1. ELEMENTE DE ALGEBRĂ BOOLEANĂ

În teoria *circuitelor numerice* şi în *electronica digitală* în general, semnalele electrice pot lua numai valori discrete, în majoritatea cazurilor aceste valori fiind asociate convențional lui „0” logic şi „1” logic. În limbaj tehnic ne vom referi la aceste două valori cu noțiunea de “bit” (***bi****nary digi****t***).

Bitul se defineşte în teoria informației şi este o unitate de măsură a acesteia, echivalentă cu informația transmisă prin furnizarea unui mesaj din două egal probabile.

Calculatoarele electronice digitale (numerice) efectuează operații logice. De aceea, pentru a studia principiile de operare ale subsistemelor de procesare logică, este necesar să se analizeze unele noțiuni de *logică matematică*. Se disting mai multe direcții de preocupare în logica matematică, printre care *logica claselor* şi *logica propozițiilor*.

În logica claselor se studiază relațiile dintre clasele (mulțimile) de obiecte, prin clasă înțelegându-se totalitatea obiectelor care au o anumită proprietate.

În logica propozițiilor se studiază propozițiile din punct de vedere al adevărului sau falsității lor (este vorba de propoziții matematice).

În afară de logica bivalentă, în care propozițiile pot fi numai adevărate sau numai false, s-au dezvoltat şi alte logici matematice în care se admit şi alte valori pentru propoziții. Aceste logici au căpătat atributul de *polivalente*.

Majoritatea sistemelor digitale lucrează în logică bivalentă, utilizând codificarea binară a informației. Există şi sisteme care lucrează pe baza unor logici polivalente.

Fie A o propoziție. Dacă ea este adevărată vom scrie: A = 1. Dacă este falsă, vom scrie: A = 0. Astfel, 1 şi/sau 0 reprezintă valori de adevăr (sau valori logice binare) pentru propoziția A. Expresiile în care intervin mai multe propoziții vor fi numite *funcții logice*.

Algebra logică binară a fost fundamentată prin lucrările matematicianului englez George Boole şi din această cauză ea mai poartă şi denumirea de *algebră Boole* sau *algebră booleană*. Pentru studiul circuitelor numerice (digitale) se foloseşte ca suport matematic algebra booleană. Ea are la bază o serie de postulate (axiome) şi teoreme.

1.1 Axiome şi teoreme booleene

Algebra booleană operează pe o mulțime B = { x / x∈{0, 1}}. În această mulțime binară se definesc trei legi de compoziție: *complementarea* (negare, “NU”, “NOT”, inversare logică), *disjuncția* (sumă logică, “+”, “SAU”, “OR”, “$\cup$”) şi *conjuncția* (produs logic, “∗”, “ŞI”, “AND”, “$\cap$”), pentru care se dau în continuare tabelele de adevăr, simbolurile grafice şi implementarea prin contacte (figura 1.1)

x	$\bar{x}$
0	1
1	0

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

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

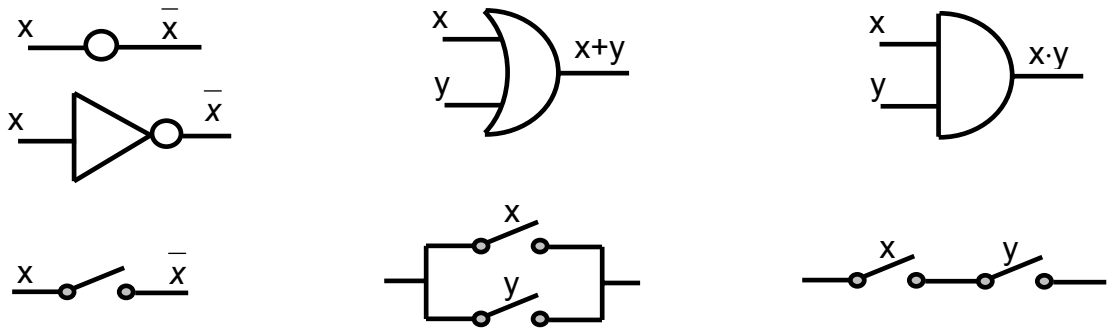


Figura 1.1 Tabelele de adevăr, simbolurile grafice şi implementarea prin contacte electrice pentru complementare, disjuncție şi conjuncție

Toate relațiile definite pe B au un *caracter dual*, adică relațiile rămân valabile dacă se fac schimbările: „ + ” cu „ ∗ ” şi respectiv „0” cu „1” (teorema dualității).

În mulțimea B se poate alege o structură de şase *axiome duale* pe baza cărora se definesc teoremele şi proprietățile care stau la baza algebrei boolene. Acestea sunt prezentate în continuare.

Axiome:

1. Mulțimea B este o mulțime închisă:

$$X,Y \in B \Rightarrow X+Y \in B;\ X,Y \in B \Rightarrow XY \in B ; \quad (1.1)$$

2. Asociativitatea:

$$X+(Y+Z) = (X+Y)+Z ;\ X(Y \cdot Z) = (X \cdot Y) \cdot Z ; \quad (1.2)$$

3. Comutativitatea:

$$X+Y = Y+X ;\ X \cdot Y = Y \cdot X ; \quad (1.3)$$

4. Distributivitatea:

$X+Y\cdot Z = (X+Y)(X+Z)\ ;\ X\cdot (Y+Z) = X\cdot Y+X\cdot Z\ ;$ (1.4)

5. Element neutru:

$X + 0 = 0 + X = X\ ;\ X\cdot 1 = 1\cdot X = X\ ;$ (1.5)

6. Complementul:

$X + \overline{X} = 1;\ X\cdot \overline{X} = 0\ ;$ (1.6)

Teoreme (proprietăți):

7. Idempotența:

$X+X+\ldots..+X = X\ ;\ XX\ldots..X = X\ ;$ (1.7)

8. Elemente neutre:

$X+1 = 1\ ;\ X\cdot 0 = 0\ ;$ (1.8)

9. Involuția:

$\overline{\overline{X}} = X,\ \overline{\overline{X}} = X;$ (1.9)

10. Absorbția:

$X+XY = X\ ;\ X(X+Y) = X\ ;$ (1.10)

11. Relațiile lui De Morgan:

$\overline{X+Y} = \overline{X}\,\overline{Y},\quad \overline{XY} = \overline{X}+\overline{Y}$ (1.11)

În general, notând cu $\sum$ suma şi respectiv cu $\prod$ produsul boolean, relațiile *De Morgan* se scriu:

$$\overline{\sum_{k=1}^{n} x_k} = \prod_{k=1}^{n} \overline{x}_k \quad \text{şi} \quad \overline{\prod_{k=1}^{n} x_k} = \sum_{k=1}^{n} \overline{x}_k \qquad (1.12)$$

Pe mulțimea B sunt valabile teoremele enunțate. Demonstrația lor se poate face folosind axiomele, dar este mai comod dacă se folosesc *tabelele de adevăr*. Tabela de adevăr stabileşte o corespondență între valorile de adevăr ale variabilelor şi valoarea de adevăr a funcției.

Exemplu:

x	y	$x+y$	$\overline{x+y}$	$\overline{x}$	$\overline{y}$	$\overline{x}\cdot\overline{y}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Figura 1.2 Relațiile lui De Morgan

Perechile de operatori NOT şi AND, respectiv NOT şi OR formează fiecare câte un *sistem complet*, adică orice relație definită pe B poate fi exprimată folosind numai operatorii unei singure perechi.

Circuitul fizic care implementează un operator logic se numeşte poartă logică. Sistemele complete prezentate au fost realizate cu câte o singură poartă: ŞI-NU (NAND, Scheffer) şi SAU-NU (NOR, funcție „nici" sau funcție Pierce). Un sistem complet de operatori poate exprima orice relație logică ca în exemplul următor, în care ne propunem să implementăm operatorii NOT, OR şi AND folosind operatori NAND şi NOR.

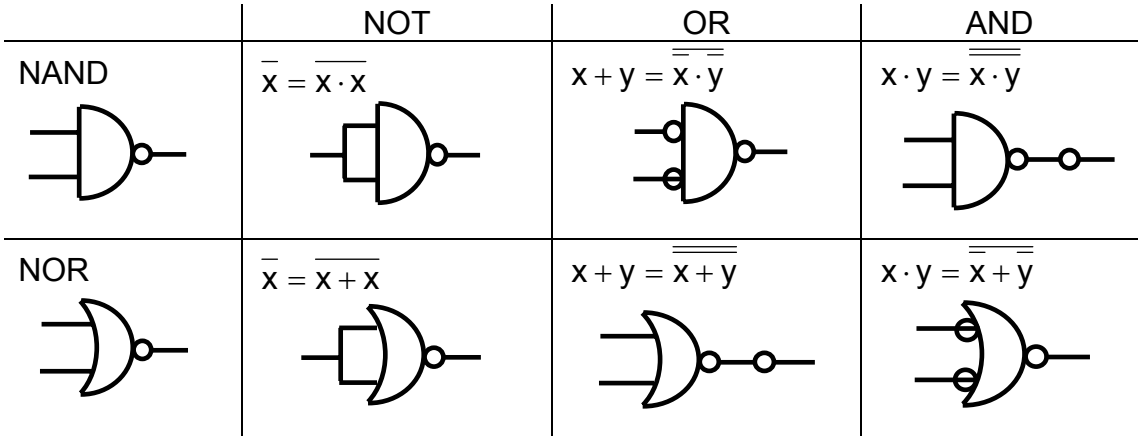


Figura 1.3 Implementarea operatorilor NOT, OR şi AND folosind operatori NAND şi NOR.

Relații booleene utile:

1) $x + 1 = (x + 1) * 1 = (x + 1)(x + \bar{x}) = x * x + x * \bar{x} + 1 * x + 1 * \bar{x} =$
$= x + 0 + x + \bar{x} = x + \bar{x} = 1$ (1.13)

2) proprietatea de absorbție: $x + (x * y) = x * (1 + y) = x$
şi relația dual: $x * (x + y) = x * x + x * y = x + xy = x$ (1.14)

3) $x + \bar{x} * y = x + y$ (1.15)

Demonstrație:

$$x + y = (x + y)(x + \bar{x})(y + \bar{y}) = (xx + x\bar{x} + yx + y\bar{x})(y + \bar{y}) = (x + xy + \bar{x}y)(y + \bar{y}) =$$
$$= xy + x\bar{y} + xy + xy\bar{y} + \bar{x}y + \bar{x}y\bar{y} = xy + x\bar{y} + \bar{x}y = x(y + \bar{y}) + \bar{x}y = x + \bar{x}y$$

(1.16)

4) $x(\bar{x} + y) = x * y$ (duala relației precedente). (1.17)

1.2 Funcții logice

O funcție $f : B^n \rightarrow B$ se numeşte *funcție booleană*. Altfel spus, o funcție booleană de *n* variabile $y = f(x_1, x_2, ..x_n)$, unde x_i sunt variabile de intrare, se caracterizează prin faptul că atât funcția cât şi variabilele nu pot lua decât două valorile distincte, 0 şi 1.

Exemplu:
Considerăm un rezervor alimentat de trei robinete x, y şi z. Ne propunem să menținem rezervorul plin cu ajutorul acestor trei robinete. Rezervorul poate fi menținut plin dacă cel puțin două robinete sunt deschise simultan. Dacă considerăm că un robinet deschis are atribuită valoarea logică 1, atunci funcția care descrie din punct de vedere logic această situație este următoarea:

$$U(x,y,z) = xy\bar{z} + x\bar{y}z + \bar{x}yz + xyz \qquad (1.18)$$

1.3 Reprezentarea funcțiilor logice

Pentru reprezentarea funcțiilor logice se folosesc în mod curent şi în principal trei metode, descrise mai jos.

A. Reprezentare prin tabela de adevăr

Această reprezentare presupune marcarea, într-un tabel, a corespondenței dintre valorile de adevăr ale variabilelor de intrare şi valoarea de adevăr a funcției în fiecare punct al domeniului de definiție.

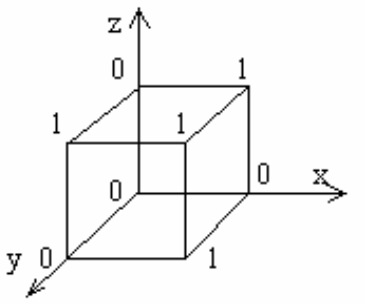
Exemplu:
Pentru cazul problemei considerate în exemplul anterior, reprezentarea prin tabel arată ca în figura 1.4

x	y	z	U
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Figura 1.4 Exemplu de reprezentare prin tabel a unei funcții logice

B. Reprezentarea prin diagrame Karnaugh

Reprezentarea prin diagrame *Karnaugh* constă în a marca punctele domeniului de definiție într-o diagramă plană şi a preciza valoarea funcției în fiecare din aceste puncte.



x \ yz	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Figura 1.5 Reprezentarea funcțiilor logice prin diagrame Karnaugh

Dacă luăm în considerare vârful cubului caracterizat prin coordonatele 000, constatăm că acest vârf este vecin cu vârfurile 001, 010, 100. În diagrama *Karnaugh* constatăm că 000 este vecin doar cu 001 şi 100. Pentru ca diagrama *Karnaugh* să fie echivalentă cu reprezentarea prin cub, ea trebuie să păstreze acelaşi vecinătăți, lucru ce devine posibil doar dacă ne imaginăm latura din stânga a diagramei *Karnaugh* în continuarea celei din dreapta, iar latura de sus în continuarea celei de jos. În acest fel, punctul 000 devine vecin şi cu punctul 010.

C. Reprezentarea prin echivalenți zecimali ai mintermilor

Reprezentarea prin *echivalenți zecimali ai mintermilor* constă în indicarea echivalenților zecimali ai conjuncțiilor pentru care valoarea funcției este 1 sau a echivalenților zecimali corespunzători valorii 0 ale funcției.

Exemplu:

$$U(x,y,z) = R_1 (3,5,6,7) \quad (1.19)$$

$$U(x,y,z) = R_0 (0,1,2,4) \quad (1.20)$$

1.4 Expresii analitice ale funcțiilor logice

În majoritatea aplicațiilor practice este necesară utilizarea formei analitice a funcțiilor booleene. În acest scop se utilizează două forme de dezvoltare:
- *forma canonică disjunctivă* (FCD) care presupune utilizarea unor funcții elementare numite *constituenți ai unității* (*termeni minimali* sau *mintermi*);
- *forma canonică conjunctivă* (FCC) care presupune utilizarea unor funcții elementare numite *constituenți ai lui zero* (*termeni maximali* sau *maxtermi*).

Notație: $$x^{\sigma} = \begin{cases} x, \text{ pentru } \sigma = 1 \\ \overline{x}, \text{ pentru } \sigma = 0 \end{cases} \quad (1.21)$$

Definiție:

Se numeşte *constituent al unității* funcția elementară $Q_k^{(n)}$ caracterizată prin faptul că ia valoarea 1 logic într-un un singur punct al domeniului de definiție. Constituentul unității va fi produsul logic al tuturor variabilelor negate sau nenegate:

$$Q_k^{(n)} = \prod_{i=0}^{n-1} x_i^{\sigma_i}, \quad k_{(10)} = \sigma_{n-1},\ldots..\sigma_{0\,(2)} \quad (1.22)$$

Pentru ca $Q_k^{(n)}$ să fie 1 într-un anumit punct al domeniului de definiție este necesar ca toți termenii produsului să fie 1 logic, ceea ce presupune ca argumentele să aibă valoarea $x_i = \sigma_i$.

Aşadar rezultă următoarea regulă de scriere a *mintermenilor* $Q_k^{(n)}$: în conjuncția variabilelor, variabilele care iau valoarea 0 în punctul respectiv al domeniului de definiție se vor lua negate, iar cele care iau valoarea 1 se vor lua nenegate.

Numim *conjuncții vecine* două conjuncții care sunt constituite din aceleaşi variabile şi diferă doar prin comlementarea uneia singure. Prin sumarea a două conjuncții vecine se obține o conjuncție cu un număr de variabile mai mic cu 1, lipsind variabila a cărei complementaritate diferă.

Exemplu:
Pentru cazul unei funcții de 4 variabile, fie suma a două conjuncții vecine:

$$Q_9^{(4)} + Q_8^{(4)} = x_3\overline{x_2}\,\overline{x_1}x_0 + x_3\overline{x_2}\,\overline{x_1}\,\overline{x_0} = x_3\overline{x_2}\,\overline{x_1}(x_0 + \overline{x_0}) = x_3\overline{x_2}\,\overline{x_1} = Q_4^{(3)} \qquad (1.23)$$

Definiție:
Se numeşte *constituent al lui zero* funcția elementară $D_k^{(n)}$ care ia valoarea 0 logic într-un singur punct al domeniului de definiție. Constituentul lui 0 va fi suma logică a tuturor variabilelor negate sau nenegate:

$$D_k^{(n)} = \sum_{i=0}^{n-1} \overline{x_i^{\sigma_i}} \qquad k_{(10)} = \overline{\sigma_{n-1}}.....\overline{\sigma_{0\,(2)}} \qquad (1.24)$$

Pentru ca $D_k^{(n)}$ să fie 0 într-un un anumit punct al domeniului de definiție este necesar ca toți termenii sumei să fie 0, ceea ce este echivalent cu $x_i = \sigma_i$.

Prin urmare rezultă următoarea regulă de scriere a *maxtermului* $D_k^{(n)}$: în disjuncția variabilelor, variabilele care iau valoarea 0 în punctul respectiv al domeniului de definiție se vor lua nenegate, iar cele care iau valoarea 1 se vor lua negate.

Disjuncțiile vecine se definesc în mod similar cu conjuncțiile vecine. Prin înmulțirea a două disjuncții vecine se obține o disjuncție având o variabilă mai puțin (dispare acea variabilă care îşi modifică complementaritatea).

$$\begin{aligned} D_9^{(4)} \cdot D_8^{(4)} &= (\overline{x_3} + x_2 + x_1 + \overline{x_0})(\overline{x_3} + x_2 + x_1 + x_0) = (D_4^{(3)} + \overline{x_0})(D_4^{(3)} + x_0) = \\ &= D_4^{(3)}D_4^{(3)} + D_4^{(3)}(\overline{x_0} + x_0) + \overline{x_0}x_0 = D_4^{(3)} + D_4^{(3)} + 0 = D_4^{(3)} = \overline{x_3} + x_2 + x_1 \end{aligned} \qquad (1.25)$$

Pentru o funcție de o variabilă x, f(x) se poate scrie sub forma:

$$f(x) = xf(1) + \overline{x}f(0) \qquad (1.26)$$

În mod similar, pentru o funcție de două variabile $f(x_1, x_0)$ avem relația:

$$\begin{aligned} f(x_1,x_0) &= x_1 f(1,x_0) + \overline{x_1} f(0,x_0) = x_1[x_0 f(1,1) + \overline{x_0} f(1,0)] + \overline{x_1}[x_0 f(0,1) + \overline{x_0} f(0,0)] = \\ &= x_1\overline{x_0} f(1,1) + x_1\overline{x_0} f(1,0) + \overline{x_1}x_0 f(0,1) + \overline{x_1}\,\overline{x_0} f(0,0) = \sum_{\sigma_1,\sigma_0 \in \{0,1\}} x_1^{\sigma_1} x_0^{\sigma_0} f(\sigma_1\sigma_0) \end{aligned} \qquad (1.27)$$

Prin inducție rezultă:

$$f(x_{n-1},...,x_0) = \sum_{\sigma_j \in \{0,1\}} \left(\prod_{j=0}^{n-1} x_j^{\sigma_j}\right) \cdot f(\sigma_{n-1},...,\sigma_0), \quad j = 0,...,n-1 \qquad (1.28)$$

Termenii de sumă pentru care $f(\sigma_{n-1},...,\sigma_0)=0$ dispar, deci:

$$f(x_{n-1},...,x_0)=\sum_{f(\sigma_{n-1},...,\sigma_0)=1}(\prod_{j=0}^{n-1}x_j^{\sigma_j}) \quad (\text{FCD}) \tag{1.29}$$

Expresia unei funcții logice este suma *mintermilor* pentru care funcția ia valoarea 1.

Exemplu:

$$U(x,y,z)=x^1y^1z^0+x^1y^0z^1+x^0y^1z^1+x^1y^1z^1=xy\bar{z}+x\bar{y}z+\bar{x}yz+xyz \tag{1.30}$$

Forma canonică conjunctivă (FCC) se obține astfel:

$$f(x_{n-1},...,x_0)=\bar{\bar{f}}(x_{n-1},...,x_0)=\overline{\overline{\sum_{f(\sigma_{n-1},...,\sigma_0)=1}(\prod_{j=0}^{n-1}x_j^{\sigma_j})}}=\overline{\sum_{\bar{f}(\sigma_{n-1},...,\sigma_0)=1}(\prod_{j=0}^{n-1}x_j^{\sigma_j})}=$$
$$=\overline{\sum_{f(\sigma_{n-1},...,\sigma_0)=0}(\prod_{j=0}^{n-1}x_j^{\sigma_j})}=\prod_{f(\sigma_{n-1},...,\sigma_0)=0}(\sum_{j=0}^{n-1}\overline{x_j^{\sigma_j}}) \tag{1.31}$$

Expresia funcției logice poate fi scrisă deci ca produsul *maxtermilor* pentru care funcția ia valoarea 0.

Exemplu:

$$U(x,y,z)=(\overline{x^0}+\overline{y^0}+\overline{z^0})(\overline{x^0}+\overline{y^0}+\overline{z^1})(\overline{x^0}+\overline{y^1}+\overline{z^0})(\overline{x^1}+\overline{y^0}+\overline{z^0})=$$
$$=(x+y+z)(x+y+\bar{z})(x+\bar{y}+z)(\bar{x}+y+z) \tag{1.32}$$

1.5 Implementarea funcțiilor logice

Implementarea unei funcții logice înseamnă realizarea ei cu ajutorul circuitelor (porților) fundamentale.

Se defineşte *cost al unei implementări* numărul de intrări în circuitele fundamentale care realizează funcția dată.

Pentru implementarea unei funcții cu circuite NAND se porneşte de la FCD:

$$f(x_{n-1},...,x_0)=\sum_{f(\sigma_{n-1},...,\sigma_0)=1}Q_k^{(n)}=\overline{\prod_{f(\sigma_{n-1},...,\sigma_0)=1}\overline{Q_k^{(n)}}},\quad Q_k^{(n)}=\prod_{j=0}^{n-1}x_j^{\sigma_j},k_{10}=\sigma_{n-1},...,\sigma_{0\,(2)} \tag{1.32}$$

Realizând $\overline{Q_k^{(n)}}$ cu circuite NAND, funcția f se obține prin cuplarea ieşirilor circuitelor NAND precedente la intrările unui alt circuit NAND.

Exemplu:

$f(x,y,z)=xy\bar{z}+x\bar{y}z+\bar{x}yz+xyz$ (problema implementată de realația1.18)

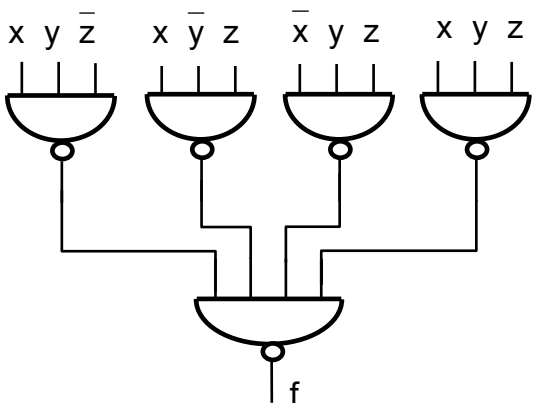


Figura 1.6 Exemplu de implementare a unei funcții logice cu circuite NAND, pornind de la FCD

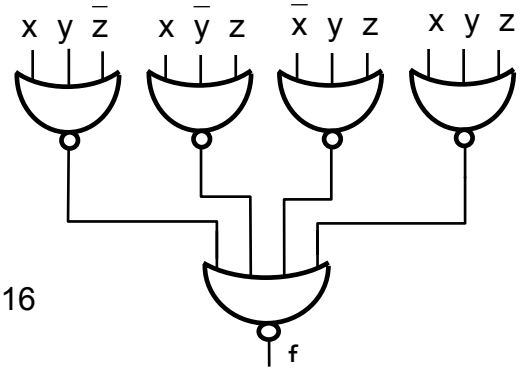
Costul implementării din figura 1.6 este: $C_1(f) = 3 \cdot 4 + 4 = 16$

Pentru implementare cu circuite NOR se porneşte de la FCC:

$$f(x_{n-1},...,x_0) = \prod_{f(\sigma_{n-1},...,\sigma_0)=0} D_k^{(n)} = \overline{\sum_{f(\sigma_{n-1},...,\sigma_0)=0} \overline{D_k^{(n)}}}, \quad D_k^{(n)} = \sum_{j=0}^{n-1} \overline{x_j^{\sigma_j}}, k_{(10)} = \sigma_{n-1},...,\sigma_0 \tag{1.33}$$

Funcția f se obține prin cuplarea ieşirilor circuitelor NOR ce implementează $\overline{D_k^{(n)}}$ la intrările unui alt circuit NOR.

Exemplu:



$C_2(f) = 3 \cdot 4 + 4 = 16$

Figura 1.7 Exemplu de implementare a unei funcții logice cu circuite NOR, pornind de la FCC

Nivelul unei implementări logice se defineşte ca fiind numărul maxim de circuite pe care le străbate un semnal de la intrare către ieşire.

În cazurile precedente s-au considerat structuri logice cu două nivele.

Exemplu:

Fie $f(x,y,z) = xy + yz + xz = \overline{\overline{xy}\,\overline{yz}\,\overline{xz}}$ (1.34)

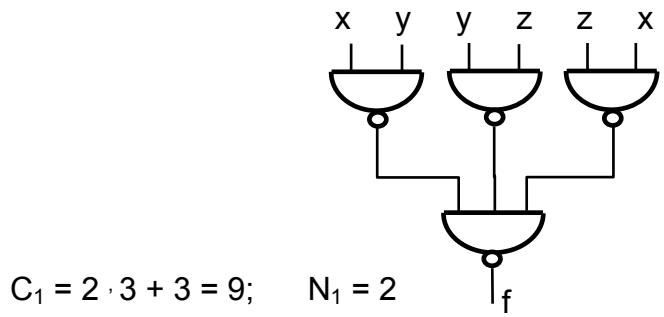


Figura 1.8 Exemplu de implementare a unei funcții logice(1.34) cu NAND, pe două nivele, fără reducerea costului

Rescriind relația (1.34) sub altă formă se obține (1.35).

$$f(x,y,z) = xy + z(x+y) = \overline{\overline{xyz}\,\overline{x+y}} = \overline{\overline{xyz}}\,\overline{\overline{xy}} \qquad (1.35)$$

Scăderea costului în varianta a doua s-a făcut pe seama creşterii nivelului, ceea ce implică însă micşorarea vitezei.

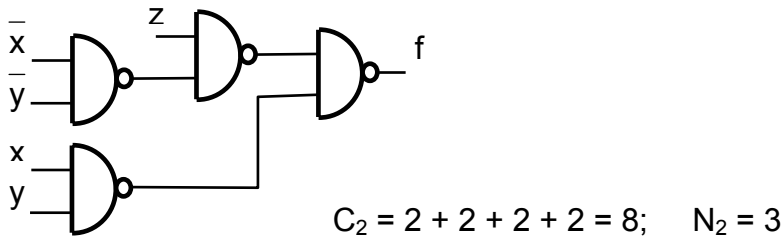


Figura 1.9 Exemplu de implementare a unei funcții logice cu NAND (1.34), pe trei nivele, cu reducerea costului

1.6 Funcții incomplet definite

În unele cazuri, pentru anumite combinații de variabile de intrare nu este precizată valoarea funcției sau aceste combinații nu apar niciodată în sistemul fizic ce materializează funcția. Astfel de funcții se numesc *funcții incomplet definite* şi prezintă valori indiferente, pe care în tabelul de adevăr le vom nota cu x:

x	y	z	f
0	0	0	0
0	0	0	1
0	1	0	1
0	1	1	x
1	0	0	x
1	0	1	x
1	1	0	x
1	1	1	x

Un alt mod de reprezentare a acestor funcții este prin echivalenți zecimali (1.36).

$$f(x,y,z) = R_1(1,2,4) + R_x(3,5,6,7) \quad (1.36)$$
$$f(x,y,z) = R_0(0) + R_x(3,5,6,7)$$
$$f(x,y,z) = R_0(0) + R_1(1,2,4)$$

1.7 Minimizarea funcțiilor logice

În proiectarea sistemelor digitale, analiza şi sinteza circuitelor numerice se bazează pe *algebra booleană*. Rezultă o legătură firească între gradul de complexitate al circuitului care se obține şi gradul de complexitate al funcției care îl descrie. Din acest motiv, pentru sinteza circuitelor numerice (circuite funcționând în regim de comutație), după etapa de definire a funcției, urmează obligatoriu etapa de minimizare a funcției în scopul obținerii unei forme simplificate (formă minimă).

Minimizarea unei funcții este procedeul prin care, pentru un nivel dat, se obține o expresie care generează un cost minim pentru un număr dat de nivele logice.

Implementarea practică a circuitului se realizează pe baza formei minimizate, ceea ce conduce la configurația optimă de circuit.

Există mai multe metode de minimizare, câteva dintre acestea fiind:

- metoda analitică
- metoda Veitch - Karnaugh
- metoda Quine - McCluskey

1.7.1 Metoda analitică

Metoda analitica se bazează pe simplificarea expresiei unei funcții pe baza axiomelor şi teoremelor algebrei booleene.

Exemplu:

$$f(x,y,z) = xy\bar{z} + x\bar{y}z + \bar{x}yz + xyz \quad ; \quad C_1 = 3\cdot 4 + 4 = 16 \quad (1.37)$$

Prelucrând forma dată a funcției, ea se poate rescrie:

$$f(x,y,z) = xy\bar{z} + x\bar{y}z + \bar{x}yz + xyz + xyz + xyz = \\ = xy(\bar{z}+z) + xz(\bar{y}+y) + yz(\bar{x}+x) = xy + xz + yz \quad ; \quad (1.38)$$

$C_2 = 3\cdot 2 + 3 = 9$

1.7.2 Metoda Veitch - Karnaugh

Această metodă transpune axiomele şi teoremele algebrei booleene pe reprezentarea funcției cu diagrame *Karnaugh*.

O diagramă *Karnaugh* poate fi privită ca o reprezentare a funcției booleene, dacă se au în vedere produsele logice ale coordonatelor, prin *mintermi*, aşa cum se observă în reprezentarea care urmează.

$x_2 \setminus x_1x_0$	00	01	11	10
0	$\bar{x}_2\bar{x}_1\bar{x}_0$	$\bar{x}_2\bar{x}_1x_0$	$\bar{x}_2x_1x_0$	$\bar{x}_2x_1\bar{x}_0$
1	$x_2\bar{x}_1\bar{x}_0$	$x_2\bar{x}_1x_0$	$x_2x_1x_0$	$x_2x_1\bar{x}_0$

Fiecare celulă din diagramă conține un *minterm*. Două celule vecine conțin *mintermi* care diferă prin valoarea unei singure variabile. Prin adunarea *mintermilor* din două celule vecine se elimină variabila care îşi schimbă valoarea. Aceasta permite simplificarea expresiei funcției care se obține şi implicit simplificarea structurii logice corespunzătoare

Exemplu:

$x_2 \setminus x_1x_0$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

FCD se obține prin sumarea *mintermilor* pentru care funcția ia valoarea *1*. Prin gruparea celulelor vecine pentru care valoarea funcției este 1 se obțin x_2x_1, x_2x_0, x_1x_0 (prin eliminarea variabilelor care îşi schimbă valoarea în cadrul aceleiaşi grupări).

Fiecare celulă ocupată de valoarea 1 trebuie să facă parte din cel puțin o grupare, dar poate fi inclusă în mai multe grupări.

Pentru exemplul considerat se obține FMD:

$$f(x_2, x_1, x_0) = x_2x_1 + x_2x_0 + x_1x_0. \quad (1.39)$$

Dacă un grup de două celule vecine este vecin la rândul său cu un alt grup de două celule vecine, acestea se pot contopi într-un singur grup de patru celule vecine, ceea ce va permite eliminarea a două variabile. În general, *un grup de 2^m celule vecine ocupate de unități permite eliminarea a m variabile*.

Exemplu:

$$f(x_3, x_2, x_1, x_0) = \bar{x}_1\bar{x}_0 + \bar{x}_2\bar{x}_0 \quad (1.40)$$

$x_3x_2 \setminus x_1x_0$	00	01	11	10
00	1	0	0	1
01	1	0	0	0
11	1	0	0	0
10	1	0	0	1

Cel mai avansat grad de simplificare se obține dacă valorile 1 dintr-o diagramă Karnaugh sunt grupate într-un număr minim de grupuri, fiecare grup conținând un număr maxim de unități, aşa cum este exemplificat în diagrama care urmează.

$x_4x_3 \setminus x_2x_1x_0$	000	001	011	010	011	111	101	100
00		1	1		1		1	1
01		1	1			1	1	1
11		1		1		1	1	
10		1		1	1		1	

Exemplu:

$$f(x_4,x_3,x_2,x_1,x_0)=\overline{x_1}x_0+\overline{x_4}\,\overline{x_2}x_0+x_4\overline{x_2}x_1\overline{x_0}+\overline{x_3}x_2x_1\overline{x_0}+ \\ +x_3x_2x_0+\overline{x_4}x_2\overline{x_1} \quad \text{(FMD)} \qquad (1.41)$$

Pentru simplitate, în diagramă nu s-au trecut decât valorile 1 ale funcției. Costul este:

$C_1 = (2 + 3 + 4 + 4 + 3 + 3) + 6 = 25$

Implementarea cu circuite NAND este prezentată în figura 1.10.

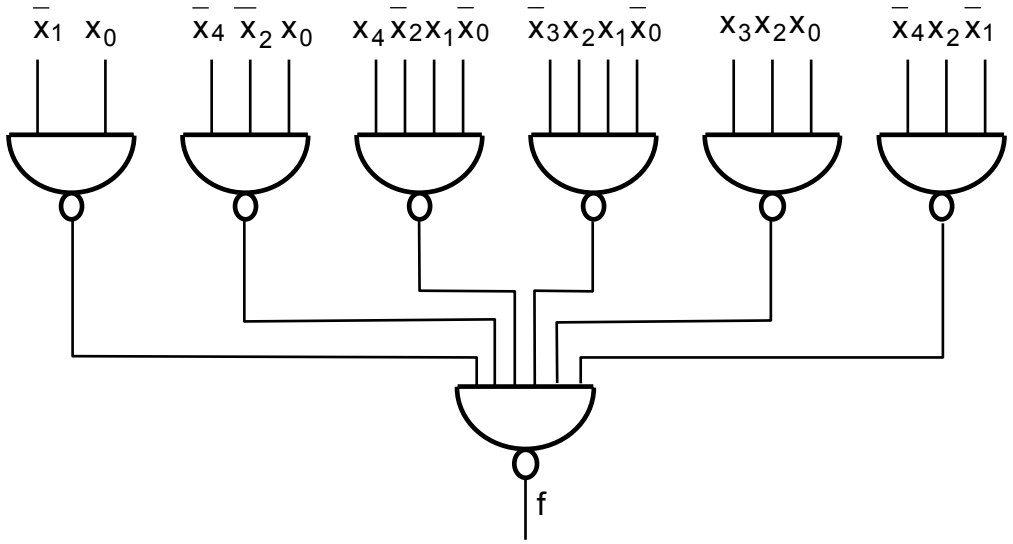


Figura 1.10 Implementarea cu circuite NAND a funcției (1.41)

Pentru minimizarea funcțiilor scrise sub forma conjunctivă, în diagrama *Karnaugh* se vor considera disjuncțiile corespunzătoare valorilor 0 ale funcției şi se va urma o procedură asemănătoare cu cea folosită la forma disjunctivă. Metoda constă în cuplarea de disjuncții vecine din care va dispărea termenul corespunzător bitului ce se modifică, în echivalenții binari.

Exemplu:

x_4x_3 \ $x_2x_1x_0$	000	001	011	010	110	111	101	100
00	0			0		0		
01	0			0	0			
11	0		0		0			0
10	0		0			0		0

$$f(x_4,x_3,x_2,x_1,x_0)=(x_4+x_2+x_0)\cdot(\overline{x_4}+x_1+x_0)\cdot(\overline{x_4}+x_2+\overline{x_1}+\overline{x_0})\cdot \\ \cdot(\overline{x_3}+\overline{x_2}+\overline{x_1}+x_0)\cdot(x_3+\overline{x_2}+\overline{x_1}+\overline{x_0}) \qquad (1.42)$$

$C_2 = (3 + 3 + 4 + 4 + 4) + 5 = 23$

Implementarea cu circuite NOR este prezentată în figura 1.11.

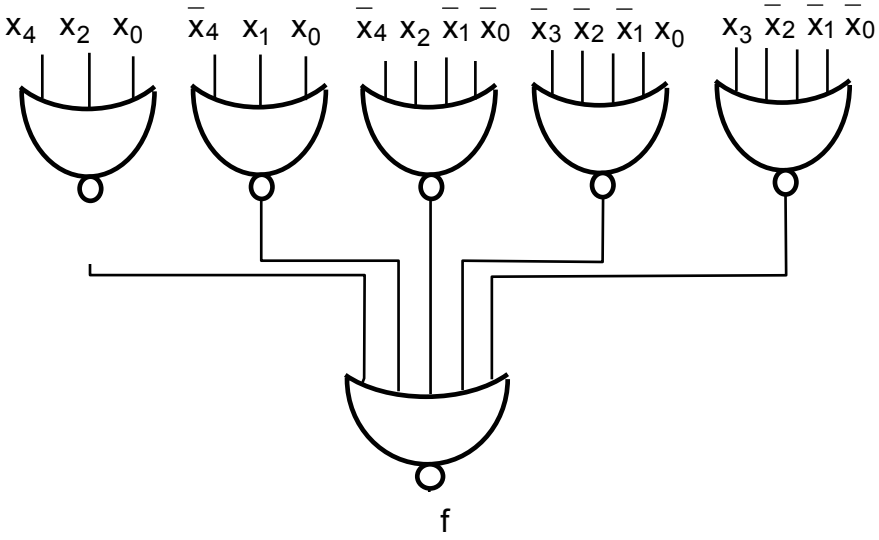


Figura 1.11 Implementarea cu circuite NOR a funcției (1.42)

În cazul *funcțiilor incomplet definite*, valorile indiferente ale funcției se iau 1 pentru forma disjunctivă şi 0 pentru forma conjunctivă dacă aceste valori participă la minimizare. Valorile indiferente care nu sunt cuplate devin 0 pentru forma disjunctivă şi 1 pentru forma conjunctivă. Considerarea valorilor indiferente determină simplificarea formei funcției care se obține în sensul reducerii numărului de variabile.

Exemplu:

$$f(x_3, x_2, x_1, x_0) = R_1(0, 1, 2, 4) + R_x(3, 5, 10, 15) \qquad (1.43)$$

$x_3x_2\backslash x_1x_0$	00	01	11	10
00	1	1	x	1
01	1	x		
11			x	
10				x

$x_3x_2\backslash x_1x_0$	00	01	11	10
00			x	
01		x	0	0
11	0	0	x	0
10	0	0	0	x

$$f(x_3,x_2,x_1,x_0) = \overline{x_3}\,\overline{x_1}\,\overline{x_0} + \overline{x_3}\,\overline{x_2}\,\overline{x_2} + \overline{x_3}\,\overline{x_2}\,\overline{x_1}; \qquad C_1 = 3\cdot 3 + 3 = 12 \qquad (1.44)$$

$$f(x_3,x_2,x_1,x_0) = (\overline{x_3} + x_1)(\overline{x_3} + x_2 + \overline{x_0})(x_3 + \overline{x_2} + \overline{x_1})(\overline{x_2} + \overline{x_1} + x_0); \qquad (1.45)$$

$$C_3 = (2 + 3\cdot 3) + 4 = 15$$

$x_3x_2\backslash x_1x_0$	00	01	11	10
00	1	1	x	1
01	1	x		
11			x	
10				x

$x_3x_2\backslash x_1x_0$	00	01	11	10
00			x	
01		x	0	0
11	0	0	x	0
10	0	0	0	x

$$f(x_3,x_2,x_1,x_0) = \overline{x_3}\,\overline{x_1} + \overline{x_3}\,\overline{x_2}; \qquad C_2 = 2\cdot 2 + 2 = 6 \qquad (1.46)$$

$$f(x_3,x_2,x_1,x_0) = \overline{x_3}(\overline{x_2} + \overline{x_1}); \qquad C_4 = 2 + 2 = 4 \qquad (1.47)$$

Concluzia este că *prin participarea valorilor indiferente la minimizarea funcțiilor incomplet definite se obține o reducere a costurilor*.

1.7.3 Metoda Quine – McCluskey

Pentru funcții ce depind de mai mult de 5 variabile, metoda Veitch - Karnaugh devine greoaie şi se preferă o altă metodă, metoda Quine - McCluskey.

În cazul formei disjunctive, minimizarea prin această metodă presupune parcurgerea etapelor prezentate în continuare.

1) *Ordonarea echivalenților binari* ai conjuncțiilor corespunzătoare valorilor 1 ale funcției după pondere.

Definiție:

Ponderea conjuncției $Q_k^{(n)} = \prod_{j=0}^{n-1} x_j^{\sigma_j}$ este numărul $P[Q_k^{(n)}] = \sum_{j=0}^{n-1} \sigma_j$, unde $\sum$ este suma algebrică.

Exemplu:

$$P[\overline{x_3}x_2x_1\overline{x_0}] = P[x_3^0x_2^1x_1^1x_0^0] = 0+1+1+0 = 2 \qquad (1.48)$$

Pentru o conjuncție $Q = Q_1Q_2$, ponderea este $P[Q]= P[Q_1]+ P[Q_2]$.

Lemă:

Pentru două conjuncții vecine ponderile diferă cu o unitate.

$$P[x_iQ] = P[x_i] + P[Q] = 1 + P[Q]$$
$$P[\overline{x_i}Q] = P[\overline{x_i}] + P[Q] = 0 + P[Q] = P[Q]$$

Reciproca nu este adevărată: $P[\overline{x_3}x_2x_1\overline{x_0}) = P[x_3\overline{x_2}x_1x_0] + 1$

2) *Determinarea implicanților primi* prin comparații succesive ale echivalenților binari.

Definiție:

Se numeşte *implicant prim al unei funcții* un termen al acesteia care nu se mai poate reduce.

Pentru determinarea *implicanților primi* se cuplează echivalenții binari care diferă doar printr-o cifră din acelaşi rang. Se obține astfel primul tabel de comparații în care dispariția variabilei corespunzătoare cifrei care se modifică se notează cu „-”. În continuare, se pot cupla două conjuncții din grupe vecine dacă simbolul „-” se află în acelaşi rang şi echivalenții binari diferă doar printr-o cifră din acelaşi rang. Rezultă al doilea tabel de comparare şi procedura se repetă. Conjuncția care nu se mai poate cupla cu nici o altă conjuncție din tabel este un *implicant prim al funcției date*.

3) *Determinarea tabelului de acoperire al funcției*

Tabelul de acoperire este un tablou rectangular, la care liniile corespund *implicanților primi*, iar coloanele corespund *echivalenților zecimali* ai conjuncțiilor pentru care funcția ia valoarea 1. Tabloul se completează cu 1 în pozițiile pentru care conjuncțiile de pe coloane realizează implicanții primi de pe linii.

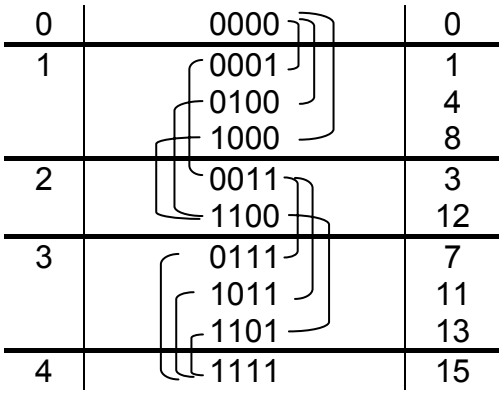
4) Calculul formal de determinare a tuturor soluțiilor funcției

Fiecărui implicant prim X i se ataşează o variabilă logică F_x care ia valoarea 1 când implicantul prim este realizat (conform tabelului de acoperire). Pentru realizarea funcției este necesar ca în expresia ei să existe toate conjuncțiile corespunzătoare valorilor 1 ale funcției. Pentru determinarea tuturor soluțiilor funcției, se exprimă această cerință cu ajutorul variabilelor F_x.

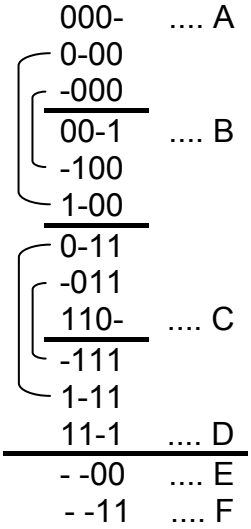
Exemplu:

$$f (x_3, x_2, x_1, x_0) = R_1 (0, 1, 3, 4, 7, 8, 11, 12, 13, 15) \qquad (1.49)$$

1) Ordonarea echivalenților binari



2) Determinarea implicanților primi



$$A = \bar{x}_3 \bar{x}_2 \bar{x}_1 ; \quad B = \bar{x}_3 \bar{x}_2 x_0 ; \quad C = x_3 x_2 \bar{x}_1 ;$$
$$D = x_3 x_2 x_0 ; \quad E = \bar{x}_1 \bar{x}_0 ; \quad F = x_1 x_0 \tag{1.50}$$

3) Determinarea tabelului de acoperire al funcției

	0	1	3	4	7	8	11	12	13	15
A	1	1								
B		1	1							
C								1	1	
D									1	1
E	1			1		1		1		
F			1		1		1			1

4) Calculul formal de determinare a tuturor soluțiilor funcției

$$(F_A + F_E)(F_A + F_B)(F_B + F_F) F_E F_F F_E F_F (F_C + F_E)(F_C + F_D)(F_D + F_F) = 1$$
$$(F_A + F_B) F_E F_F (F_C + F_D) = 1 \tag{1.51}$$
$$F_A F_C F_E F_F + F_A F_D F_E F_F + F_B F_C F_E F_F + F_B F_D F_E F_F = 1$$

Funcția *f* poate avea 4 expresii:

$$f = A + C + E + F \tag{1.52}$$
$$f = A + D + E + F$$
$$f = B + C + E + F$$
$$f = B + D + E + F$$

În prima variantă de obține

$$f(x_3, x_2, x_1, x_0) = \overline{x_3} \cdot \overline{x_2} \cdot \overline{x_1} + x_3 \cdot x_2 \cdot \overline{x_1} + \overline{x_1} \cdot \overline{x_0} + x_1 \cdot x_0 \tag{1.53}$$

Implementarea cu circuite NAND este prezentată în figura 1.12.

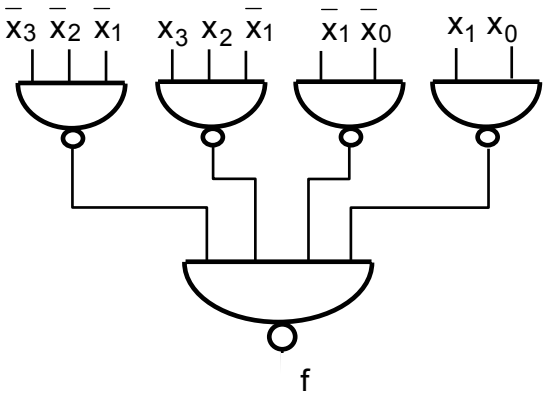


Figura 1.12 Implementarea cu circuite NAND a funcției (1.53)

În cazul formei conjunctive a funcțiilor, procedura este similară, dar se vor considera valorile 0 ale funcției şi disjuncțiile corespunzătoare.

Metoda Quine – McCluskey se pretează implementării automate a sistemelor numerice. Algoritmul bazat pe această metodă poate fi transpus în aplicații software care determină automat structura logică a circuitului.