Curs 5

3.2.5 Memoria fixă ROM

Memoriile fixe ROM sunt circuite integrate pe scară largă având schema bloc din fig. 3.29 şi fiind organizate sub forma unor cuvinte (în cazul nostru 2^n) de lungime dată (pentru fig. 3.29 lungimea dată este "m" biti).

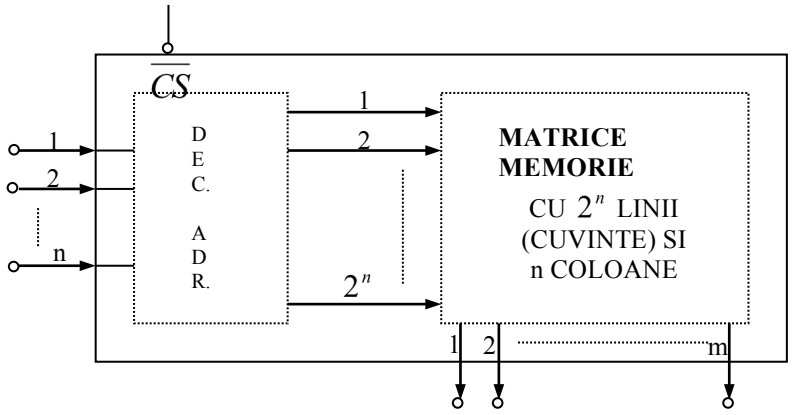


Figura 3.29

Memoria fixă are "n" intrări de adresare, o intrare (sau mai multe) de validare ($\overline{CS}$) şi "m" ieşiri de date. Fiecare cuvânt al memoriei este selectat printr-o combinaţie unică de valori ale variabilelor de adresare.

Cu o memorie fixă poate fi implementat un sistem de funcţii logice caz în care variabilele se aplică la intrările de adresare ale memoriei, iar valoarea funcţiilor logice este obţinută la ieşirile memoriei. De menţionat că fiecare funcţie logică este dată de un bit de la ieşire.

Din punct de vedere logic o memorie cu capacitatea de 2^n cifre binare este un selector cu "n" intrări de adresare şi 2^n intrări de date reprezentate de celulele memoriei (fig.3.30). Aşa cum s-a arătat şi la prezentarea multiplexorului, o astfel de structură realizează o funcţie logică de variabilele de adresare, funcţie ale cărei valori sunt specificate de intrările de date ale multiplexorului care, în cazul de faţă, sunt impuse de conţinutul celulelor de memorie.

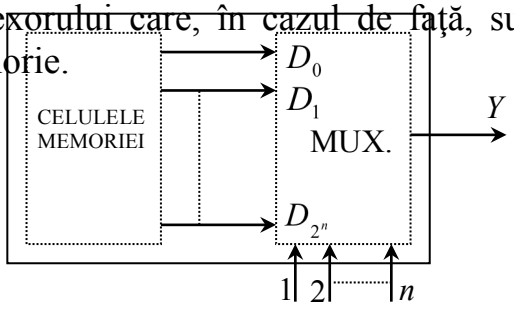


Figura 3.30

Considerând de exemplu o funcţie logică de trei variabile a, b, c, având tabelul de adevăr prezentat în tab.3.3, implementarea ei se poate face cu ajutorul unei memorii având capacitatea de memorare de opt cifre binare la intrările de adresare ale căreia se aplică cele trei variabile: a, b şi c si în care s-a înscris, la fiecare adresă, valoarea funcţiei corespunzătoare produsului logic standard ce corespunde adresei respective (fig.3.31).

a	b	c	f_1	f_2	f_3	f_4
0	0	0	0	1	0	1
0	0	1	0	0	1	1
0	1	0	0	1	1	1
0	1	1	1	0	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	1	0	0	1	1	1
1	1	1	1	1	0	0

Tabelul 3.4

a	b	c	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Tabelul 3.3

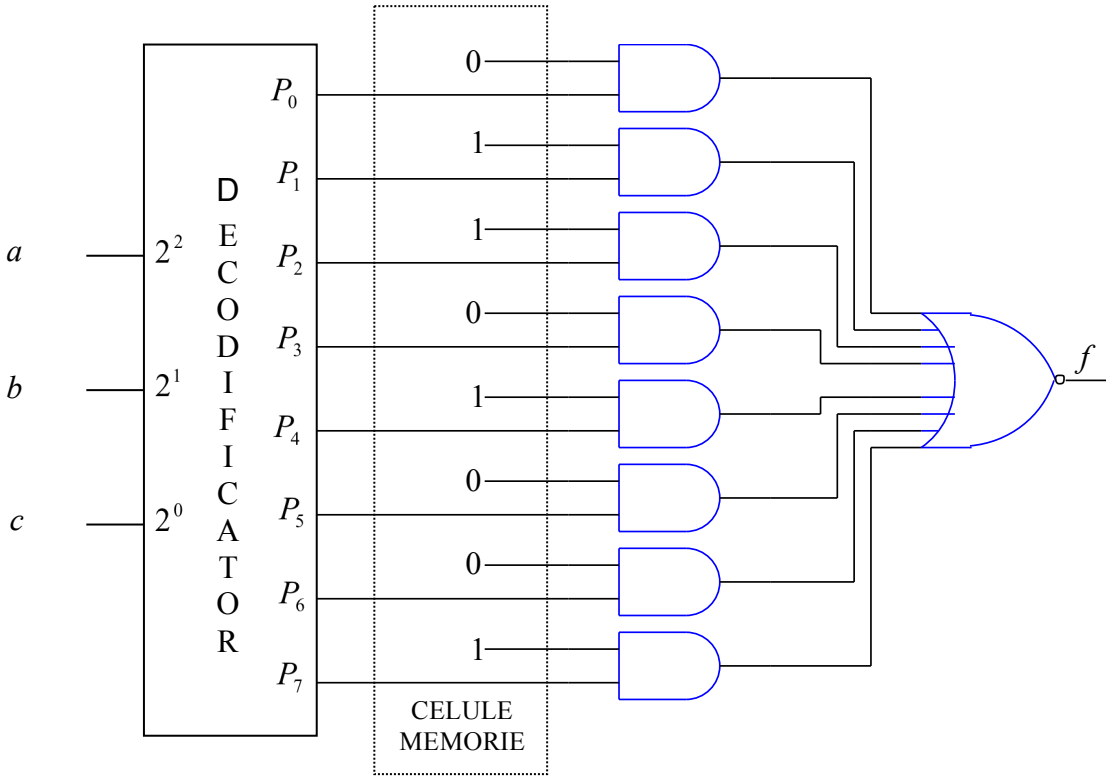


Figura 3.31

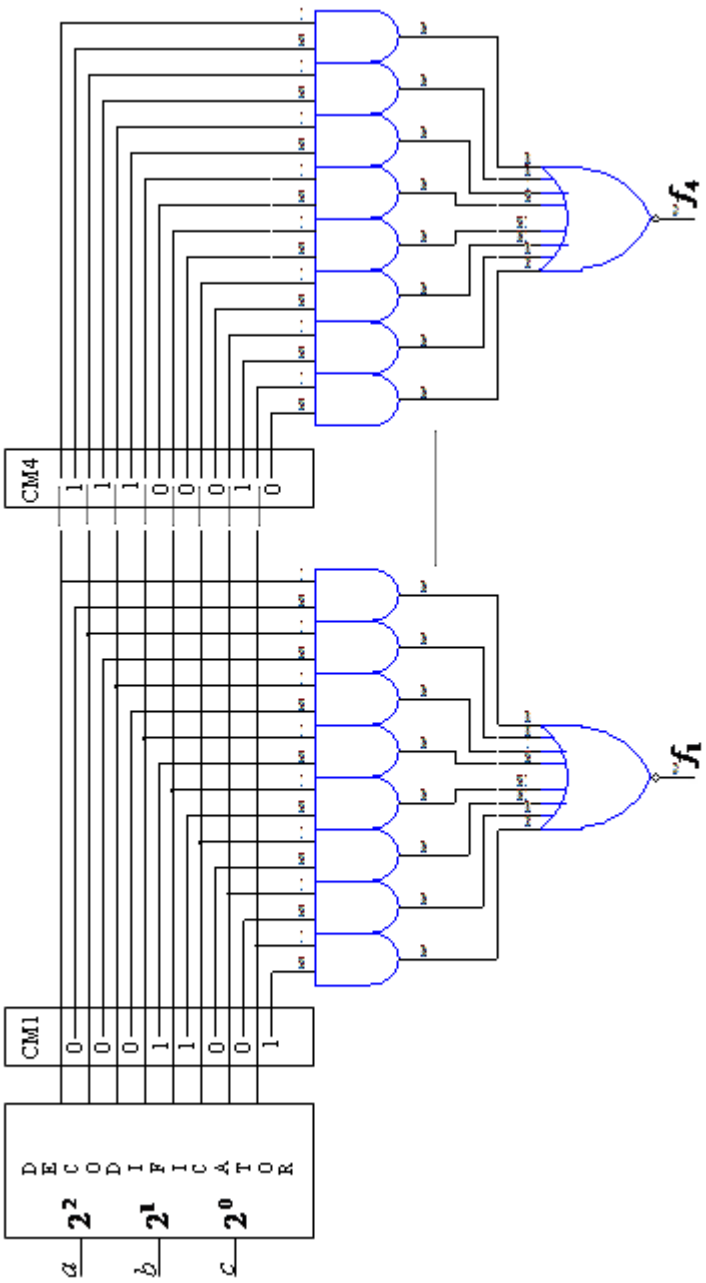


Figura 3.32

În cazul în care se urmăreşte implementarea unui sistem de funcţii logice se multiplică numărul celulelor de memorie ce sunt adresate de aceeaşi ieşire a decodificatorului de adresă. De exemplu în cazul în care se urmăreşte implementarea unui sistem de patru funcţii logice de trei variabile (tab.3.4) celulele de memorie se dispun matricial astfel ca o ieşire a decodificatorului să conducă la "citirea" a patru celule simultan.

De remarcat că în cazul implementării funcţiilor logice cu ajutorul memoriilor problema minimizării acestora se pune doar în unele cazuri

speciale, ce vor fi enumerate, deoarece, în mod obişnuit, în memorie se introduce chiar tabelul de definiţie al funcţiei.

Observaţie: La implementarea funcţiilor logice cu memorii fixe nu are importanţă complexitatea funcţiei. Structura memoriei ce trebuie folosită este dată de numarul de variabile şi numărul de funcţii. Rezultă deci că implementarea cu memorii fixe este recomandabilă atunci când funcţiile sunt complicate.

Problema care se pune la implementarea unui sistem de funcţii cu o memorie fixă este aceea de a folosi o memorie cât mai mică. Se cere deci să se reducă pe cât posibil numărul variabilelor şi numărul funcţiilor.

Reducerea numărului variabilelor de intrare este deosebit de importantă deoarece capacitatea memoriei ce trebuie utilizată se dublează pentru fiecare variabilă de intrare. O metodă posibilă de reducere a numărului de variabile constă în codificarea acestora. De exemplu 16 variabile independente ce nu se pot modifica simultan pot fi codificate cu patru variabile binare.

În mod similar numărul funcţiilor implementate cu o memorie fixă poate fi redus dacă se foloseşte un decodificator extern care să decodifice funcţiile cerute din ieşirile memoriei.

3.2.6 Structuri logice programabile

Structurile logice programabile sunt circuite logice combinaţionale capabile să implementeze funcţii logice exprimate în formă normală disjunctivă. Structurile programabile conţin două matrici: o matrice de porţi SI care decodifică adresa şi o matrice de porţi SAU care generează funcţiile logice (un SAU logic între adresele pentru care funcţia ia valoarea logică 1).

Schema logică de principiu a unei structuri logice programabile cu 16 intrări şi 8 ieşiri având o matrice cu 48 de porţi SI cu câte 32 de intrări şi o matrice cu 8 porţi SAU cu câte 48 de intrări este prezentata în fig. 3.33.

Structurile logice programabile se clasifica in:

a) structuri PLA atunci cand sunt programate la fabricatie prin procedee de mascare;
b) structuri FPLA in cazul in care pot fi programate de catre utilizator.

Structurile logice programabile rezolvă problema codificării variabilelor de intrare, care a fost ridicată în cazul implementării funcţiilor logice cu memorii fixe, facilitând astfel implementarea unor funcţii logice de un număr foarte mare de variabile. Astfel, spre deosebire de memorii unde se decodifică toate adresele posibile, în cazul structurilor programabile sunt

decodificate numai adresele utilizate (produsele logice pentru care funcţia ia valoarea logică 1).

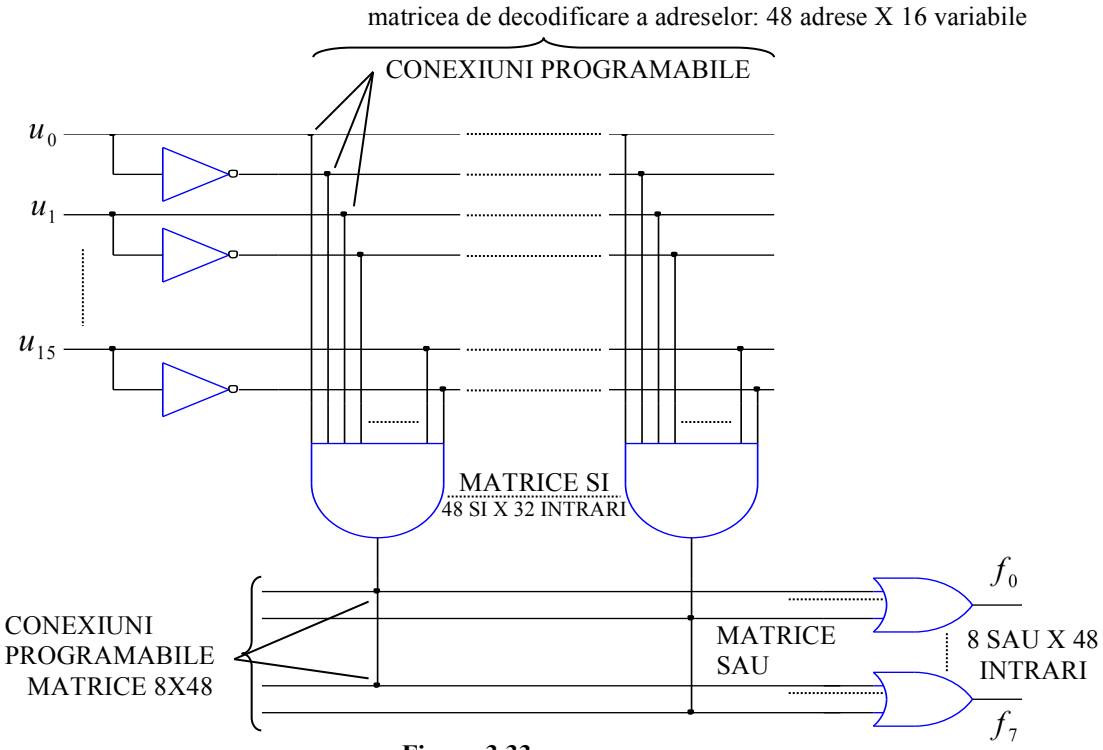


Figura 3.33

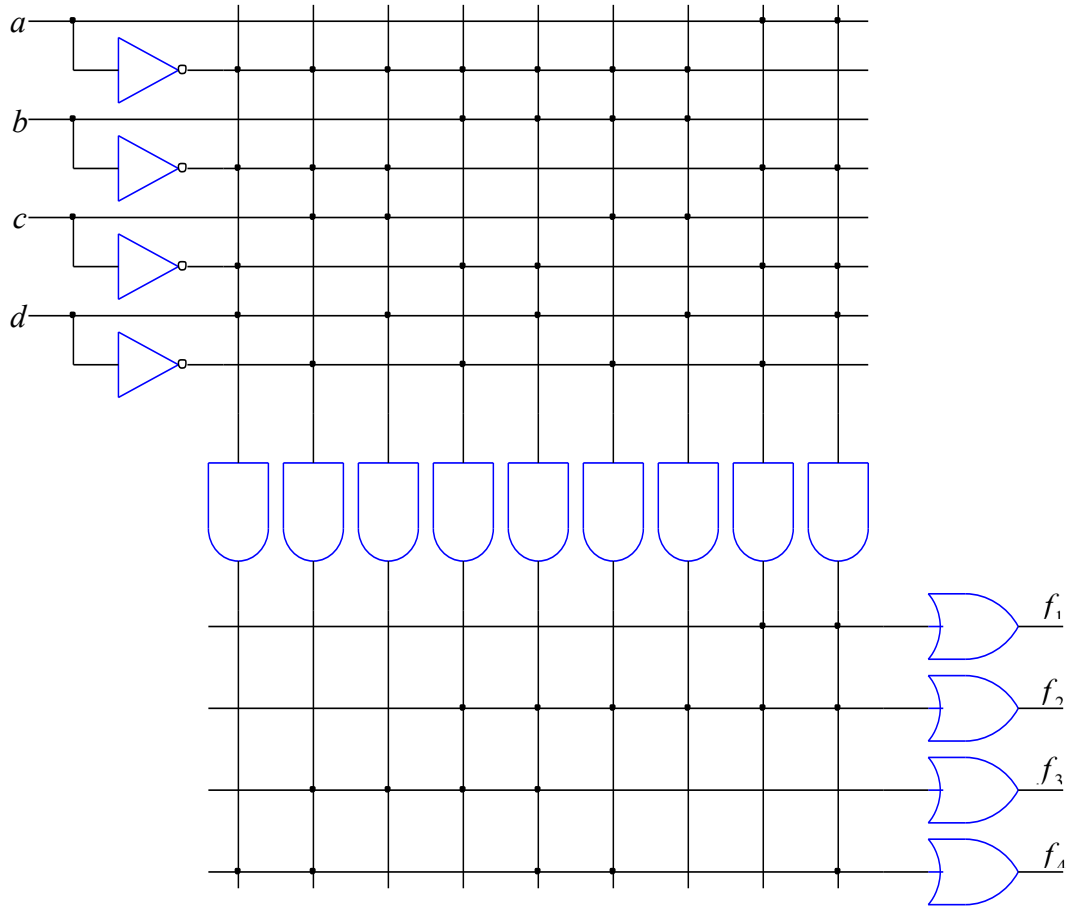


Figura 3.34

Adresele utilizate sunt specificate în formularul de programare. Pentru exemplificare, în tabelul 3.5 şi în fig. 3.34 sunt date formularul de programare şi schema logică simplificată a unei structuri logice cu 4 intrări şi 4 ieşiri care realizează conversia din cod zecimal codificat binar în cod Gray.

NR. ADRESA	INTRARI				IESIRI			
	a	b	c	d	f_1	f_2	f_3	f_4
1	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	1
3	0	0	1	0	0	0	1	1
4	0	0	1	1	0	0	1	0
5	0	1	0	0	0	1	1	0
6	0	1	0	1	0	1	1	1
7	0	1	1	0	0	1	0	1
8	0	1	1	1	0	1	0	0
9	1	0	0	0	1	1	0	0
10	1	0	0	1	1	1	0	1

Tabelul 3.5

Structura are în componenţa sa o matrice cu 9 porti SI a câte 8 intrări şi 4 porţi SAU cu câte 9 intrări. Cele patru funcţii f_1, f_2, f_3 si f_4 se obţin ca sume de produse logice.

La implementarea funcţiilor logice cu structuri programabile se pot folosi toate metodele de minimizare cunoscute. În practică însă minimizarea nu este necesară decât atunci când, în formă canonică, numărul de produse logice standard este mai mare decat numărul adreselor disponibile în structura programabilă. Minimizarea are sens numai în cazul în care se reuşeşte implementarea funcţiilor logice cerute cu o structură programabilă de dimensiuni mai mici, având în vedere că aceste structuri se produc în variante care diferă prin numărul de intrări, numărul de ieşiri şi numărul de adrese (cuvinte). Cum, pentru o aplicaţie dată, numărul de intrări şi de ieşiri este dat, rezultă că reducerea ce se poate obţine vizează numărul adreselor.

Pentru exemplul dat, cel al unui convertor din cod zecimal codificat binar în cod Gray se obţine următorul formular de programare şi structura din fig. 3.35.

$$f_1 = a$$
$$f_2 = a + b$$
$$f_3 = b.\bar{c} + \bar{b}.c$$
$$f_4 = c.\bar{d} + \bar{c}.d$$

NR. ADRESA	INTRARI				IESIRI			
	a	b	c	d	f_1	f_2	f_3	f_4
1	1	*	*	*	1	1	0	0
2	*	1	*	*	0	1	0	0
3	*	1	0	*	0	1	1	0
4	*	0	1	*	0	0	1	0
5	*	*	1	0	0	0	0	1
6	*	*	0	1	0	0	0	1

Tabelul 3.6

S-a obţinut o structură cu 6 adrese. Mai trebuie menţionat şi faptul că dacă pentru implementare s-ar fi folosit o memorie fixă, ar fi fost utilizate 16 adrese corespunzătoare celor 16 combinaţii posibil de realizat cu cele patru variabile de intrare.

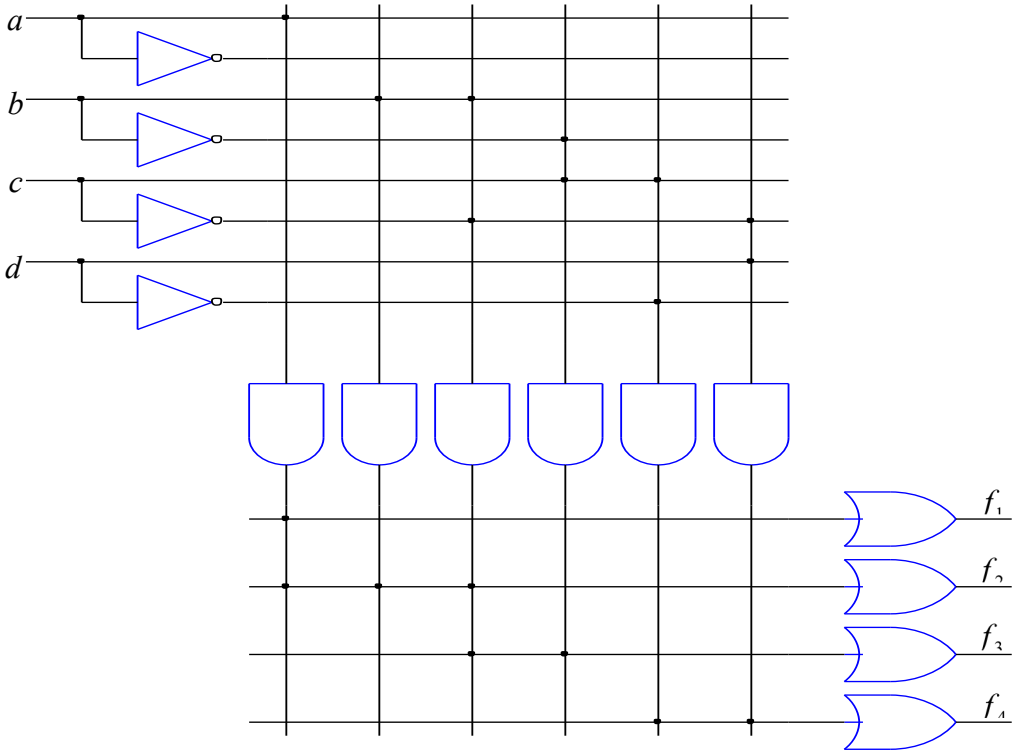


Figura 3.35

3.2.7 Aplicaţii ale circuitelor logice combinaţionale. Sumatorul

Sumatorul se defineşte ca fiind circuitul logic combinaţional care asigură, direct sau indirect, efectuarea însumării a două numere binare ţinând cont de un eventual transport iniţial. În fig. 3.39 este prezentat modul de reprezentare a unui sumator pe "n" biţi care ţine cont de un eventual transport iniţial T_i şi generează, dacă este cazul, un transport final T_f.

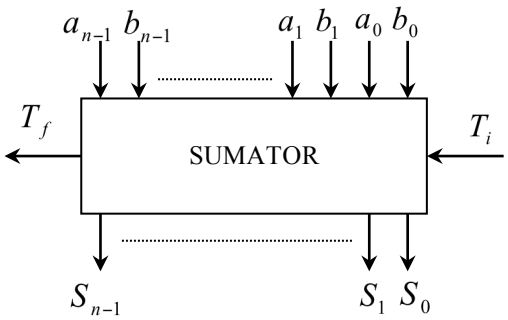


Figura 3.39

Circuitul logic combinaţional care asigură, direct sau indirect, însumarea a două numere binare cu câte un bit fără a lua în considerare transportul de la bitul cu ponderea imediat inferioară este denumit semisumator. Tabelul de adevăr al unui semisumator la intrarea căruia se aplică două numere binare de câte un bit, a_0 şi respectiv b_0, este prezentat în tab.3.7.

INTRARI		IESIRI	
a_0	b_0	S_0	T_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabelul 3.7

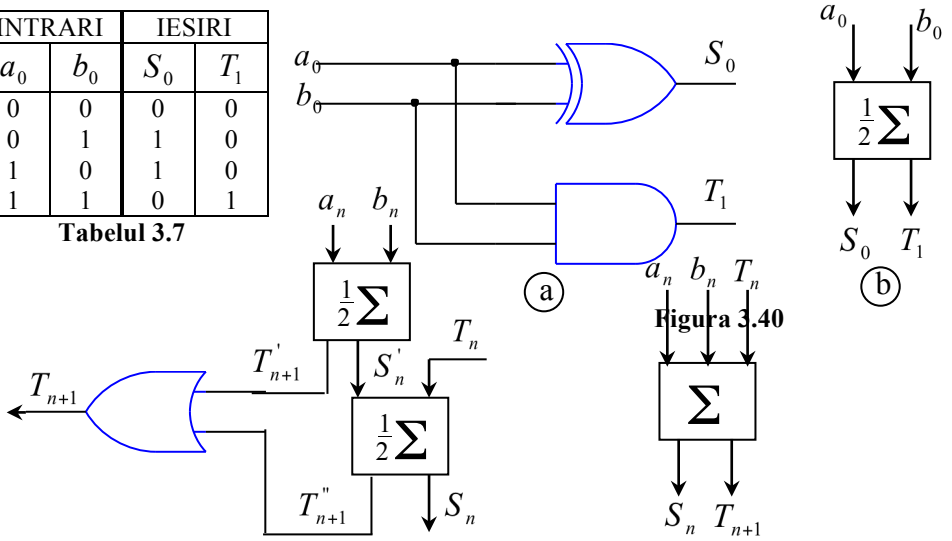


Figura 3.40

Figura 3.41

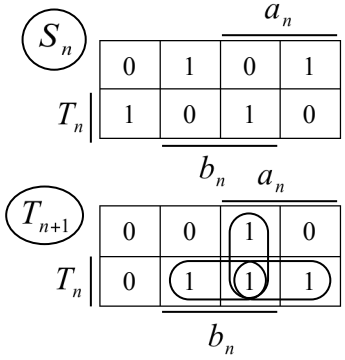
Din tabel rezultă următoarele expresii pentru variabilele de ieşire sumă (S_0) şi transport către bitul de rang imediat superior T_1

$$S_0 = \overline{a_0}.b_0 + a_0.\overline{b_0} = a_0 \oplus b_0$$
$$T_1 = a_0.b_0 \qquad (3.1)$$

expresii ce sunt implementate de schema logică din fig.3.40a. Reprezentarea simbolică a acestei scheme este redată în fig.3.40b.

T_n	a_n	b_n	S_n	T_{n+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabelul 3.8



$$S_n = \overline{a_n}\,\overline{b_n}T_n + \overline{a_n}b_n\overline{T_n} + a_nb_nT_n + a_n\overline{b_n}\,\overline{T_n}$$
$$T_{n+1} = b_nT_n + a_nb_n + a_nT_n$$

Figura 3.42

În principiu, sumatorul complet pentru un bit oarecare de rangul *n* se poate realiza utilizând două semisumatoare conform schemei din fig.3.41, unde cu T_n s-a notat transportul de la bitul de rang imediat inferior, iar cu T_{n+1} transportul către bitul de rang imediat superior. Primul semisumator efectuează o sumă parţială S_n' şi un transport partial T_{n+1}' în timp ce al doilea, ţinând cont de transportul T_n, generează suma S_n şi transportul T_{n+1}''. De remarcat că T_{n+1}' si T_{n+1}'' nu pot fi simultan egale cu 1.

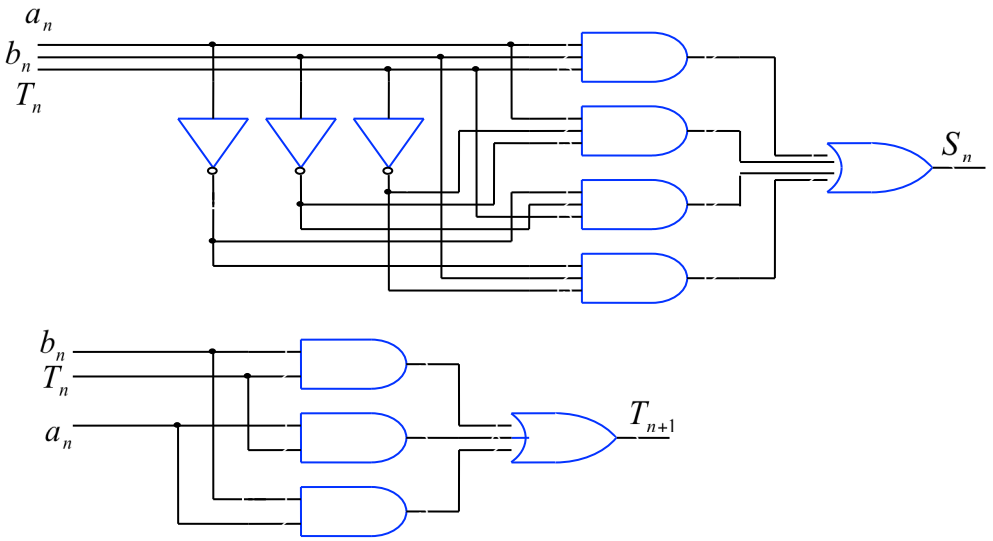


Figura 3.43

În practică însă nu se foloseşte această soluţie, ci se pleacă de la tabelul de adevăr al unui sumator, expresiile variabilelor de iesire minimizându-se cu una din metodele cunoscute (tabelul 3.8).

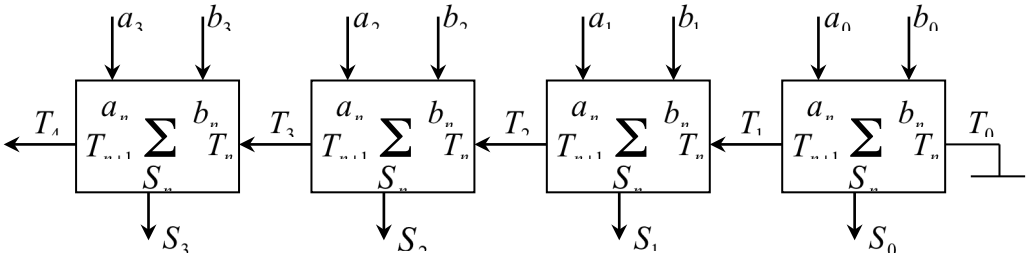


Figura 3.44

Schemele combinaţionale care implementează acest sumator sunt date în fig.3.43.În figura 3.44 este dată schema unui sumator multibit obţinut prin interconectarea mai multor sumatoare. Propagarea transportului este de tip serie.

3.3. Automatele de ordinul 1

Conform principiilor de clasificare definite în paragraful 3.1.2 trecerea de la un automat de ordinul zero la un automat de ordinul 1 se poate face prin introducerea unei reacţii negative. Exemplul cel mai simplu de automat de ordinul 1 îl constituie structura din fig. 3.45a şi ea corespunde unui circuit basculant bistabil asincron de tip RS care este cel mai simplu automat de ordinul 1. Extensia din fig. 3.45b constituie structura tot a unui automat de ordinul 1 şi ea poate fi materializată fizic de circuitul basculant bistabil RS sincronizat.

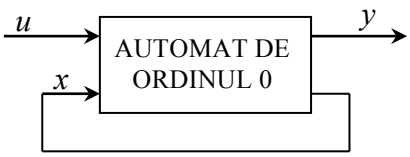


Figura 3.45a

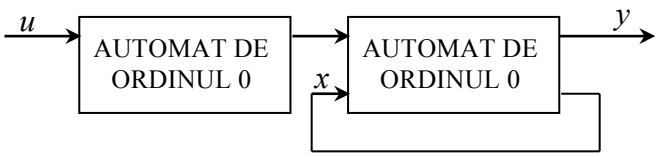


Figura 3.45b

Indiferent de circuitele care implementează un automat de ordinul 1 acesta posedă un comportament parţial independent de semnalele aplicate la intrare datorită reacţiei negative introduse. Autonomia datorată conexiunii cu reacţie face că ieşirile unui astfel de circuit să nu mai urmărească fidel toate variaţiile semnalelor de intrare astfel că numai un set limitat de comenzi va determina modificarea semnalelor de la ieşire. Existenţa unui set limitat de comenzi face că circuitul care implementează automatul de ordinul 1 să poata fi interpretat că un element de memorare (de zăvorâre->latch) numai a anumitor evenimente.

3.3.1. Circuitul basculant bistabil RS (latch-ul)

Circuitul basculant bistabil RS nesincronizat (sau asincron) constituie cel mai simplu element de memorare. Poate fi implementat de către două porţi SI-NU interconectate că în fig. 3.46, caz în care funcţionarea lui decurge conform tabelului 3.9.

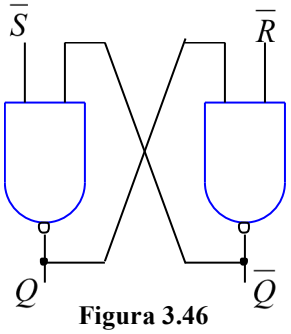


Figura 3.46

$\overline{S}$	$Q(t)$	$\overline{R}$	$Q(t+\tau)$
1	0	*	0
0	0	1	1
1	1	0	0
*	1	1	1

Tabelul 3.9

Cele două intrări ale circuitului, $\overline{R}$ şi $\overline{S}$, sunt active când la ele se aplică semnalul 0 logic. Prin convenţie se consideră că circuitul basculant bistabil este în starea "1" logic când la ieşirea Q avem nivel logic 1. Dacă Q=1 atunci la ieşirea complementară $\overline{Q}$ vom avea nivel logic 0.

Înscrierea unui semnal 1 logic în circuitul basculant bistabil (aducerea bistabilului în starea 1) se face prin aplicarea unui semnal 0 logic la intrarea de înscriere $\overline{S}$ (set). Ştergerea bistabilului, adică aducerea lui în starea 0, se face prin aplicarea unui semnal 0 logic la intrarea de ştergere $\overline{R}$ (reset). În tabelul 3.9 s-a notat cu $Q(t)$ starea prezentă şi cu $Q(t+\tau)$ starea după o întârziere τ, întârziere egală cu timpul necesar pentru propagarea semnalului de la intrare la ieşire.

Observaţie: la circuitul basculant bistabil RS nesincronizat având schema din fig. 3.46 nu se admite activarea simultană a semnalelor de înscriere şi de ştergere întrucât, dacă se forţează pe 0 logic pe ambele intrări, atunci ambele ieşiri, Q şi $\overline{Q}$ vor avea valoarea 1 logic, situaţie neadmisă întrucât ieşirile Q şi $\overline{Q}$ trebuie să fie complementare (circuitul este bi-stabil).

De reţinut că un circuit basculant bistabil RS nesincronizat se poate obţine şi dacă în structura din fig. 3.46 se înlocuiesc porţile SI-NU cu porţi SAU-NU, singura deosebire constând în faptul că intrările R şi S sunt active în acest caz când se aplică semnal logic 1 şi că, similar cu structura din figura 3.46 nu se admite comanda simultană a intrărilor R şi S.

Din analiza funcţionării circuitului basculant bistabil (CBB) RS asincron rezultă că, deşi prin intrările R şi S poate fi controlat modul său de functionare, momentul funcţionării, adică momentul la care au loc tranziţiile de stare, nu poate fi controlat, ceea ce reprezintă un inconvenient destul de mare pentru majoritatea sistemelor discrete de conducere ce utilizează CBB RS asincrone.

Dezavantajul poate fi înlăturat prin modificarea schemei din fig. 3.46 după modelul structurii din fig. 3.45b obţinându-se astfel CBB RS sincronizat (fig. 3.47).

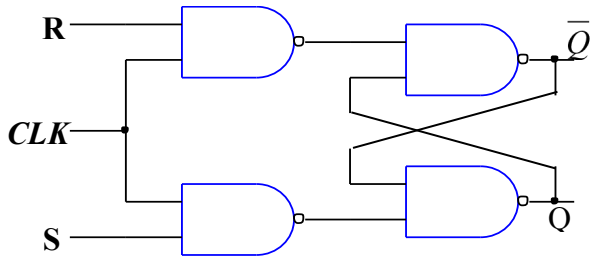


Figura 3.47

CLK		Q_n	Q_{n+1}
1	0	0	0
1	1	0	1
1	0	1	0
1	1	1	1
0	*	Q_n	Q_n

Tabelul 3.10

În cazul acestui CBB, dacă semnalul CLK=0, activarea semnalelor *R* şi *S* nu se face simţită la intrările latch-ului propriu-zis care este realizat cu porţile 3 şi 4. Dacă însă semnalul CLK=1 atunci intrările *R* şi *S* acţionează asupra intrărilor latch-ului propriu-zis în mod similar intrărilor $\overline{R}$ şi $\overline{S}$ din fig. 3.46. Rezultă deci că semnalul CLK (clock) marchează momentele în care intrările *R* şi *S* pot modifica starea CBB.

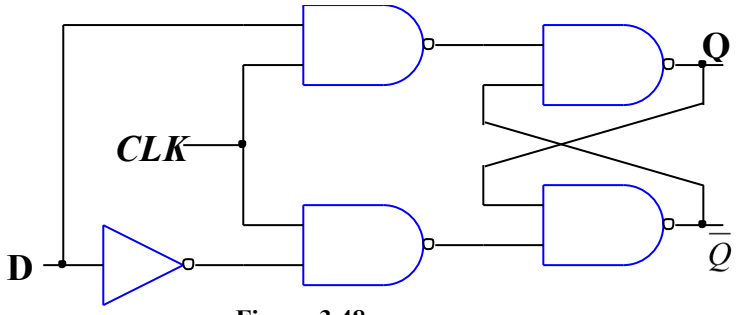


Figura 3.48

Soluţia prezentată este însă numai parţial eficientă din punctul de vedere al controlului momentului tranziţiilor de stare întrucât, dacă pe durata CLK=1 intrările *R* şi *S* îşi modifică în mod adecvat valoarea, momentul tranziţiei de stare depinde încă şi de semnalele *R* şi *S*. O rezolvare a acestei deficienţe constă în impunerea de restricţii privind momentul modificării intrărilor *R* şi *S* (nu când CLK=1) sau prin utilizarea structurilor de tip master-slave ce vor fi prezentate ulterior.

O altă restricţie, ce a fost menţionată deja, în functionarea CBB de tip RS constă în interdicţia ca intrările *R* şi *S* să fie comandate simultan. Soluţionarea constă în utilizarea unor latch-uri de tip D a căror structură este dată în fig. 3.48, iar funcţionarea lor este reflectată în tab. 3.10. Acest tip de bistabil nu permite intrărilor *R* şi *S* să aibă simultan aceeaşi valoare, dar menţine restricţia ca intrarea *D* să nu fie comandată pe palierul activ al semnalului de sincronizare CLK.

3.3.2 Memoria RAM

Structura logică a unei memorii RAM (random access memory), memorie cu acces aleator, se obţine dacă celor două nivele ale circuitului bistabil adresabil, nivelul de decodificare şi nivelul de memorare, li se adaugă un al treilea nivel şi anume, nivelul de multiplexare (fig. 3.49). Schema din fig.3.49 pune în evidenţă modul de funcţionare a unui circuit de memorie având capacitatea de 2^n biti. Schemele logice concrete ale acestor circuite conţin o serie de optimizări ale nivelului de decodificare şi a celui de multiplexare.

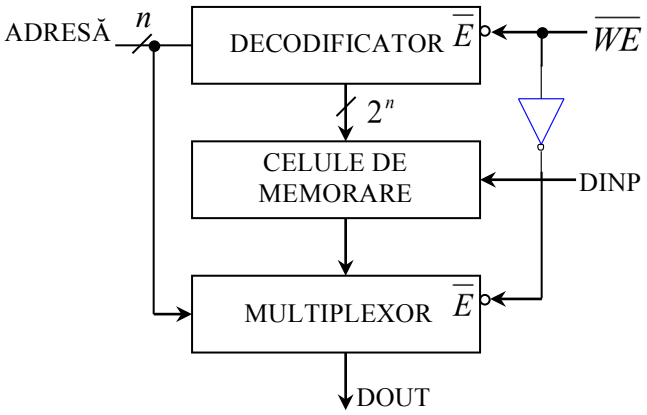


Figura 3.49

Modul de funcţionare al circuitului de memorie având schema din fig. 3.49 este următorul: decodificatorul selectează celula de memorie în care se va înscrie informaţia binară aplicată la intrarea DINP, iar multiplexorul permite selectarea oricărui bistabil la ieşirea DOUT. Circuitele decodificator şi multiplexor sunt adresate cu acelaşi cod binar denumit adresa locaţiei de memorie.

În afara intrărilor de adresare, intrare date DINP şi ieşire date DOUT, un circuit de memorie RAM mai are în mod uzual încă două intrări şi anume:

- $\overline{WE}$ (write enable) a cărui valoare condiţionează ciclul de scriere sau citire în/din memorie;
- $\overline{CS}$ care permite selectarea circuitului de memorie.

3.3.3 Principiul master-slave

Circuitul basculant bistabil adresabil şi memoria RAM reprezintă extensii de tip paralel ale circuitului basculant bistabil RS care este elementul tipic pentru automatele de ordinul 1. Aşa cum s-a precizat deja o deficienţă a acestor structuri o constituie imposibilitatea controlului eficient a momentului în care au loc tranziţiile de stare. Această deficienţă poate fi înlăturată prin utilizarea structurilor master-slave (fig. 3.50) care constituie extensii de tip serie ale structurilor tip.

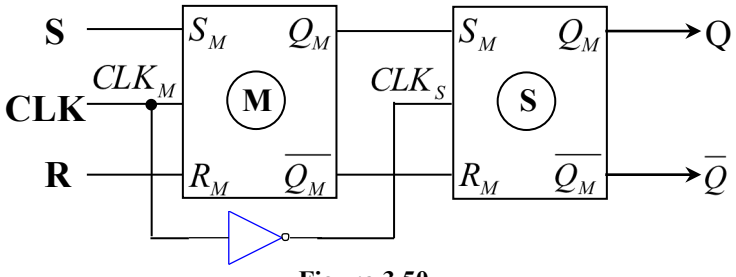


Figura 3.50

Conectarea în serie a celor două latch-uri, M(master) şi S(slave) are drept consecinţă modificarea ieşirilor Q şi $\overline{Q}$ pe frontul căzător (negativ) al impulsului de sincronizare, fenomen ce rezultă în urma analizării funcţionării schemei din fig. 3.50.

Astfel dacă semnalul CLK=1, bistabilul M poate să comute, funcţie bineînteles de semnalele logice aplicate la intrările R şi S pe toată durata palierului acestui semnal. În acest interval de timp însă bistabilul S este blocat (nu-şi poate modifica starea) deoarece semnalul $CLK_S = 0$. În momentul tranziţiei negative a semnalului CLK bistabilul M se blochează ($CLK_M = 0$) şi se deschide accesul la bistabilul S ($CLK_S = 1$), adică bistabilul S poate să-şi modifice starea (să basculeze) funcţie de ieşirile bistabilului M. Pe toată durata cât CLK=0 starea bistabilului S nu se va mai putea modifica întrucât ieşirile bistabilului M sunt stabile ($CLK_M = 0$). În concluzie, modificarea ieşirilor Q şi $\overline{Q}$ nu se poate produce decât pe frontul negativ al impulsului de sincronizare în conformitate cu valoarea anterioară a semnalelor R şi S. Comutarea se produce astfel la un moment de timp strict determinat de semnalul CLK (fig. 3.51).

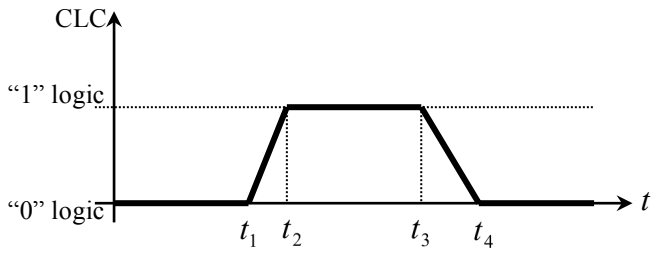


Figura 3.51

t_1. Se izolează bist. M de bist. S
t_2.Sunt validate datele de la intrările lui M
t_3.Sunt invalidate intrările de date
t_4.Se transferă datele din M în S şi se modifică ieşirile Q şi $\overline{Q}$

Într-un mod similar poate fi realizată şi o structură master-slave ce comută pe frontul pozitiv (crescător) al semnalului de sincronizare.

Dacă în structura din fig. 3.50 se introduce o poartă NU care să realizeze dependenţa $R = \overline{S}$ atunci se obţine un circuit bistabil de tip D (fig. 3.52) a cărui funcţionare este prezentată în tabelul 3.11 în care prin Q_n s-a notat starea bistabilului după "n" impulsuri de sincronizare, iar cu Q_{n+1} starea bistabilului D după "n+1" impulsuri de sincronizare.

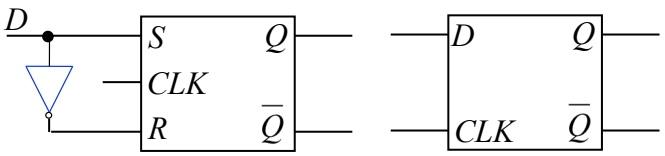


Figura 3.52

D	Q_n	Q_{n+1}
0	Q_n	0
1	Q_n	1

Tabelul 3.11

De remarcat că bistabilul de tip D ce funcţionează după principiul master-slave comută pe frontul impulsului de sincronizare în timp ce bistabilul D prezentat în paragraful 3.3.1 (fig. 3.48) poate comuta pe toată durata când CLK=1.

3.3.4 Registrul

3.3.4.1 Registrul serie

Un CBB master-slave, care este o extensie de tip serie a elementului tipic pentru automatul de ordinul 1, latch-ul, poate fi extins la rândul său prin conectare în cascadă (serie) formând o structură cu un număr par de latch-uri ce poartă denumirea de registru de deplasare serie. Utilizând CBB de tip master-slave poate fi realizată şi o extensie serie-paralel cunoscută sub denumirea de registru paralel de stocare. De asemenea se pot concepe şi structuri în care, prin modul în care sunt interconectate CBB de tip D cu ajutorul unor reţele combinaţionale, să se poată obţine atât registre serie, cât şi paralel, cu funcţionarea controlată de intrări de selecţie.

În fig. 3.53 este reprezentat un registru de deplasare serie realizat prin interconectarea a patru bistabile de tip D.

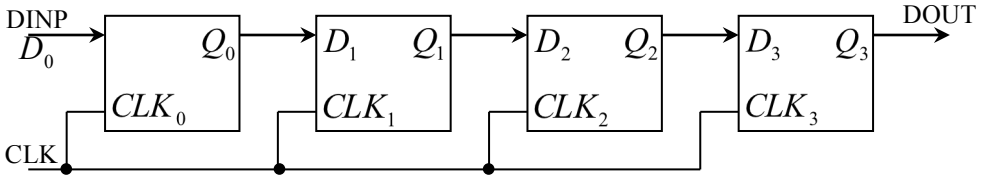


Figura 3.53

Datorită modului în care au fost interconectate cele patru bistabile şi a faptului că aplicarea semnalului de sincronizare se face simultan, modul de comutare al registrului poate fi descris astfel:

$$DINP^n = D_0^n = Q_0^{n+1}; Q_0^n = D_1^{n+1}; Q_1^n = D_2^n = Q_2^{n+1}; Q_2^n = D_3^n = Q_3^{n+1} = DOUT^{n+1} \quad (3.2)$$

de unde rezultă că $DOUT^n = DINP^{n-4}$.

3.3.4.2 Registrul paralel

Registrul paralel de stocare constituie o extensie de tip serie-paralel a latch-ului elementar şi este o structură realizată cu bistabile de tip D acţionate sincron, conform schemei din fig. 3.54.

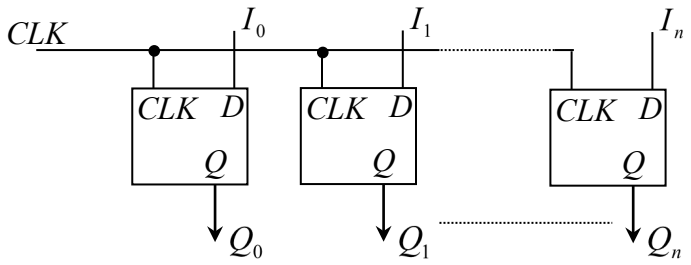


Figura 3.54

Funcţia principală a unui astfel de registru într-un sistem numeric de prelucrare a informaţiilor constă în stocarea temporară a unor informaţii binare care devin astfel mai uşor accesibile. Registrul constituie astfel memoria zonelor de viteză maximă în sistemele numerice de prelucrare a informaţiilor.

3.3.4.3 Registrul serie-paralel

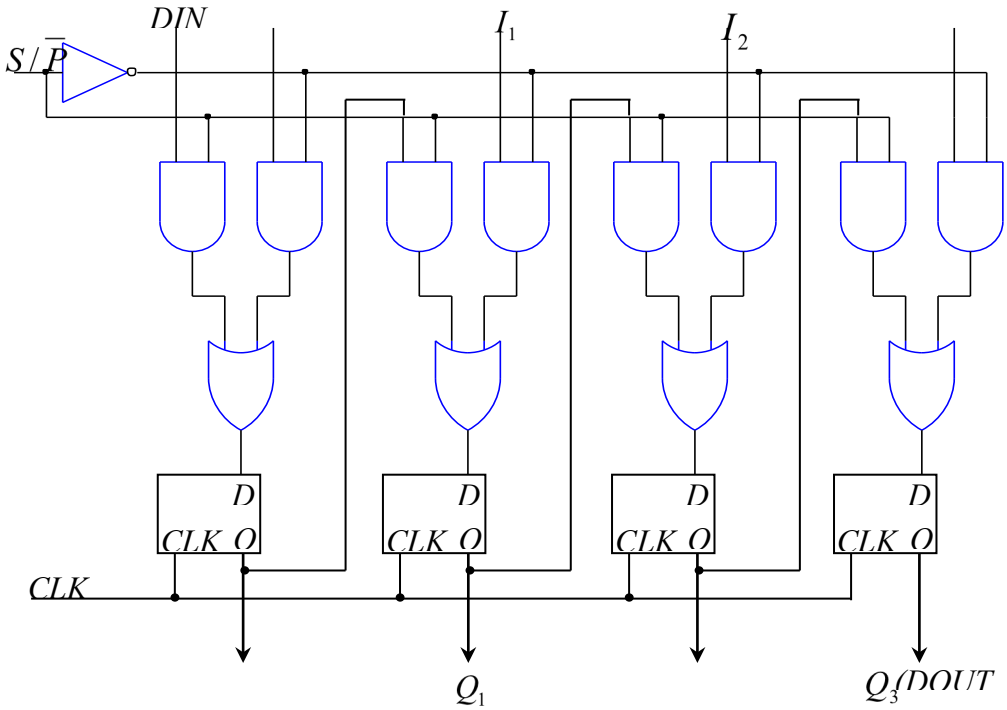


Figura 3.55

Registrele serie şi paralel sunt utilizabile doar în sistemele de conducere în care transferul datelor este de tip serie, respectiv paralel. Există însă multe situaţii în care, la nivelul sistemelor de conducere, se impune modificarea modului de transfer a datelor de la transferul serie la transferul paralel sau invers. Aceasta este cauza care a condus la apariţia unui nou tip de registru, aşa numitul registru serie-paralel.

Bistabilele de tip D din componenţa unui astfel de registru trebuie să poată primi date de la:

a) bistabilele de tip D anterioare, în cazul transferului serie;

b) din exteriorul registrului, în caz de transfer paralel.

Pentru aceasta registrul serie-paralel trebuie să conţină o zonă combinaţională care să selecteze sursa de date pentru bistabilele interne în funcţie de valoarea unui semnal de comandă ce selectează modul de lucru al registrului.

În fig.3.55 este prezentată schema unui registru serie-paralel pe patru biţi.

Structura combinaţională de pe intrările D_i ale bistabilelor îndeplineşte funcţia unui multiplexor cu două intrări de date (Q_{i-1} şi I_i) şi o intrare de adresare *S/P*. Astfel dacă *S/P*=1 registrul realizează o deplasare serie a datelor de la intrare (DINP) spre ieşire (DOUT). Dacă *S/P*=0, atunci este deschis accesul datelor I_0, I_1, I_2, I_3 la intrările de date ale bistabilelor şi primul impuls de tact va realiza încărcarea paralelă a registrului.

Interconectarea a două registre serie-paralel pe patru biţi conform schemei din fig. 3.56 conduce la obţinerea unui registru serie-paralel pe 8 biţi.

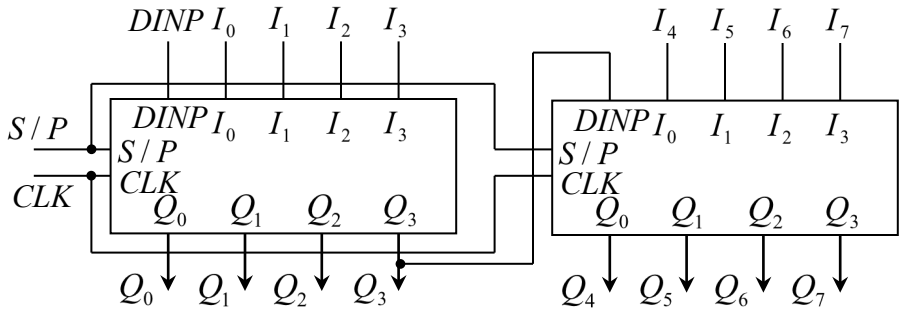


Figura 3.56

Un astfel de registru poate fi utilizat la conversia serie-paralel sau paralel-serie a unor cuvinte pe 8 biţi într-un sistem de conducere.

Conversia paralel-serie se poate realiza prin aplicarea la intrarea de tact a registrului aflat în regim de funcţionare serie (*S/P*=1), a 8 impulsuri de

tact, după ce în prealabil informaţia ce se doreşte a fi serializată a fost încărcată în registru în regimul *S/P*=0.