3.1 Introducere

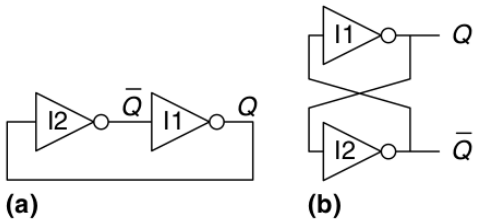
În capitolul precedent, noi am cercetat procesul de analiză şi proiectare a cirutelor logice combinaţionale. Ieşirea în circuit logic combinaţional depinde doar de valoarea la intrare la momentul curent. Noi putem crea un circuit optimizat, pentru a satiface specificaţia, în formă de tabel de adevăr sau o ecuaţie booleană.

În acest capitol, vom analiza şi proiecta circuite logice secvenţiale. Valoarea la ieşirea circuitului logic secvenţial depinde cum de valorile curente de intrare, atînt şi de precedente. Prin urmare, circuitele logice secvenţiale au memorie. Logica secvenţială poate în mod explicit memora anumite valori aplicate anterior la intrare, sau poate «comprima» valorile de intrare anterioare într-o cantitate mai mică de informaţie, numită *starea sistemului*. Starea circuitului logic secvenţial de setare a bitului se numeşte *starea variabilă*. Aceasta conţine toată informaţia despre trecut, necesară pentru a explica comportamentul circuitului în viitor.

Capitolul se începe cu studierea latch şi flip-flop. Ele sunt circuite secvenţiale simple, ce memorează un bit de informaţie. În linii generale, circuitele secvenţiale sunt complicate în analiză. Pentru a simplifica proiectarea, noi ne vom limita cu scheme secvenţiale sincorone. Ele sunt compuse din logica combinatorică şi set de flip-flop, ce va conţine informaţia despre starea circuitului. În acest capitol se descrie maşina fintă, cu ajutorul cărora este posibil uşor de proiectat circitele secvenţiale. În sfârşit, noi vom analiza rapiditatea circuitelor secvenţale şi vom discuta calculul în paralel ca o modalitate de ridicare a performanţelor.

3.2 Bistabile

Blocul principal pentru construcţia memoriei este elementul *bistabil*, un element cu două stari stabile. Îm figura 3.1(a) observăm un simplu bistabil compusă din o pereche de inversoare închise în buclă. În figura 3.1(b) vedem acelaşi circuit redesenat astfel, încât se va evidenţia simetria. Inversoarele sunt cuplate încrucişat, ceea ce înseamnă că intrarea I1 este ieşirea I2 şi vice-versa.



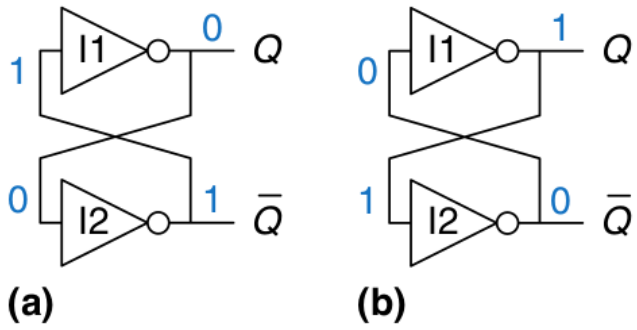
Circuitul nu are intrări, dar are două ieşiri, Q şi $\underline{Q}$. Analiza acestui circuit diferă de analiza circuitului combinaţionla deoarece este circuitul este ciclic: Q depinde de $\underline{Q}$, şi $\underline{Q}$ depinde de Q.

Considerăm două cazuri, Q este 0 sau Q este 1. Lucrînd prin intermediul consecinţelor ale fiecărui caz, avem:

Cazul I: Q=0

Cum se arată în figura 3.2(a), I2 primeşte la intrare FALSE, deci Q la ieşire are TRUE. I1 primeşte la intrare TRUE şi asigură un rezultat FALSE la Q. Acest lucru este în concordanţă cu originalul ipoteză că Q= 0, astfel cazul este declarat a fi stabil.

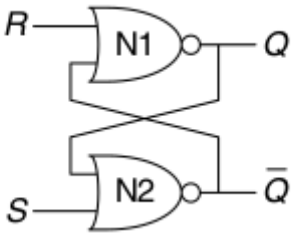
Cazul II: Q=1
Cum se arată în figura 3.2 (b), I2 primeşte la intrare TRUE şi produce la ieşirea Q - FALSE . I1 primeste la intrare FALSE şi produce un TRUE la ieşirea Q. Acest lucru este din nou stabil.
Circuitul se numeşte bistabil, deoarece inversoarele cuplate încrucişat au două stări stabile, Q=0 şi Q=1. Un punct subtil este faptul că circuitul are a trei-a stare, cu ambele ieşiri de aproximativ la jumătatea distanţei între 0 şi 1. Această stare se numeşte *metastabilă* şi va fi discutat în paragraful 3.5.4.
Un element cu N stări stabile păstrează $\log_2 N$ biţi de inormaţie. Astfel, bistabilul păstrază un bit. Starea inversoarelor cuplate încrucişat este păstrată într-o singură variabilă binară de stare, Q. Valoarea Q ne spune totul despre trecut, care este necesar pentru a explica comportamentul circuitului în viitor.



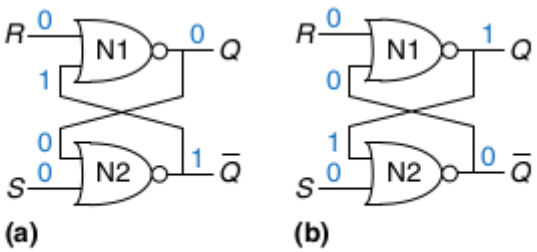
Mai exact, în cazul în care Q=0, acesta va rămâne 0 pentru totdeauna, şi în cazul în care Q=1, acesta va rămâne pentru totdeauna 1. Circuitul are o altă ieşire, $\underline{Q}$, dar $\underline{Q}$ nu conţine nici o informaţie suplimentară, deoarece dacă Q este cunoscut, $\underline{Q}$ de asemenea este cunoscut. Pe de altă parte, $\underline{Q}$ este de asemenea o acceptabilă alegere pentru variabila de stare.
Atunci când puterea este aplicată mai întâi la un circuit secvenţial, starea iniţială este necunoscut şi de obicei imprevizibilă. Acesta poate fi de fiecare dată diferită când circuitul este pornit.
Cu toate că inversoarele cuplate încrucişat pot păstra un bit de informaţie, ele nu sunt practice, deoarece utilizatorul nu are intrări pentru a controla starea. Cu toate acestea, alte elemente bistabile, cum ar fi latch şi flip-flop, furnizează intrări pentru a controla valoarea variabilei de stare. Aceste circuite sunt discutate în restul paragrafelor.

3.2.1 SR-latch

Una dintre cele mai simple circuite secvenţiale este SR-latch, compuse din două elemente cuplate încrucişat NOR, ca în figura 3.3. Un latch are două intrări, S şi R, şi două ieşiri, Q şi $\underline{Q}$. Un SR-latch este similar inversoarele cuplate încrucişat, dar starea unui latch se controlează cu ieşrile S şi R, care setează şi resetează ieşirea Q.



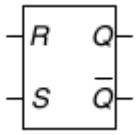
Pentru a cunoşte şi a ne familiariza mai bine cu lucrul circuitului, se construieşte tabelul de adevăr, iată cu ce vom începe.
O modalitate mai buna de a înţelege un circuit nefamiliar este de a lucra cu tabelul de adevăr, astfel cu asta şi von începe. Să ne amintim că o poarta NOR crează la ieşire FALSE atunci când la intrare este TRUE. Se iau în considerare cele patru combinaţii posibile de R şi S.
Cazul I: R=1, S=0
N1 are cel puţin un TRUE de la intrarea, R, deci se crează la ieşirea Q- FALSE. N2 arecît Q atât şi S- FALSE, deci se crează un TRUE la ieşirea Q.
Cazul II: R=0, S=1
N1 primeştela intrări 0 şi Q. Din motivul că noi nu ştim încă Q, noi nu putem determina ieşirea Q. N2, primeşte cel puţin o intrare TRUE, S crează FALSE la ieşirea Q. Acum ne putem întoarce la N1, ştiind că ambele intrări sunt FALSE, rezultă că ieşirea Q are TRUE.
Cazul III: R=1, S=1
N1 şi N2, au minimul câte un TRUE (R sau S), astfel fiecarecrează la ieşire FALSE. De aceea Q şi Q sunt ambele FALSE.
Cazul IV: R=0m S=0
N1 primeşte intrări de 0 şi Q. Din motivul că noi nu ştim încă Q, noi nu putem determina de ieşirea. N2 primeşte intrările de 0 şi Q. Din motivul că noi nu ştim încă Q, nu putem determina ieşirea. Acum suntem blocaţi. Aceasta ne aminteşte de inversoate cuplate încrucişat. Dar noi ştiu că Q trebuie să fie 0 sau 1. Deci, noi putem rezolva problema studiind fiecare caz parţial.



Cazul IVa: Q=0
Deoarece S şi Q sunt FALSE, N2 crează la ieşirea Q – TRUE, cum se observă în fgura 3.4(a). Acum N1 are TRUE la ieşirea Q, deci la ieşirea Q va fi FALSE, cum şi am presupus.
Cazul IVb: Q=1
Deoarece Q este TRUE,N2 creză FALSE la ieşirea Q, cum e în figura 3.4(b). Acum N1 are două intrări FALSE, R şi Q, deci ieşirea Q este TRUE, cum şi am presupus.

Combinând toate ecestea, să presupunem că Q are o valoare cunoscută anterior,pe care o vom numi Qprev, înainte de Cazul IV. Qprev poate fi 0 sau 1, şi reprezintă starea sistemului. Când R şi S sunt 0, Q va memora această valoare veche, Qprev, şi va fi complement Q. Acest circuit are memorie.

Case	S	R	Q	$\bar{Q}$
IV	0	0	Q_{prev}	$\bar{Q}_{prev}$
I	0	1	0	1
II	1	0	1	0
III	1	1	0	0



Tabelul de adevăr în figura 3.5 smează aceste patru cazuri. Intrările S şi R răspund de set şi reset. Serarea bitului înseamnă de a-i dea valoarea TRUE. Resetarea bitului îneamnă de a-i dea valoarea FALSE. Ieşirile Q şi Q sut cumplemetare. Când R este activat, Q este resetat la 0, iar Q ia valoarea opusă. Când S este activ, Q este setat la 1, iar Q ia valaorea opusă. Atunci când nici o intrare nu este activată, Q îşi aminteşte valoarea sa veche, Qprev. Asertarea S şi R simultan nu prea are sens, deoarece aceasta înseamnă că dispozitivul de blocare trebuie să fie setat şi resetat în acelaşi timp, ceea ce este imposibil. Bietul Circuitul confuz răspunde prin ambele ieşiri 0.
SR latch este reprezentat prin simbolul din figura 3.6. Utilizarea simbolul este o aplicaţie de captare şi modularitate. Există diverse moduri de a construi un SR latch, cum ar fi folosirea diferitor porţi logice sau tranzistori.
Cu toate acestea, orice element al circuitului, specificat prin tabelul de adevăr din figura 3.5, iar simbolul din figura 3.6 se numeşte SR latch.
La fel ca inversoarele cuplate încrucişat, SR latch este un element bistabil cu un bit de stare stocat în Q. Cu toate acestea, starea poate fi controlată prin intrările S şi R. Atunci când R este activat, starea este resetată la 0. Când S este activat, starea este setată la 1. Atunci când nici o intrare nu este activată, starea îşi păstrează valoarea sa veche.
Observaţi că întreaga istorie a intrărilor poate fi reprezentat de o singură vriabilă de stare Q. Nu contează ce a avut loc în trecut, setare sau resetare, tot ceea ce este necesar pentru a prezice comportamentul viitor al SR latch este stabilirea valorii recente, setarea sau resetarea.

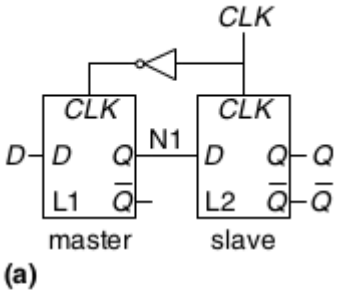
3.2.2 D latch

SR latch este incomod, deoarece acesta se comportă ciudat, atunci când ambele intrări S şi R sunt active simultan . Mai mult ca atât, intrările S şi R unesc întrebările *ce* şi *când*. Dând unitate logică la aceste ieşiri, determină nu doar *ce* se va întâmpla, dar şi *când*. Proiectarea circuitelor devine mai uşoară atunci când aceste întrbări, *ce* şi *când* sunt separate. D latch din figura 3.7 (a) rezolvă aceste probleme. Ea are două intrări. Intrarea pentru *date*, D, ceea ce controlează starea următoare care ar trebui să fie. Intrarea de ceas, CLK, controlează atunci când statul ar trebui schimbat.
Pentru analiza latch-ului, noi din nou scriem tabelul de adevăr, dat în Figura 3.7(b). Pentru comditate, începem cu analiza nodurilor D, S şi R. Dacă CLK=0, atunci S şi R sunt FALSE, indiferent de valoarea lui D. Dacă CLK=1, o poartă AND va produce TRUE, iar cealaltă FALSE, în fencţie de valoarea lui D. Având S şi R, Q şi Q vor fi determinate foloind Figura 3.5. Observăm că, atunci când CLK=0, Q salvează valoarea precedentă a lui Qprev.

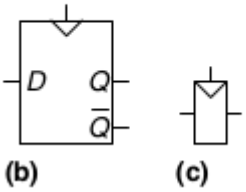
Când CLK=1, Q=D. În toate cazurile, Q este unt complement a lui Q. D latch evită cuzul de afirmare simultană a intrărilor S şi R (S-1 şi R=1).

În aşa mod, noi vedem că cesul controlează atunci când datele trec prin latch. Când CLK=1, latch-ul este ”transparent”. Datele de la D trec spre ieşirea Q, ca cum latch-ul ar fi un simplu buffer. Când CLK=0, latch-ul este ”netransparent”. Acesta nu permite datelor noi să circule spre ieşirea Q, şi Q îşi patrează valoarea precedentă. De aceea, D latch-ul uneori este numit *latch transparent* sau *latch sensibil la nivele*. Simbolul pentru D latch este reprezentat în Figura 3.7(c).
Starea D latch-uli se schimbă incontinuu, atât timp cât CLK=1. Mai târziu în acest capitol noi vom vedea, că mai comod este să schimbăm starea schemei numai în anumite momente de timp. D flip-flop descris în peregraful următor şi nu doar asta.

3.2.3 D Flip-Flop



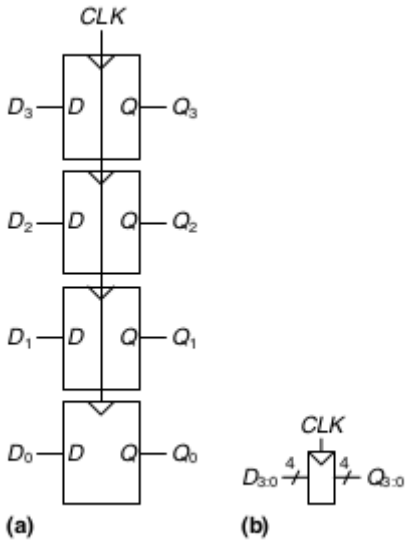
Un D flip-flop poate fi construit din două D latch consecutive , controlate de un clock complementar, aşa cum se arată în figura 3.8 (a). Primul latch, L1, este numit master. Al doilea latch, L2, se numeşte slave. Nodul
între ele este numit N1. Simbolul pentru D flip-flop este arătat în Figura 3.8 (b). Atunci când nu avem nevoie de ieşirea Q, simbolul este adesea comprimat/simplificat ca în figura 3.8 (c).



Atunci când CLK=0, dispozitivul de blocare master este deschis, iar sclavul este închis. Prin urmare, orice valoare de la D se propagă/trece până la N1. Atunci când CLK= 1, mesterul se blochiază şi slave-ul devine deschis. Valoarea lui N1 se propagă până la ieşirea Q, dar N1 se taie de D. Prin urmare, indiferent de valoarea ce a fost la D, imediat înainte de trecerea CLK de la 0 la 1,imediat ajunge la ieşirea Q dupa ce semnalul de clock devine 1. În tot celălat timp, Q îşi păstrază valoarea precedentă, deoarece latch-ul închis pemanent blochiază drumul ăntre D şi Q.
Cul alte cuvinte, un D Flip-Flop copie valoarea de pe D pe Q pe frontul din faţă a semnallui de clock şi memorizează acestă stare tot timpul celălalt. Citiţi acestă definiţie până când o veţi memora. Cea mai răspândită graşeală a proiectanţilor începători de scheme digitale – ei uită ce este ce face un Flip-Flop. Deseori frontul din faţă a semnalul de clock se numeşte pur şi simplu front. Intrarea D determina starea nouă, din viitor. Frontul determină momentul de timp când starea va fi schimbată.
D Flip-Flop cunoscut ca MS Flip-Flop, Master-Slave Flip-Flop şi ca Flip-Flop sincronizat de front(margini). Triunghiul de denumiri ne arată că intrarea este sincronizată de front. La multe Flip-Flop-uri ieşirea Q lipseşte şi, de obicei, se folosesc când nu este neboie de Q.

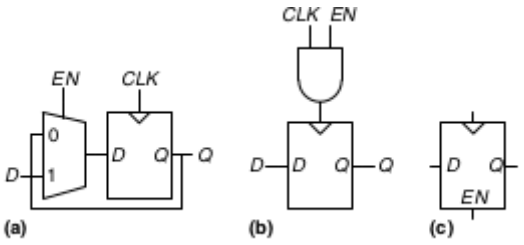
3.2.4 Registrul

Un registru de N- biţi, este un ansamble de N flip-flop cu un semnal de clock comun. Astfel, toţi biţii registrului se reînoiesc concomitent. Registrul este blocul de cheie în construcţia majorităţii schemelor secvenţiale. Figura
3.9 prezintă schema şi simbolul pentru un registru de patru biţi cu intrări D3: 0 şi ieşirile Q3: 0. D3: 0 şi Q3: 0 sunt ambele magistrale de 4 biţi.



3.2.5. Enable Flip flop

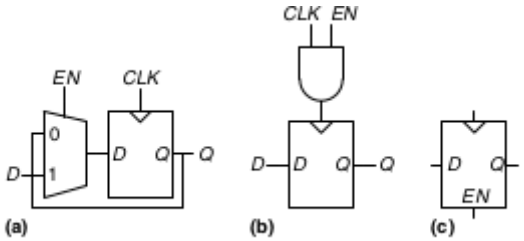
Un enable flip-flop mai are o altă intrare numită EN sau ENABLE(permis) pentru a determina dacă datele sunt încărcate pe frontul clock-ului. Atunci când EN este TRUE, enable flip-flop-ul se comportă ca un D flip-flop obişnuit.
Atunci când EN este FALSE, enable flip-flop-ul ignoră semnalul de clock şi îşi păstrează starea sa. Enable flip-flop sunt utile atunci când dorim să încărcăm o noua valoare într-un flip-flop doar o parte din timp, dar nu pe fiecare
front de clock.
Figura 3.10 prezintă două moduri de a construi un enable flip-flop dintr-un D flip-flop şi o poartă în plus. În figura 3.10 (a), un multiplexor de intrare alege dacă să treacă valoarea la intrarea D, în cazul în care EN are 1 logic, sau să recicleze starea vechi de la ieşirea Q, în cazul în care EN are 0 logic. În figura 3.10 (b), clock-ul este închis.



În cazul în care EN este TRUE, impulsul la semnalului de clock se transferă în mod normal. În cazul în care EN este FALS, intrareaCLK este, de asemenea, FALSE şi flip flop-ul îşi păstrează vechea valoare. Luaţi în seamă că EN nu trebuie să se schimbe în timp ce CLK=1,

ca nu cumva flip flop-ul să vadă o eroare de clock(comutator la un moment incorect). În general, adăugarea elementelor logice pe clock este o idee proastă. Controlul tactului introduce reţinere în semnalul de clock şi poate aduce la erori de timp, despre care se va vorbi în paragraful 3.5.3. Deci, o puteţi face doar în cazul în care sunteţi siguri de ceea ce faceţi.

Simbolul pentru un enable flip flop este prezentat în figura 3.10 (c).



3.2.6 Flip Flop cu funcţia de resetare.

În flip-flop cu functia de resetare se mai adaugă o ieşire, cu denumirea RESET. Când RESET este FALSE, flip-flop cu resetare se comporta ca un D flip-flop obişnuit. Când RESET este TRUE, astfel de flip-flop ignoră intrarea D şi resetează ieşirea la 0. Flip-flop-urile cu funcţia de resetare sunt utile, când dorim să mărim viteza de stabilire a stării (adică 0) la toate flip-flop-urile sistemei la prima conectare.

Astfel de flip-flop-uri sunt capabile să reseteze sincorn sau asincron. Flip-flop-uri cu resetare sincronă resetoază doar pe frontul semnalului de CLK. Flip-flop-uri cu resetare asincronă resetează îndată cu primirea 1-ului logic la intrarea RESET, indiferent de semnalul de clock.

În figra 3.11(a) este reprezentat cum se crează un flip-flop cu funcţia de resetare sincronă dintr-un D flip-flop obişnuit şi elementul AND. Când RESET devine FALSE, elemntul AND transmite 0 la intrarea flip-flop-ului. Când RESET devine TRUE, elentul And transmite semnelul D la intrarea flip-flop-ului. În acest exemplu RESET este un semnal slab activ, ceea ce înseamnă că resetul se efectuează când la intrare avem 0, nu 1. Adăugând un inversor, s-ar primi o schemă cu un semnal activ ridicat. Figurile 3.11(b) şi 3.11(c) arată simbolurile pentru un flip-flop resetabil cu RESET activ ridicat.

Flip-flop-ul cu funcţia de reset asincron necesită schimbări în sistema sa internă şi sunt lăsate pentru priectare în Exerciţiul 3.10., dar sunt cele mai dese ori disponibile pentru proiectanţi ca componente standarte.

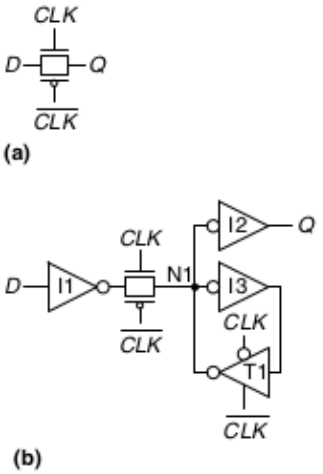
După cum aţi putut deja observa, uneori se folosesc şi flip-flop-uri cu funcţia de setare. Când semnalul SET este activ, în astefel de flip-flop se încarcă 1 logic şi execută sincron şi asincron. La flip-flop-uri resetabile şi setabile la poate fi şi intarea ENABLE şi pot fi grupate registra de N-biţi.

3.2.7. Proiectarea flip-flop-ului şi latch-ului la nivelul de tranzistori.

În exemplul 3.1 se arată, că dacă flip-flop-ul este construit din elemente logice, atunci în ele se utilizează un număr mare de tranzistoare. Dar functia fundamentală a unui latch este de a fi deschis sau închis, ce îl asemănă cu un switch. În paragraful 1.7.7 este descris, că utilizarea poartei de transmisie este o metodă efectivă de a crea switch CMOS. Rezultă, că ne putem folosi de avantajele switch-ului pentru a micşora numărul tranzistorilor.

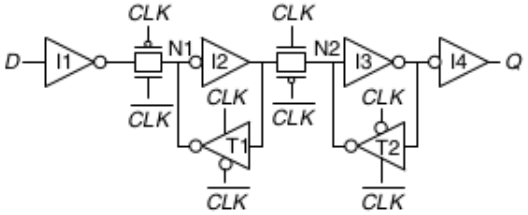
Un D latch compact poate fi construit dintr-o singură poartă de transmisie, aşa cum este arătat în Figura 3.12(a). Cînd CLK=1 şi $\underline{CLK}$=0, poarta de transmisie este activă, D trece la Q şi latch-ul este deschis. Cînd CLK=0 şi $\underline{CLK}$=1, poarte de transmisie nu mai este activă şi Q devine izolat de D iat latch-ul este închis. Acest latch suferă de două mari nejunsuri:

- Potenţial flotant la ieşire: Cînd latch-ul este închis, valoarea ieşirii Q nu este trasă nici la un nivel logic. În acest caz nodul Q se numeşte flotant sau dinamic. Peste un oarecare timp, zgomotul şi scurgerile de pot schimba valoarea ieşirii Q.
- Fără bufer: Lipsa buferilor a adus la lucrul incorect al unelor microscheme comerciale. Un pic (virf) de zgimot, aduce la apariţia tensiunii negative la intrarea D şi poate activa tranzistorul nMOS, făcând latch-ul aprins, chiar de CLK=0. Analogic, picul la intrarea D, mai mare decât tensiunea alimentarii, poate deschide tranzistor pMOS, astfel, el poate fi deschis cu zgomotele la ieşirea Q, inflenţind asupra valorii intrarii D. Regula de bază este: nici poarta de transmisie, nici nodul stării circuitului logic secvenţial nici o dată nu trebuie să se folosească unde poate exista probabilitatea apariţiei zgomotelor.



În Figura 3.12(b) este reprezentat un D latch mai sigur din 12 tranzistori, ce se foloseşte în microschemele comerciale acltuale. Deşi el este creat la bază din porţi de transmisie cronometrare, în el sunt adăugaţi intevertori I1 şi I2, pentru a îndeplini rolul buferilor de intrare şi ieşire. Starea latch-ului se determină cu ajutorul nudului N1. Invetorul I3 şi buferul cu 3 stări T1 oferă feedback-ul pentru a transforma N1 într-un nod static. Dacă nodul N1 se va abate de starea staţionară sub influenţa zgomotelor, atunci, când CLK=0, buferul T1 îi va reântoarce valoarea logică validă.

În Figura 3.13 este demonstrat un D flip-flop construit din 2 latch-uri controlate de $\underline{CLK}$ şi CLK. Sau redus unii invertori în plus, şi acum pentru crearea unui flip-flop trebuie doar 20 de tranzistori.



3.2.8. Rezumare (Concluzii)

Latch-urile şi Flip-flop-urile sunt blocurile funcţionale fundamentale ale schemelor logice secvenţiale. Un D latch este deschis atunci când CLK=1, astfel permiţând senalului de la intrarea D să ajungă la ieşirea Q. D flip-flop transmite semnalul de la D la Q doar pe frontul semnalului de clock. În toate celelalte cazuri flip-flop-urile şi latch-urile îşi păstrază starea precedentă. Registrul se numeşte ansamblul din câteva D flip-flop-uri cu semnalul de clock comun.