2. Bazele logice ale calculatoarelor numerice

2.1. Variabile şi funcţii logice

Caracteristica esenţială a tuturor generaţiilor de calculatoare numerice realizate până în prezent o constituie natura discretă a operaţiilor pe care acestea le efectuează. Considerente de ordin tehnologic impun utilizarea în construcţia calculatorului a dispozitivelor cu două stări care condiţionează codificarea informaţiei şi efectuarea calculelor în sistem binar.

Analiza şi sinteza circuitelor de comutaţie aferente calculatoarelor numerice utilizează ca principal instrument matematic algebra logică (booleană).
În continuare se prezintă unele elemente atât ale algebrei logice cât şi ale unor circuite logice fundamentale.

2.1.1. Algebra booleană

Fie mulţimile $\boldsymbol{M}=\{x_1, x_2,\ldots,x_n,\}$ cu $x_i \in \boldsymbol{Z}$ şi $\boldsymbol{O}=\{+,\bullet\}$ (componentele mulţimii $\boldsymbol{O}$ sunt două operaţii care vor fi definite ulterior). Structura $A=(M,O)$ reprezintă o algebră dacă:

a) mulţimea M conţine cel puţin două elemente;

b) mulţimea M reprezintă parte stabilă în raport cu cele două operaţii respectiv $x_1+x_2 \in$M, $x_1 \bullet x_2 \in M$ pentru orice $x_1, x_2 \in M$;

c) cele două operaţii au următoarele proprietăţi:

- comutativitate:

$x_1+x_2 = x_2+x_1;$

$x_1 \bullet x_2 = x_2 \bullet x_1,$

- asociativitate:

$(x_1+x_2)+x_3 = x_1+ (x_2 + x_3)$;

$(x_1 \bullet x_2) \bullet x_3 = x_1 \bullet (x_2 \bullet x_3)$

- distributivitatea uneia faţă de cealaltă:

$(x_1+x_2) \bullet x_3 = x_1 \bullet x_3 + x_2 \bullet x_3$;

$x_1+(x_2 \bullet x_3)= x_1 \bullet x_2 + x \bullet_1 x_3.$

d) mulţimea conţine un element nul - ***0*** şi unul unitate -***1*** care constituie elemente neutre faţă de cele două operaţii şi pentru cvare sunt valabile proprietăţile:

$x_1+0=0+x_1=x_1;$

$x_1 \bullet 1=1 \bullet x_1=x_1$ unde $x_1 \in M$.

e) fiecărui element $x \in M$ îi corespunde un unic invers $\overline{x} \in M$ cu proprietăţile :

$\overline{x} \cdot x = 0$ (principiul contradicţiei)

$\overline{x} + x = 1$ (principiul terţului exclus)

Dacă elementele mulţimii M pot lua numai două valori ***(0 şi 1)*** structura de mai sus reprezintă o *algebră booleană.*

La definirea axiomatică a algebrei s-au folosit notaţiile +, •, $\overline{x}$ pentru cele două legi de compoziţie, respectiv pentru elementul invers. În logică şi tehnică există denumiri şi semnificaţii specifice, evidenţiate în tabelul 2.1.

Tabelul 2.1

Matematică		Logică		Tehnică	
Denumire	Simbol	Denumire	Simbol	Denumire	Simbol
Prima operaţie	+	Disjuncţie	$\cup$	SAU	$\cup$
A doua operaţie	$\bullet$	Conjuncţie	$\cap$	ŞI	$\cap$
Element invers	$\overline{x}$	Negaţie	$\neg x$	NU	$\overline{x}$

Pornind de la axiome se deduc teoremele prezentate în tabelul 2.2 care se constituie în reguli de calcul în cadrul algebrei booleene.

Tabelul 2.2

Nr.	Denumire	Forma produs	Forma sumă
T1	Dublă negaţie (involuţia)	$\overline{\overline{x}} = x$	$\overline{\overline{x}} = x$
T2	Absorbţia	$x_1 \cdot (x_1 + x_2) = x_1$	$x_1 + x_1 \cdot x_2 = x_2$
T3	Elemente neutre	$x \cdot 0 = 0$	$x + 1 = 1$
T4	Idempotenţa (tautologia)	$x \cdot x \cdot \ldots \cdot x = x$	$x + x + \ldots + x = x$
T5	De Morgan	$\overline{x_1 \cdot x_2} = \overline{x_1} + \overline{x_2}$	$\overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}$

Oricare dintre cele 5 teoreme poate fi demonstrată utilizând axiomele cu ajutorul cărora s-a definit structura algebrei.

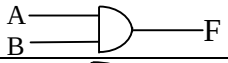
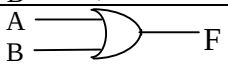
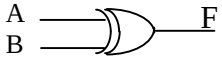
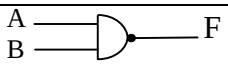
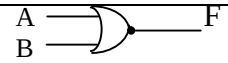
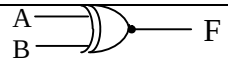
2.1.2. Funcţii logice importante

O funcţie $y=f(x_1, x_2, \ldots, x_n)$ reprezintă o funcţie logică dacă domeniul de definiţie este reprezentat de produsul cartezian $\{0,1\}^n$, cu alte cuvinte $\boldsymbol{f:\{0,1\}^n \rightarrow \{0,1\}}$.

Având în vedere această definiţie se poate spune că o funcţie logică (booleană) pune în corespondenţă o combinaţie binară asociată produsului cartezian cu una din valorile **0** sau **1**.

Domeniul de definiţie al unei funcţii logice de n variabile este format din 2^n puncte (combinaţii), iar numărul total de funcţii este de 2^{2^n}. De exemplu cu 2 variabile pot fi formate 16 funcţii, dintre care în tabelul 2.3 se prezintă cele mai importante.

Tabelul 2.3

Denumire funcţie	**Ecuaţie logică**	**Simbol**
ŞI	$F = A \cap B$	
SAU	$F = A \cup B$	
SAU EXCL.	$A \oplus B = A \bullet \overline{B} + \overline{A} \bullet B = \overline{A \otimes B}$	
NICI EXCL.	$A \otimes B = \overline{A \bullet B} + A \bullet B = \overline{A \oplus B}$	
ŞI- NU	$A \uparrow B = \overline{A \bullet B} = \overline{A} + \overline{B}$	
SAU - NU	$A \downarrow B = \overline{A + B} = \overline{A} \bullet \overline{B}$	

Funcţiile *ŞI, SAU, NU* se numesc *funcţii logice de bază* întrucât cu ajutorul lor se poate exprima orice altă funcţie logică. Ilustrarea semnificaţiei operatorilor logici se poate realiza prin diagrame *Venn*, tabele de adevăr, diagrame *Karnaugh*, scheme cu comutatoare etc.

Reprezentarea cea mai comodă şi pretabilă formalizării este cea realizată cu ajutorul tabelelor de adevăr. Pentru funcţiile din tabelul 2.3 se prezintă tabelul de adevăr 2.4.

Tabelul 2.4

A	B	ŞI	SAU	SAU EXCL	NICI EXCL	ŞI - NU	SAU - NU
0	0	0	0	0	1	1	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	1	0	1	0	0

Reprezentarea cu ajutorul *diagramei Karnaugh* constă în marcarea punctelor domeniului de definiţie într-o diagramă plană şi precizarea valorilor funcţiei în fiecare din aceste puncte. De exemplu în figura 2.1 este reprezentată diagrama Karnaugh pentru o funcţie de trei variabile cu marcarea vecinătăţilor punctului *010*.

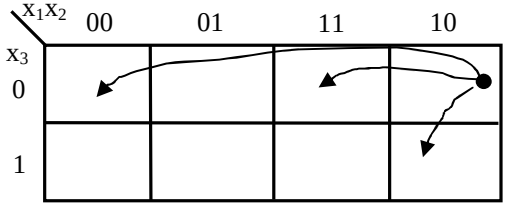


Fig. 2.1. Diagrama Karnaugh pentru o funcţie de trei variabile.

După cum se observă, trecerea de la o combinaţie la alta pe laturile diagramei Karnaugh se face prin modificarea unui singur bit.

O funcţie logică se poate reprezenta dezvoltat în două forme şi anume:

- *forma disjunctivă canonică (FDC),* cu utilizarea constituenţilor unităţii;
- *forma conjunctivă canonică (FCC),* cu utilizarea constituenţilor lui zero.

FDC presupune exprimarea funcţiei ca o *disjuncţie de conjuncţii (*reuniune de intersecţii) în care variabilele care au valoarea ***0*** *se consideră negate.*

FCC presupune exprimarea funcţiei ca o *conjuncţie de disjuncţii (*intersecţie de reuniuni) în care variabilele care au valoarea ***1*** *se consideră negate.*

2.1.3. Minimizarea funcţiilor logice

Minimizarea unei funcţii booleene implică reducerea la minimum a numărului de variabile şi a simbolurilor de funcţii implicate în reprezentarea acesteia. Metodele de minimizare pot fi încadrate în două categorii: *analitice* şi *grafice.*

Metodele analitice constau în principal din calcule efectuate în funcţia dată pe baza axiomelor şi teoremelor algebrei binare.

Metodele grafice presupun constituirea unor tabele sau matriţe de combinaţii, din care prin grupări şi asocieri corespunzătoare rezultă reduceri. Din categoria acestor metode, în continuare se vor face referiri la cea care utilizează diagrama Karnaugh.

După cum s-a văzut, două celule adiacente într-o diagramă Karnaugh diferă prin valoarea unei singure variabile. Dacă termenilor din două asemenea celule li se aplică proprietatea de distributivitate şi principiul terţului exclus se elimină variabila care îşi schimbă valoarea.

Referitor la acest procedeu de reducere şi implicit de minimizare pot fi formulate următoarele observaţii:

a) un grup de 2^m celule vecine ocupate cu unităţi permite eliminarea a m variabile;

b) pentru reducere, fiecare celulă trebuie să facă parte dintr-o grupare, dar poate fi inclusă în mai multe;

c) cel mai avansat grad de simplificare se obţine dacă unităţile dintr-o diagramă Karnaugh sunt grupate într-un număr minim de grupări fiecare grup conţinând un număr minim de unităţi;

d) pentru a putea aplica în mod succesiv proprietatea de distributivitate şi teorema terţului exclus, numărul unităţilor din grupările formate trebuie să fie o putere întreagă a lui 2.

Reguli similare pot fi deduse şi pentru deducerea formei conjunctive minime. În acest caz, în diagrama Karnaugh se vor grupa zerourile. Se va scrie apoi disjuncţia grupurilor de zerouri vecine, iar forma minimă va fi conjuncţia grupurilor de coordonate.

Etapa care succede minimizării este aceea a implementării funcţiei logice. Această implementare se realizează cu elemente de comutaţie de diverse tipuri cum ar fi: contacte şi relee, porţi logice etc.

2.2. Circuite logice combinaţionale

Caracteristica principală a circuitelor logice combinaţionale *(CLC)* o reprezintă dependenţa mărimilor de ieşire ale acestora *numai* de combinaţiile aplicate la intrare, nu şi de timp.

Schema bloc a unui CLC este prezentată în figura 2.11. Acesta dispune de intrările x_0, x_1, … x_{m-1} şi generează în exterior ieşirile y_0, y_1, …, y_{n-1}. Funcţionarea CLC poate fi descrisă cu ajutorul unei funcţii logice (de comutaţie).

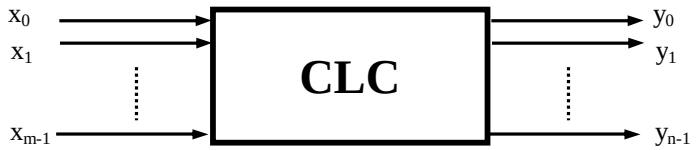


Fig. 2.2. Circuit logic combinaţional

Analiza *CLC* pleacă de la cunoaşterea schemei acestuia şi urmăreşte stabilirea funcţionării, concretizată prin tabela de adevăr sau prin scrierea expresiilor variabilelor de ieşire funcţie de cele de intrare.

Sinteza *CLC* presupune parcurgerea următoarelor etape pentru stabilirea structurii circuitului:

- definirea funcţiilor logice;
- minimizarea acestora;
- obţinerea schemei circuitului.

În structura unui calculator numeric se întâlnesc numeroase tipuri de *CLC* între care reprezentative sunt: *convertoarele de cod, codificatoarele şi decodificatoarele, multiplexoarele şi demultiplexoarele, comparatoarele, detectoarele şi generatoarele de paritate, ariile logice programabile, memoriile şi circuitele aritmetice.*

În continuare vor fi prezentate elemente privind sinteza unor CLC uzuale din structura unui calculator numeric.

2.2.1. Convertoare de cod

Convertoarele de cod sunt CLC care permit trecerea dintr-un cod binar în altul. Sinteza unui asemenea CLC se va exemplifica pentru un convertor *din cod binar în cod Gray*. În figura 2.3 se prezintă elementele aferente sintezei acestui tip de convertor, în care $B_3\,B_2\,B_1\,B_0$ reprezintă cuvântul binar aplicat la intrare, iar $G_3\,G_2\,G_1\,G_0$ cuvântul binar obţinut la ieşire.

B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

a

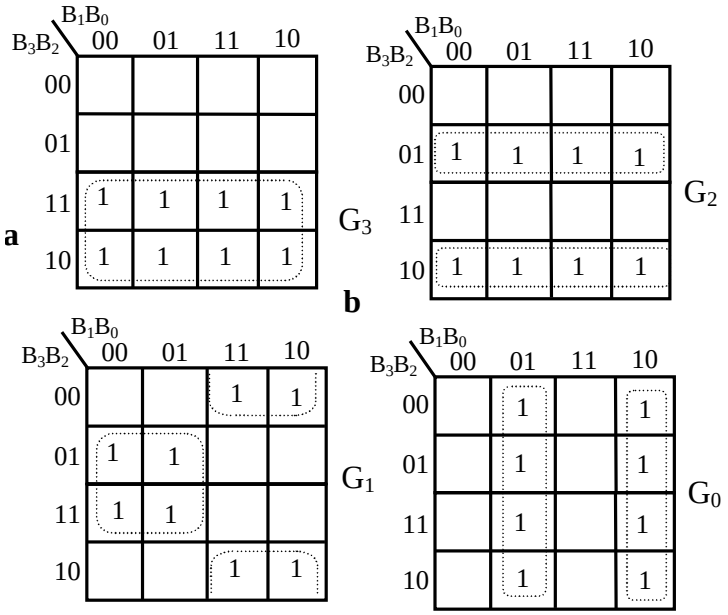


Fig. 2.3. Convertor de cod binar natural - Gray:
a - tabela de corespondenţă;
b - diagramele Karnaugh asociate

Făcând reducerile în diagramele Karnaugh rezultă:

$$G_3 = B_3$$

$$G_2 = \overline{B}_2 B_3 + \overline{B}_3 B_2 = B_2 \oplus B_3$$

$$G_1 = \overline{B}_1 B_2 + \overline{B}_2 B_1 = B_1 \oplus B_2$$

$$G_0 = \overline{B}_1 B_0 + \overline{B}_0 B_1 = B_1 \oplus B_0$$

În figura 2.4 se prezintă două variante de implementare ale relaţiilor de mai sus.

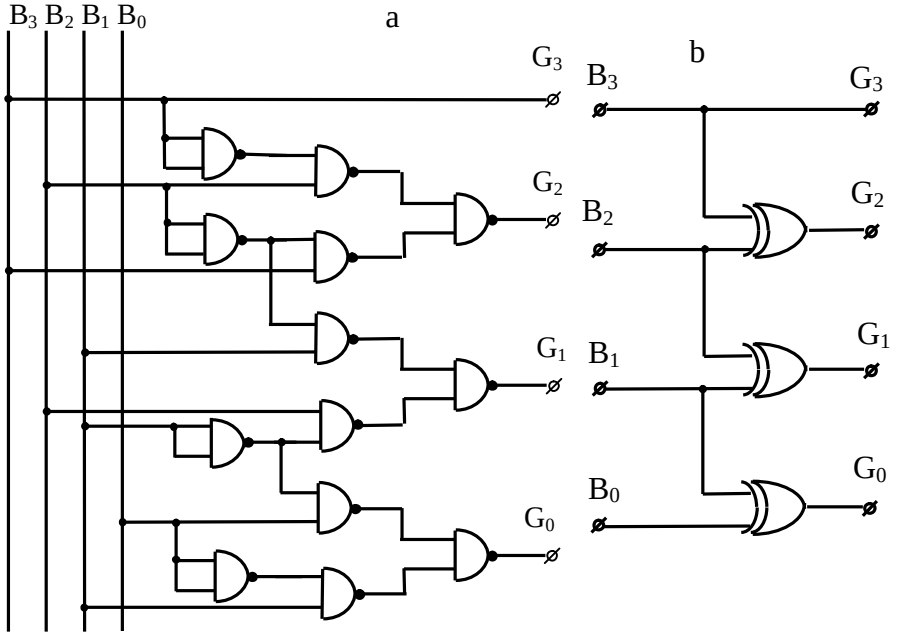
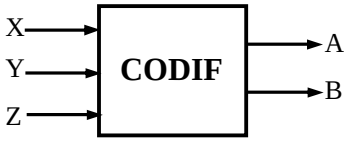


Fig. 2.4. Schema convertorului din cod binar natural în cod Gray:
a - realizarea cu porţi *NAND*; b - realizarea cu circuite *SAU EXCLUSIV* .

2.2.2. Codificatoare şi decodificatoare

Codificatoarele sunt *CLC* la care activarea unei intrări, dintr-un grup de *m,* conduce la apariţia unui cuvânt de cod la ieşire format din *n* biţi ($m \le 2^n$). În figura 2.5 se prezintă elemente aferente unui codificator cu m=3 şi n=2.



X	Y	Z	A	B
0	0	0	0	0
1	0	0	0	1
0	1	0	1	0
0	0	1	1	1

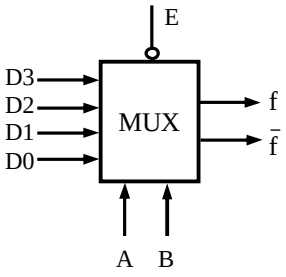
$$A = \overline{X} \cdot (Y \oplus Z)$$
$$B = \overline{Y} \cdot (X \oplus Z)$$

Fig. 2.5. Codificator cu m=3 şi n=2 (schema bloc, tabela de adevăr, funcţii logice.

Decodificatoarele sunt *CLC* care activează una sau mai multe ieşiri funcţie de cuvântul de cod aplicat la intrare. Decodificarea este necesară în aplicaţii care se referă la adresarea memoriilor, afişarea numerică, multiplexarea datelor etc.

2.2.3. Multiplexoare şi demultiplexoare

Multiplexoarele sunt *CLC* care permit transferul datelor de la una din intrările selectate cu o adresă (cuvânt de selecţie) către o ieşire unică. Din punct de vedere funcţional *MUX* pot fi privite ca o reţea de comutatoare comandate. *MUX* pot fi analogice sau numerice, ultimele fiind specifice CN. În continuare se va face sinteza unui MUX 4:1 numeric şi implementarea cu porţi logice.



E	B	A	f = Canal
0	0	0	D3
0	0	1	D2
0	1	0	D1
0	1	1	D0
1	*	*	-

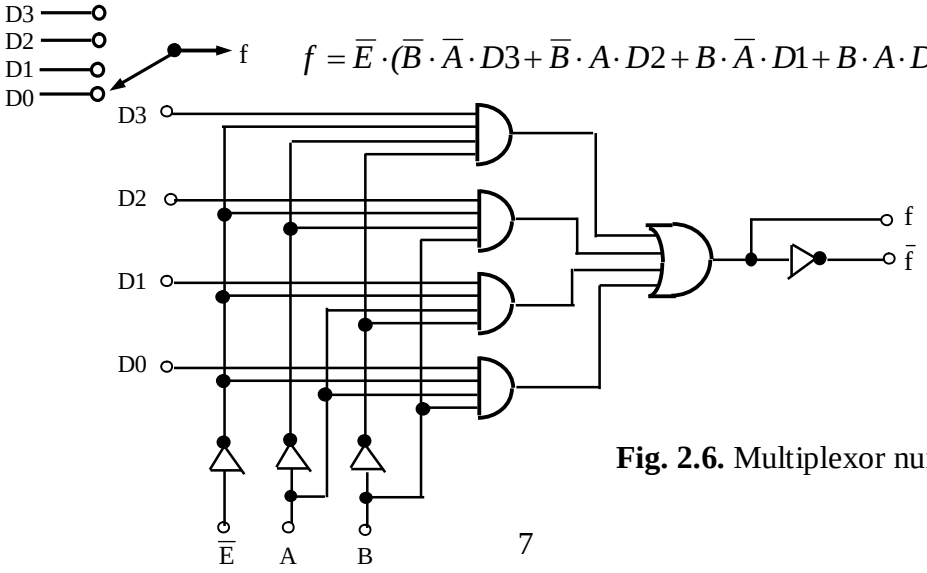
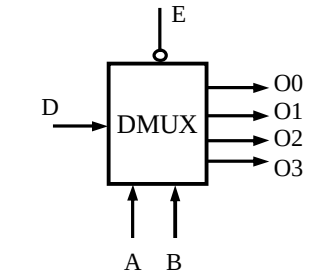
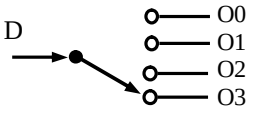


Fig. 2.6. Multiplexor numeric 4:1.

Demultiplexoarele sunt *CLC* care realizează transmiterea datelor de la o unică intrare către o ieşire selectabilă cu ajutorul unui cuvânt de selecţie (adresă). Ca şi *MUX* demultiplexoarele reprezintă practic o reţea de comutatoare comandate, putând fi numerice sau analogice. În figura 2.7 se prezintă elemente specifice sintezei unui DMUX numeric 1:4.



E	B	A	O0	O1	O2	O3
0	0	0	D			
0	0	1		D		
0	1	0			D	
0	1	1				D
1	*	*	-	-	-	-



$O0 = \overline{E} \cdot \overline{B} \cdot \overline{A} \cdot D \quad O2 = \overline{E} \cdot B \cdot \overline{A} \cdot D$

$O1 = \overline{E} \cdot \overline{B} \cdot A \cdot D \quad O3 = \overline{E} \cdot B \cdot A \cdot D$

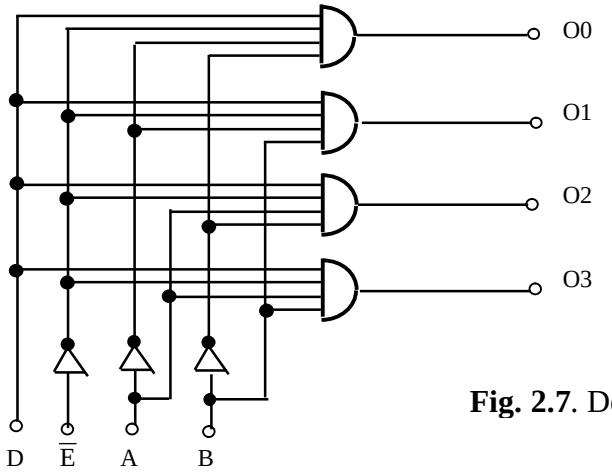


Fig. 2.7. Demultiplexor 1:4

2.2.4. Circuite de complementare

Circuitul de complementare este un *CLC* care funcţie de comenzile aplicate realizează una din următoarele funcţii:

- complementează faţă de unu biţii cuvântului de la intrare;
- lasă cuvântul de la intrare neschimbat;
- forţează în unu toţi biţii cuvântului de la ieşire;
- forţează în zero toţi biţii cuvântului de la ieşire.

În figura 2.8 se prezintă elementele aferente unui circuit de complementare pe 4 biţi. Din tabela de adevăr se obţin următoarele funcţii logice ale ieşirilor:

$$Y_1 = \bar{x}_1 \overline{A}\overline{B} + x_1 \overline{A}B + A\overline{B} = (\overline{x_1 \oplus B}) + A\overline{B}$$

$$Y_2 = \bar{x}_2 \overline{A}\overline{B} + x_2 \overline{A}B + A\overline{B} = (\overline{x_2 \oplus B}) + A\overline{B}$$

$$Y_3 = \overline{x}_3\overline{A}\overline{B} + x_3\overline{A}B + A\overline{B} = (\overline{x_3 \oplus B}) + A\overline{B}$$

$$Y_4 = \overline{x}_4\overline{A}\overline{B} + x_4\overline{A}B + A\overline{B} = (\overline{x_4 \oplus B}) + A\overline{B}$$

a căror implementare s-a realizat cu porţi *ŞI, SAU* şi *SAU-EXCLUSIV*.

Comenzi		I e ş i r i			
A	B	y_1	y_2	y_3	y_4
0	0	$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_3$	$\overline{x}_4$
0	1		x_2	x_3	x_4
1	0	1	1	1	1
1	1	0	0	0	0

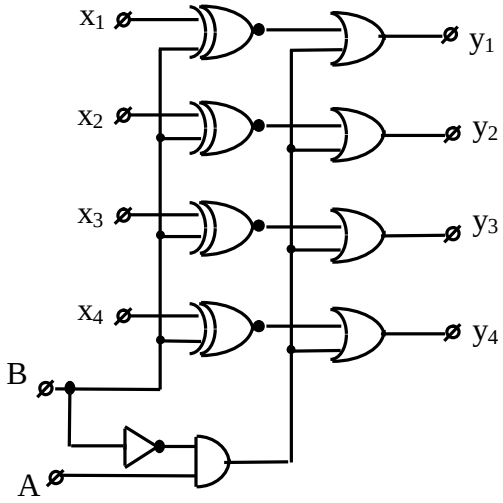
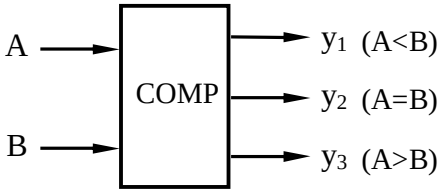


Fig. 2.8. Circuit de complementare pe 4 biţi.

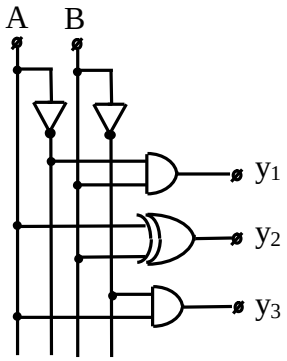
2.2.5. Comparatoare

Comparatoarele numerice sunt *CLC* care permit determinarea relaţiei existente între două numere. Ieşirile unui comparator sunt reprezentate de *trei funcţii* care corespund tipului de relaţie existent între numerele aplicate la intrare *(<,=,>)*.

În figura 2.9 sunt prezentate elemente aferente sintezei unui comparator pe un bit.



A	B	y_1	y_2	y_3
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0



$y_1 = \overline{A} \cdot B$

$y_2 = \overline{A} \cdot \overline{B} + A \cdot B = \overline{A \oplus B}$

$y_3 = A \cdot \overline{B}$

Fig. 2.9. Comparator pe un bit.

Prin interconectarea mai multor comparatoare pe un bit se obţin comparatoare pentru cuvinte binare formate din mai mulţi biţi.

2.2.6. Detectoare de paritate

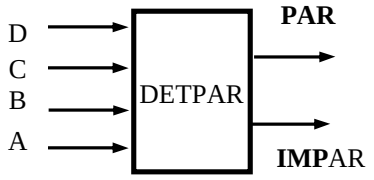
Detectoarele de paritate sunt *CLC* cu *n* intrări şi două ieşiri *PAR* şi *IMPAR* care sunt complementare. Ieşirea *PAR* are valoarea **1** atunci când numărul de valori logice **1** în combinaţia de la intrare este *par* şi **0** atunci când acest număr este *impar*.

În figura 2.10 se prezintă elementele aferente sintezei unui detector de paritate cu *n=4* intrări.

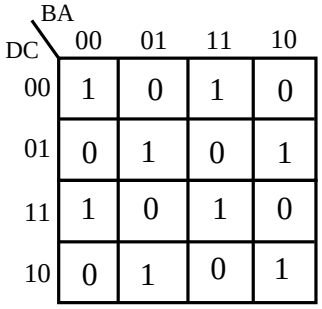
După cum se observă în tabela de adevăr funcţiile *PAR* şi *IMPAR* sunt complementare, respectiv IMPAR= $\overline{PAR}$. Din această cauză în figura 2.10 a fost reprezentată diagrama Karnaugh pentru funcţia *PAR*. Aşa cum reiese din diagramă, nu se poate opera nici o reducere asupra funcţiei care va fi:

$$PAR = \overline{D}\,\overline{C}\,\overline{B}\overline{A} + \overline{D}\,\overline{C}BA + \overline{D}C\overline{B}A + \overline{D}CB\overline{A} + D\overline{C}\,\overline{B}A + D\overline{C}B\overline{A} + DC\overline{B}\overline{A} + DCBA$$

În relaţia de mai sus prin aplicarea proprietăţilor operaţiilor logice rezultă:



a



c

D	C	B	A	PAR	IMP
0	0	0	0	1	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	0	1
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	0

b

Fig. 2.10. Detector de paritate: a – schema bloc; b - tabela de adevăr; c - diagrama Karnaugh a funcţiei PAR.

$$PAR = \overline{D}\,\overline{C}(\overline{B}\overline{A} + BA) + \overline{D}C(\overline{B}A + B\overline{A}) + D\overline{C}(\overline{B}A + B\overline{A}) + DC(\overline{B}\overline{A} + BA)$$

$$PAR = (\overline{D}\,\overline{C} + DC)(\overline{B}\overline{A} + BA) + (\overline{D}C + D\overline{C})(\overline{B}A + B\overline{A})$$

Dar

$$\overline{D}\,\overline{C} + DC = \overline{\overline{\overline{D}\,\overline{C} + DC}} = (\overline{\overline{\overline{D}\,\overline{C}}}) \cdot (\overline{\overline{DC}}) = \overline{(\overline{\overline{D}} + \overline{\overline{C}}) \cdot (\overline{D} + \overline{C})} = \overline{(D + C) \cdot (\overline{D} + \overline{C})} =$$

$$= \overline{D\overline{C} + \overline{D}C} = \overline{D \oplus C}$$

$$\overline{B}\overline{A} + BA = \overline{B \oplus A}$$

Notăm

$$D \oplus C = C \oplus D = Y$$

$$B \oplus A = A \oplus B = X$$

Rezultă:

$$PAR = \overline{YX} + YX = \overline{Y \oplus X} = \overline{(D \oplus C) \oplus (B \oplus A)}$$

$$IMPAR = \overline{PAR} = (D \oplus C) \oplus (B \oplus A)$$

relaţii a căror implementare se prezintă în figura 2.11.

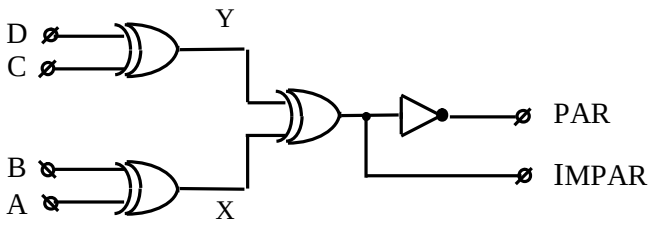


Fig. 2.11. Implementarea detectorului de paritate.

2.2.7. Sumatoare

Semisumatorul elementar, pentru care schema logică şi tabela de adevăr sunt prezentate în figura 2.12, adună două numere a câte un bit x_i, y_i şi generează la ieşire 2 biţi: suma s_i şi transportul c_i către rangul următor.

x_i	y_i	S_i	c_{i+1}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

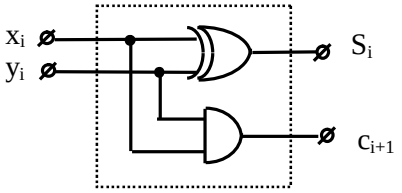


Fig. 2.12. Semisumatorul elementar.

Schema din fig. 2-22 a rezultat pe baza relaţiilor:

$$s_i = \bar{x}_i y_i + x_i \bar{y}_i = x_i \oplus y_i ;$$

$$c_{i+1} = x_i y_i .$$

Sumatorul elementar este un *CLC* care adună două numere binare x_i, y_i cu un transport de intrare c_i, generând la ieşire doi biţi: suma s_i şi transportul c_{i+1} către rangul superior, conform tabelului 2.5.

Tabelul 2.5

x_i	y_i	c_i	S	c_{i+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

Din tabelul 2.5 rezultă:

$$s_i = \bar{x}_i \bar{y}_i c_i + \bar{x}_i y_i \bar{c}_i + x_i \bar{y}_i \bar{c}_i + x_i y_i c_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = \bar{x}_i y_i c_i + x_i \bar{y}_i c_i + x_i y_i \bar{c}_i + x_i y_i c_i = c_i (x_i \oplus y_i) + x_i y_i$$

Relaţiile de mai sugerează obţinerea sumatorului elementar din două semisumatoare conform figurii 2.13.

Pentru adunarea a două cuvinte de *n* biţi este necesar să se înserieze *n* astfel de sumatoare ca în figura 2.14.

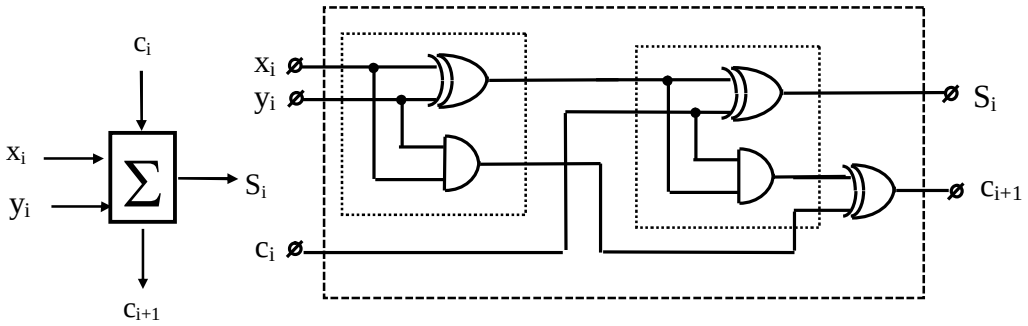


Fig. 2.13. Sumatorul elementar.

Cele două numere care urmează a se aduna se găsesc în registrele A şi B, iar rezultatul în registrul C. Transportul este depus într-un bistabil exterior care pentru un microprocesor este indicatorul de transport CY (Carry).

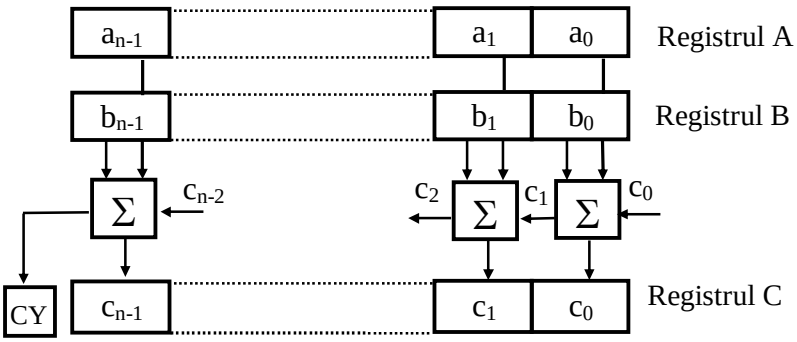


Fig. 2.14. Sumator pentru cuvinte de n biţi.

2.3. Circuite logice secvenţiale

Circuitele logice secvenţiale *(CLS)* sunt circuite ale căror mărimi de ieşire, la un moment dat, depind atât de combinaţia mărimilor de intrare, cât şi de starea sa.

Modelul matematic al *CLS*, pentru un anumit moment de timp *t*, este definit de două seturi de ecuaţii care reflectă tranziţia *stărilor* şi pe cea de *ieşirilor* şi care pot fi grupate în cvintuplul

$C_S=(X,Y,Q,f,g)$,

unde:

$X=\{x_1,x_2,\ldots x_n\}$ este mulţimea variabilelor binare de intrare;

$Y=\{y_1,y_2,\ldots y_m\}$ - mulţimea variabilelor binare de ieşire;

$Q=\{q_1,q_2,\ldots q_p\}$ - mulţimea variabilelor binare de stare;

$f: X \times Q \to Q$ - funcţia de tranziţie a stărilor;

$g: X \times Q \to Y$ - funcţia de tranziţie a ieşirilor.

Funcţiile de tranziţie a *stărilor* respectiv *ieşirilor* sunt de forma

$$q_i^{'} = f(x_1, x_2, \ldots, x_n, q_1, q_2, \ldots, q_p) \quad i=1,2,\ldots,p$$

$$y_k = g(x_1, x_2, \ldots, x_n, q_1, q_2, \ldots, q_p) \quad k=1,2,\ldots,p$$

Funcţiile $q_i^{'}$ şi y_k reflectă procesele de modificare a *stărilor* respectiv *ieşirilor,* ambele dependente doar de intrări şi de starea actuală

Din punct de vedere al structurii CLS conţin elemente combinaţionale şi elemente de memorie, figura 2.15.

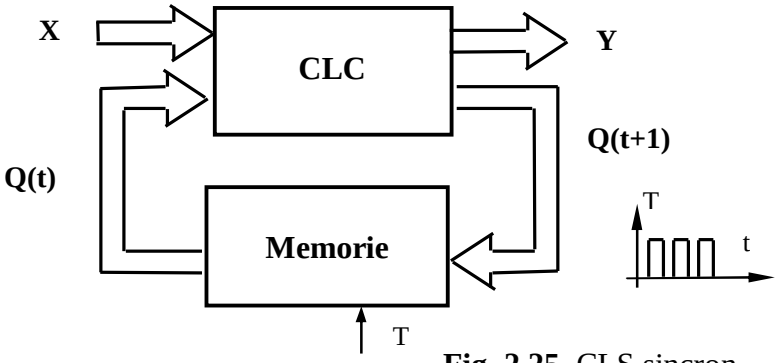


Fig. 2.25. CLS sincron.

În ceea ce priveşte modul de schimbare a stării elementelor de memorie există două tipuri de *CLS*:

• *CLS sincrone* la care modificarea stării se face sincron cu *un impuls de tact,* în funcţie de intrări şi de starea curentă;

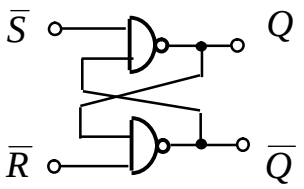
• *CLS asincrone* la care modificarea stării se produce la momente aleatoare depinzând numai de intrări şi de starea curentă.

În continuare vor fi prezentate pentru CLS care se regăsesc în structura unui CN cum ar fi: *bistabile, numărătoare, registre, circuite de memorie.*

2.3.1. Circuite basculante bistabile

Circuitele basculante bistabile (CBB) au două stări stabile la ieşire, iar prin aplicarea unor semnale de comandă trec dintr-o anume stare în starea complementară. Practic un *CBB* implementează un element de memorie care păstrează *un bit* de informaţie. Între cele mai răspândite *CBB* sunt cele de tip *RS, D, T şi JK.*

- *Bistabilul RS asincron* este format din două porţi NAND, fiecare având drept una din intrări ieşirea celeilalte (figura 2.26).



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

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

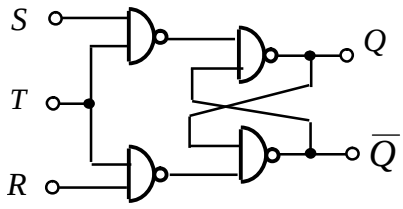
Fig. 2.26. B-RS asincron: schema logică, tabelele de adevăr si de excitatie.

Tabelul de adevăr defineşte *starea ieşirii funcţie de intrări* iar tabelul de excitaţie *defineşte intrarea care determină o anumită evoluţie a ieşirii* – intrările *S* (Set) şi *R* (Reset) sunt active pe *0*.

Combinaţia ***00*** la intrare este interzisă deoarece ieşirile sunt identice şi nu complementare. Combinaţia ***11*** la intrare păstrează starea anterioară *Q(t),* ieşirea fiind o nedeterminare faţă de intrare. Dacă înainte de ***11*** a fost la intrare ***10*** ieşirea este ***0,*** iar dacă înainte de ***11*** a fost ***01*** ieşirea este ***1.***

Se observă că pentru a memora ***1*** (pentru seta *CBB*) trebuie aplicată combinaţia $\overline{S}$ =0 , $\overline{R}$ =1 în timp ce combinaţia $\overline{S}$ =1 , $\overline{R}$ =0 determină memorarea cifrei binare ***0*** (resetarea *CBB*).

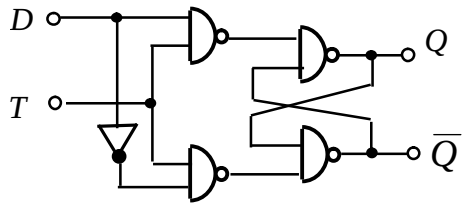
- *Bistabilul RS sincron.* La acest tip de CBB modificarea stării este determinată de un impuls de tact T (figura 2.17). Se observă că CBB RS sincron derivă din cel asincron prin adăugarea unor porţi suplimentare acţionate pe una dintre intrări de semnalul de tact.



T	S	R	Q(t+1)	$\overline{Q}$ (t+1)
1	0	0	Q(t)	$\overline{Q}$ (t)
1	**0**	**1**	**0**	**1**
1	**1**	**0**	**1**	**0**
1	1	1	ned	ned
0	x	x	Q(t)	$\overline{Q}$ (t)

Fig. 2.17. B-RS sincron: schema logică, tabelul de adevăr .

- *Bistabilul D* are o asemenea structură (figura 2.18) încât permite eliminarea stării de nedeterminare specifice *CBB* RS sincron, respectiv a acelei stări pentru care *S=R=1*.

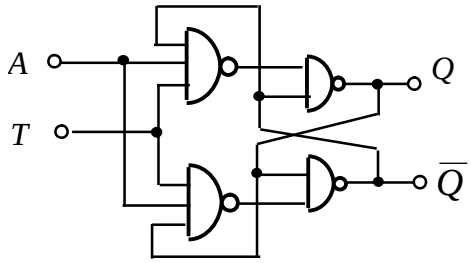


T	D	Q(t+1)	$\overline{Q}$ (t+1)
0	0	Q(t)	$\overline{Q}$ (t)
0	1	Q(t)	$\overline{Q}$ (t)
1	**0**	**0**	**1**
1	**1**	**1**	**0**

Fig. 2.18. Bistabilul D: schema logică, tabelul de adevăr .

Valoarea logică aplicată la intrarea *D* se transferă la ieşire doar la aplicarea semnalului de ceas (deci cu o întârziere de un tact).

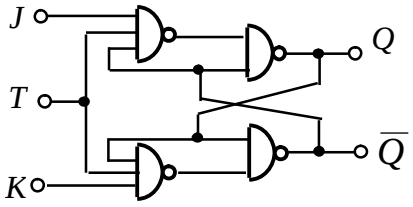
- *Bistabilul T* are proprietatea că schimbă starea la fiecare impuls de tact, dacă o intrare de validare *A* are ***1*** logic (figura 2.19). Dacă *A=0* starea bistabilului (respectiv ieşirea *Q*) rămâne neschimbată.



T	A	Q(t+1)	$\overline{Q}$ (t+1)
0	x	Q(t)	$\overline{Q}$ (t)
1	0	Q(t)	$\overline{Q}$ (t)
1	1	$\overline{Q}$ (t)	Q(t)

Fig. 2.19. Bistabilul T: schema logică, tabelul de adevăr .

- *Bistabilul JK* este un bistabil sincron care admite comenzi simultane pe ambele intrări fără a prezenta o stare instabilă. După cum se observă din figura 2.20 acesta are în plus faţă de *RS* două intrări de reacţie care sunt activate simultan cu semnalele de comandă. Conform tabelului de adevăr la combinaţia *J=0, K=1* ieşirea *Q=0*, iar la combinaţia *J=1, K=0* ieşirea *Q=1*. Circuitul funcţionează şi dacă pe ambele intrări se aplică ***1*** respectiv dacă *J=K=1*.



J	K	Q(t+1)	$\overline{Q}$ (t+1)
0	0	Q(t)	$\overline{Q}$ (t)
0	**1**	**0**	**1**
1	**0**	**1**	**0**
1	1	$\overline{Q}$ (t)	Q(t)

Fig. 2.20. CBB-JK sincron: schema logică, tabelul de adevăr .

2.3.2. Numărătoare

Numărătoarele sunt *CLS* care numără (contorizează) impulsurile aplicate la intrare şi memorează rezultatul.

În funcţie de sistemul de numeraţie folosit se întâlnesc numărătoare *binare, hexazecimale, decadice etc.* Un numărător care poate număra atât înainte cât şi înapoi se numeşte *reversibil*. Practic un numărător realizează, pentru un număr natural *N,* operaţia de identificare a claselor de resturi modulo C ($\overline{0}, \overline{1}, \ldots \overline{c-1}$).

De exemplu, un numărător *modulo 10* va avea aceeaşi stare *3* pentru oricare din următoarele numere aplicate la intrare. *N=3, 13, 23, 33, …, 103, 113, …, 203, 313*. Numărul maxim înscris într-un numărător modulo *c* este *c-1,* deoarece pentru *N=c,* acesta va indica zero.

Numărătoarele asincrone sunt cele la care informaţia de la intrare se propagă spre ieşire pas cu pas.

Numărătoarele sincrone sunt caracterizate prin aceea că toţi bistabilii care le compun basculează simultan funcţie de informaţiile aplicate la intrare şi de semnalul de tact.

Numărătoarele în buclă sunt registre de deplasare a căror ieşire este conectată la intrare.

2.3.3. Registre

Registrele sunt *CLS* destinate memorării vectorilor binari. Numărul de biţi egal cu numărul elementelor de memorie reprezintă *capacitatea registrului* sau *lungimea* cuvântului registru. În mod obişnuit registrele sunt constituite dintr-un set de bistabile şi o logică combinaţională auxiliară. Fiecare bit D_i al unui cuvânt binar este păstrat într-un bistabil B_i unde *i=0,1, …,n-1* (registrul are capacitatea de a memora *n* biţi iar *i* este rangul bistabilului *Bi*).

Registrele pot efectua o serie de operaţii cum ar fi:

a) **încărcarea** datelor serială sau paralelă – figurile 2.21 a,b;

b) **deplasare** date stânga sau dreapta - figurile 2.21 c,d;

c) **rotaţie** stânga sau dreapta - figurile 2.21 e,f;

d) **ştergere**.

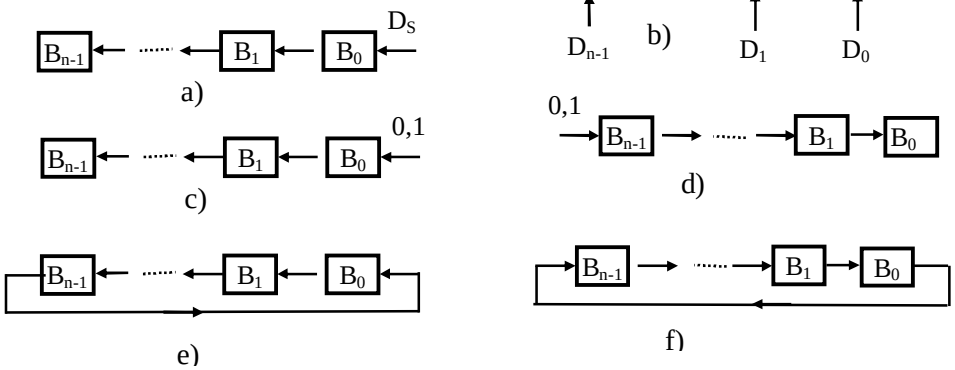


Fig. 2.21. Operaţii cu registre: D_{n-1} – MSB, D_0 – LSB.

- Încărcarea serială se realizează prin *n* impulsuri de tact:

`$B_i \leftarrow B_{i-1}$, $i=1,2,\ldots,n-1$, $R_0 \leftarrow D_S$,

unde *Ds* sunt biţii care se încarcă în B_0 după ce are loc deplasarea spre stânga. Un astfel de registru este folosit la un receptor la care datele sosesc serial pe o linie de 1 bit. Acestea sunt împachetate în cuvinte a câte *n* biţi şi transmise apoi paralel.

- Încărcarea paralelă se realizează într-un singur impuls de tact

$B_i \leftarrow D_i$, $i=0,1,\ldots,n-1$,

un exemplu de utilizare fiind încărcarea unui cuvânt de *n* biţi de pe o magistrală sau dintr-un alt registru.

- Deplasările stânga/dreapta sunt asemănătoare încărcării seriale, cu deosebirea că se execută un singur pas:

$B_i \leftarrow B_{i-1}$, $i=1,2,\ldots,n-1$, $B_0 \leftarrow$ *0 sau 1* (stânga),
$B_{i-1} \leftarrow B_i$, $i=1,2,\ldots,n-1$, $B_{n-1} \leftarrow$ *0 sau 1* (dreapta),

în bistabilele B_0 sau B_{n-1} încărcându-se *0* sau *1* funcţie de contextul utilizării registrului.

- Rotaţiile stânga/dreapta sunt asemănătoare deplasărilor cu deosebirea că bitul care părăseşte registrul reintră în registru (în B_0 la *rotaţia stânga* şi în B_{n-1} la *rotaţia dreapta):*

$B_i \leftarrow B_{i-1}$, $i=1,2,\ldots,n-1$, $B_0 \leftarrow B_{n-1}$ *(stânga),*
$B_{i-1} \leftarrow B_i$, $i=1,2,\ldots,n-1$, $B_{n-1} \leftarrow B_0$ *(dreapta).*

Pentru exemplificare în figura 2.22 se prezintă un registru pe 4 biţi care permite realizarea operaţiilor de ştergere, încărcare şi citire paralelă.

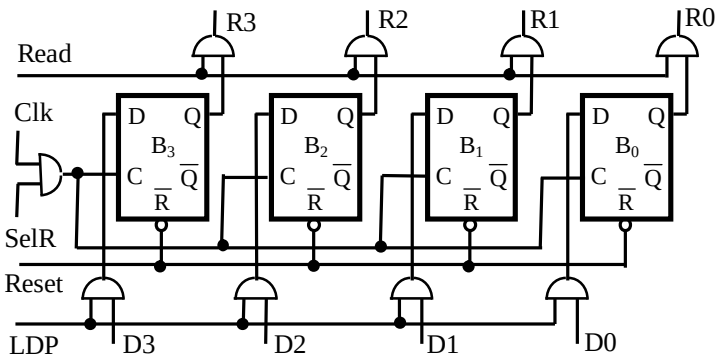


Fig. 2.22. Registru de 4 biti cu încărcare si citire paralelă.

Înscrierea celor 4 bistabile *D* ale registrului se face sincron cu impulsul de tact dacă semnalul de selecţie registru *SelR* este activ. Semnalul *LDP* validează intrările de pe liniile *D3…D0,* iar semnalul *Read* validează ieşirile *R3…R0* ale registrului. Ştergerea registrului (încărcare *CBB* cu *0*, se realizează prin activarea semnalului *Reset.*

În calculatoare *registrele* sunt utilizate la procesarea unor informaţii cum ar fi: *adrese, coduri de instrucţiuni, operanzi, rezultate parţiale sau definitive, informaţii de stare* etc. Una dintre cele mai importante operaţii o constituie transferul între registre. În continuare vor fi prezentate două modalităţi de transfer ilustrate în figurile 2.23 şi 2.24.

Transferul de la un registru sursă *C* la două registre destinaţie *A* şi *B* (figura 2.23) este validat prin activarea simultană a semnalelor *LoadA, LoadB* şi *ReadC.* Selecţia registrelor care se înscriu se realizează prin activarea semnalelor *SelA* şi *SelB.*

În cazul transferului de la două registre sursă *A* şi *B* la un registru destinaţie *C* (figura 2.24), ieşirile registrelor *A* şi *B* sunt reunite în porţile *SAU* de la intrarea registrului destinaţie *C*. Transferul *C-B*, de exemplu se realizează prin activarea semnalelor *ReadB, LoadC şi SelC.*

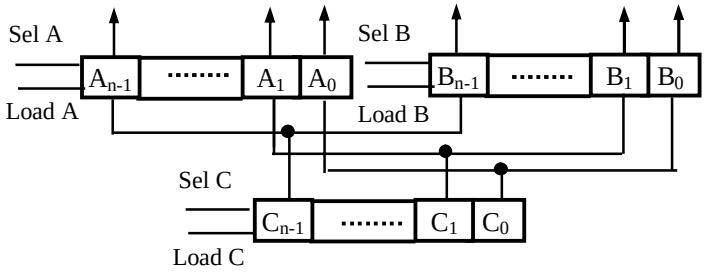


Fig. 2.23. Transfer : un registru sursă – două registre destinaţie.

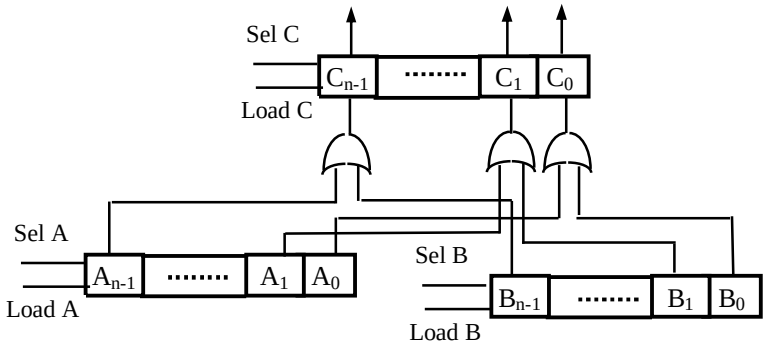


Fig. 2.24. Transfer : două registre sursă – un registru destinaţie.