_6', \boldsymbol{d}_5', \boldsymbol{c}_4', \boldsymbol{d}_3', \boldsymbol{c}_2', \boldsymbol{c}_1', \boldsymbol{c}_0'\}$ cuvântul recepţionat. Sindromul erorii va include un simbol suplimentar $\boldsymbol{s}_0$ şi va avea următoarea formă:

$$\boldsymbol{s} = \{\boldsymbol{s}_4, \boldsymbol{s}_2, \boldsymbol{s}_1, \boldsymbol{s}_0\}$$

Biţii $\boldsymbol{s}_4, \boldsymbol{s}_2$ şi $\boldsymbol{s}_1$ vor fi calculaţi conform relaţiilor (4.4), iar bitul $\boldsymbol{s}_0$ conform următoarei relaţii:

$$\boldsymbol{s}_0 = \boldsymbol{d}_7' \oplus \boldsymbol{d}_6' \oplus \boldsymbol{d}_5' \oplus \boldsymbol{c}_4' \oplus \boldsymbol{d}_3' \oplus \boldsymbol{c}_2' \oplus \boldsymbol{c}_1' \oplus \boldsymbol{c}_0' \qquad (4.6)$$

Comparând relaţia (4.5) cu (4.6), rezultă că, dacă în cuvântul recepţionat a apărut un număr impar de erori, atunci $\boldsymbol{s}_0 = 1$, iar dacă a apărut un număr par de erori, atunci $\boldsymbol{s}_0 = 0$.

Dacă pe canalul de transmisiuni pot să apară cel mult două erori, rezultă următoarele patru situaţii:

a) Toate componentele sindromului sunt nule:

$$\boldsymbol{s}_4 = \boldsymbol{s}_2 = \boldsymbol{s}_1 = \boldsymbol{s}_0 = 0$$

În acest caz se decide că ceea ce s-a recepţionat este corect.

b) Componenta $\boldsymbol{s}_0 = 1$, iar celelalte componente ale sindromului nu sunt toate nule. În acest caz, în cuvântul recepţionat a apărut o eroare a cărei poziţie se determină conform echivalentului zecimal al componentelor sindromului $\boldsymbol{s}_4, \boldsymbol{s}_2$ şi $\boldsymbol{s}_1$:

$$(s_4 s_2 s_1)_2 = (I)_{10}$$

c) Componenta $s_0 = 0$, iar celelalte componente ale sindromului nu sunt toate nule. În acest caz se decide că în cuvântul recepţionat sunt două erori necorectabile, caz în care se va cere retransmisia cuvântului respectiv.

d) Componenta $s_0 = 1$, iar celelalte componente ale sindromului sunt toate nule:

$$s_4 = s_2 = s_1 = 0$$

În acest caz rezultă că eroarea a avut lor în poziţia c_0.

4.6. Lucrarea de laborator Nr. 4

Tema: Codor şi decodor Hamming detector şi corector de o eroare

Scopul lucrării: Studierea codurilor Hamming şi obţinerea deprinderilor practice de elaborare a codorului şi decodorului Hamming pentru cuvinte de cod de diferite lungimi.

Tema pentru acasă

1. Să se calculeze numărul simbolurilor de control pentru codul Hamming detector şi corector de o eroare, fiind cunoscut numărul de simboluri informaţionale (conform variantei din tabelul 4.5) şi să se formeze cuvântul de cod V prin plasarea corespunzătoare a simbolurilor informaţionale şi de control.
2. Să se deducă relaţiile pentru calcularea simbolurilor de control şi să se efectueze sinteza codorului Hamming.
3. Să se deducă relaţiile pentru calcularea sindromului erorii şi să se efectueze sinteza decodorului Hamming.

Desfăşurarea lucrării

1. Se asamblează codorul Hamming conform relaţiilor pentru calcularea simbolurilor de control. Pentru introducerea şi vizualizarea simbolurilor informaţionale se vor utiliza

elementele **Binary Switch** şi **Binary Probe**. Cuvântul de cod generat se va memora într-un registru de *n* biţi.

2. Se asamblează decodorul Hamming conform relaţiilor pentru calcularea sindromului erorii. Pentru descifrarea erorii se va utiliza un decodificator binar de *n+1* biţi. Cuvântul de cod corectat se va memora într-un registru de *n* biţi.

3. Pentru verificarea circuitelor asamblate se vor introduce erori în mod aleatoriu în cuvintele de cod formate şi se vor analiza cuvintele de cod corectate.

Tabelul 4.5. Variante pentru îndeplinirea lucrării de laborator

Nr.	Numărul simb. inf. (n_0)	Simbolurile informaţionale
1	5	10111
2	6	110101
3	7	1011111
4	8	11101010
5	9	110101010
6	10	1000111100
7	11	10111011101
8	12	101110110110
9	13	1110100011001
10	14	10001001100111

Întrebări

1. Numiţi tipurile de redundanţă utilizate la proiectarea sistemelor fiabile.
2. Ce reprezintă redundanţa informaţională?
3. Care sunt strategiile de bază pentru protecţia datelor împotriva erorilor?
4. Explicaţi care este deosebirea între codurile detectoare şi corectoare de erori şi codurile simple.
5. Ce categorii de coduri detectoare şi corectoare de erori cunoaşteţi?
6. Definiţi distanţa Hamming şi distanţa minimă de cod.
7. Care este principiul de selectare a valorii bitului suplimentar pentru codurile cu paritate?
8. Numiţi avantajele şi dezavantajele codurilor cu paritate.
9. Când se alege paritatea cu soţ şi paritatea fără soţ?

10. Explicaţi principiul de codare în cazul parităţii încrucişate.
11. Ştiind că într-o transmisie de date se utilizează detectarea erorilor prin paritate încrucişată, care este blocul de informaţie ataşat emisiei cifrelor zecimale de la 4 la 9, codificate primar cu ajutorul codului EXCES 3. O secvenţă este reprezentată prin codificarea unei singure cifre zecimale.
12. Cunoscând că a fost transmis un bloc de date codificat prin paritate încrucişată s-a recepţionat:

    0 1 1 0 1
    1 0 0 0 1
    1 0 0 1 1
    1 0 1 0 0
    1 0 1 1 1
    1 1 0 0 0
    1 0 1 1 1,

    să se verifice corectitudinea recepţiei.
13. Cum este determinat numărul biţilor de control pentru codurile Hamming?
14. În ce caz un cod este numit perfect?
15. Care sunt considerentele de plasare a biţilor de control în codurile Hamming?
16. Ce reprezintă sindromul erorii?
17. Reprezentaţi un cuvânt de cod pentru codul Hamming (15,11), notând prin d_i simbolurile informaţionale şi prin c_i simbolurile de control. Obţineţi relaţiile de calcul a simbolurilor de control şi a sindromului erorii.
18. Să se determine secvenţele codului Hamming corespunzătoare cifrelor zecimale 7,8 şi 9 ştiind că în codificarea primară s-a folosit codul 8421.
19. Să se determine secvenţele codului Hamming corespunzătoare cifrelor zecimale 5,6 şi 7 ştiind că în codificarea primară s-a folosit codul 2421.
20. Fie mesajul recepţionat 1111000.

    a) Ştiind că reprezintă o secvenţă a codului Hamming, să se verifice corectitudinea lui. În caz de eroare, presupunând că o singură poziţie este eronată, să se asigure corecţia.

    b) Cărei cifre zecimale îi corespunde mesajul corectat, dacă într-o primă codificare a fost folosit codul 8421.

21. Cele 6 simboluri generate de o sursă sunt transmise pe un canal binar cu perturbaţii folosind un cod Hamming grup corector de o eroare.
    a) Să se determine numărul simbolurilor informaţionale, numărul simbolurilor de control şi lungimea cuvintelor de cod.
    b) Să se explice, pentru un exemplu concret, ce se întâmplă dacă în cuvântul recepţionat apar două erori pe poziţiile 1 şi 2.
22. Din ce părţi componente este format codorul Hamming şi care sunt funcţiile pe care acestea le îndeplinesc?
23. Din ce părţi componente este format decodorul Hamming şi care sunt funcţiile pe care acestea le îndeplinesc?
24. Cu ce este egală d_{min} pentru codul Hamming detector de două erori şi corector de o eroare?
25. Ce determină simbolul suplimentar de control al parităţii pentru codul Hamming detector de două erori şi corector de o eroare?
26. Cum este determinată situaţia de apariţie a două erori pentru codul Hamming detector de două erori şi corector de o eroare?
27. Cu ce vor fi egale componentele sindromului pentru codul Hamming detector de două erori şi corector de o eroare, dacă în cuvântul de cod a avut loc o singură eroare?
28. Să se determine secvenţele codului Hamming detector de două erori şi corector de o eroare pentru cifrele zecimale 5,6 şi 7 ştiind că în codificarea primară s-a folosit codul EXCES 3.

BIBLIOGRAFIE

1. A. Gremalschi. Diagnosticarea tehnică a echipamentelor microprocesor, Chişinău, Universitas: 1992.
2. Gheorghe M.Panaitescu, Transmiterea şi codarea informaţiei, Note de curs, Universitatea "Petrol-Gaze" Ploiesti, Catedra electrotehnică si electronică, 2009
3. Gheorghe M.Panaitescu, Sisteme tolerante la defecte, Note de curs, Universitatea "Petrol-Gaze" Ploiesti, Catedra automatică si calculatoare, 2007.
4. Alexandru Valachi ş. a. Analiza, sinteza şi testarea dispozitivelor numerice, Editura Nord-Est, Iaşi, 1993.
5. Nicolae Mărăşescu, Fiabilitate şi diagnoză, Editura Fundaţiei Universitare "Dunărea de jos", Galaţi, 2004.
6. Cătuneanu M. V., Bacivarof A. Structuri electronice de înaltă fiabilitate. Toleranţa la defectări, Editura militară, Bucureşti, 1989.
7. Toacşe G., Nicula D. Electronica digitală, Bucureşti, Teora, 1996.
8. Ярмолик В.Н. Контроль и диагностика цифровых узлов ЭВМ. – Мн.: Наука и техника, 1988.
9. Р. Беннеттс. Проектирование тестопригодных логических схем. – М., Радио и связь: 1990.
10. Г.Б. Уильямс. Отладка микропроцессорных систем. – М., Энергоатомиздат: 1989.
11. Пархоменко П. П. и др. Основы технической диагностики, в 2^x томах, М.: Энергия, 1977, 1981.
12. Parag K. Lala. An Introduction to Logic Circuit Testing, A Publication in the Morgan & Claypool Publishers series, ebook, www.morganclaypool.com.

Anexă

MINISTERUL EDUCAŢIEI AL REPUBLICII MOLDOVA

Universitatea Tehnică a Moldovei

Catedra Calculatoare

Testarea Sistemelor de Calcul

REFERAT
LA LUCRAREA DE LABORATOR Nr. 1

Tema: Generarea testelor prin metoda activării unei căi

Elaborat — **studentul gr. C-081**
Ion Moraru

Verificat — **lect. sup., dr.**
Viorica Sudacevschi

Chişinău
2011

CUPRINS