CIRCUITE COMBINAȚIONALE UZUALE

1. Scopul lucrării

Lucrarea prezintă unele circuite combinaționale uzuale şi utilizarea acestor circuite la implementarea circuitelor digitale mai complexe. Circuitele combinaționale prezentate sunt: convertoare de cod, decodificatoare, codificatoare, multiplexoare, demultiplexoare, memorii ROM şi rețele logice programabile.

2. Considerații teoretice

2.1. Convertoare de cod

Convertoarele de cod au, în cazul general, *n* intrări şi *m* ieşiri, şi se utilizează pentru transformarea informației din codul cu *n* biți în codul cu *m* biți.

Pentru proiectarea unui convertor de cod se poate utiliza tabelul de corespondențe între cuvintele binare ale celor două coduri. Fiecare poziție din codul sursă se notează cu o variabilă, totalitatea acestora reprezentând intrările circuitului combinațional. Fiecare poziție din codul destinație se notează cu o variabilă, totalitatea acestora reprezentând ieşirile circuitului. Tabelul de corespondențe se transformă astfel în tabel de adevăr pentru funcțiile realizate de circuit, care arată dependența variabilelor de ieşire de cele de intrare.

Se consideră conversia din codul binar-zecimal 8421 (BCD) în codul binar-zecimal exces 3. Pozițiile cuvântului binar din codul BCD se notează cu *D*, *C*, *B*, *A*, iar cele din cuvântul binar al codului exces 3 cu E_3, E_2, E_1, E_0. Rezultă un tabel de adevăr (Tabelul 3.1).

Tabelul 3.1. Tabelul de adevăr pentru conversia din codul BCD în codul exces 3.

Zecimal	BCD	Exces 3
	DCBA	$E_3E_2E_1E_0$
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Deoarece pentru combinațiile variabilelor de intrare corespunzătoare valorilor 10-15 codurile exces 3 nu sunt definite, şi în mod normal aceste combinații nu apar, ele pot fi considerate redundante. Variabilele de ieşire se pot exprima în funcție de cele de intrare astfel:

$$
\begin{aligned}
E_3 &= \Sigma\,(5, 6, 7, 8, 9) + \Sigma_\Phi\,(10, 11, 12, 13, 14, 15)\\
E_2 &= \Sigma\,(1, 2, 3, 4, 9) + \Sigma_\Phi\,(10, 11, 12, 13, 14, 15)\\
E_1 &= \Sigma\,(0, 3, 4, 7, 8) + \Sigma_\Phi\,(10, 11, 12, 13, 14, 15)\\
E_0 &= \Sigma\,(0, 2, 4, 6, 8) + \Sigma_\Phi\,(10, 11, 12, 13, 14, 15)
\end{aligned}
\tag{3.1}
$$

Diagramele Karnaugh corespunzătoare variabilelor de ieşire sunt prezentate în Figura 3.1.

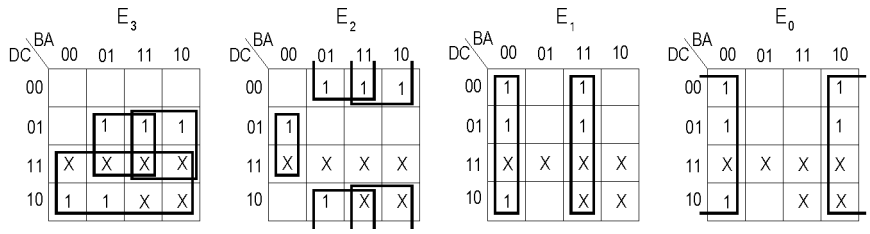

Figura 3.1. Diagramele Karnaugh ale ieşirilor convertorului de cod din BCD în exces 3.

După minimizare se obține:

$$
\begin{aligned}
E_3 &= D + AC + BC = \overline{\overline{D}\cdot\overline{AC}\cdot\overline{BC}}\\
E_2 &= A\overline{C} + B\overline{C} + \overline{A}\overline{B}C = (A+B)\overline{C} + \overline{(A+B)}C = (A+B)\oplus C\\
E_1 &= \overline{A}\overline{B} + AB = \overline{A\oplus B}\\
E_0 &= \overline{A}
\end{aligned}
\tag{3.2}
$$

Circuitul convertor din codul BCD în codul exces 3 este prezentat în Figura 3.2.

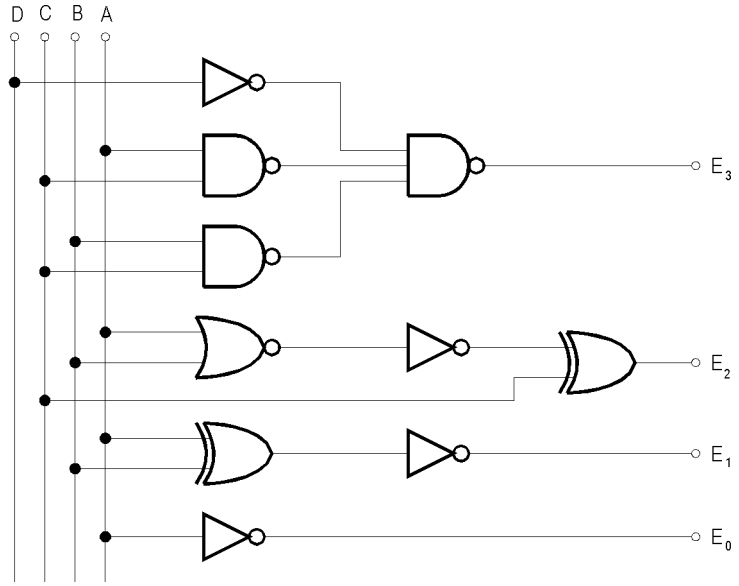

Figura 3.2. Schema logică a convertorului de cod din BCD în exces 3.

2.2. Decodificatoare

Decodificatorul este un circuit combinațional având *n* intrări şi *m* ieşiri, care identifică un cod de intrare prin activarea unei singure linii de ieşire, corespunzătoare acestui cod. Numărul maxim al liniilor de ieşire (numărul de căi) corespunde numărului de combinații ale variabilelor de intrare ($m \le 2^n$). Un decodificator cu 2^n căi se notează cu DCD n:2^n.

Decodificatorul se utilizează în numeroase aplicații, ca de exemplu adresarea memoriilor, selectarea (validarea) unor circuite sau a unor periferice, afişarea datelor etc.

2.2.1. Decodificator de adresă

Acest decodificator activează linia de ieşire a cărei adresă este prezentată la intrare.

Decodificatorul cu 3 intrări de adresă şi $2^3 = 8$ ieşiri este prezentat în Figura 3.3, iar tabelul de adevăr este prezentat în Tabelul 3.2.

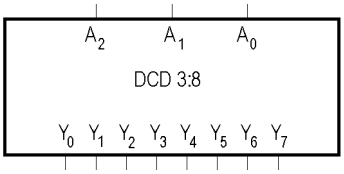

Figura 3.3. Reprezentarea unui decodificator 3:8.

Tabelul 3.2. Tabelul de adevăr al unui decodificator 3:8.

$A_2A_1A_0$	$Y_0Y_1Y_2Y_3Y_4Y_5Y_6Y_7$
0 0 0	1 0 0 0 0 0 0 0
0 0 1	0 1 0 0 0 0 0 0
0 1 0	0 0 1 0 0 0 0 0
0 1 1	0 0 0 1 0 0 0 0
1 0 0	0 0 0 0 1 0 0 0
1 0 1	0 0 0 0 0 1 0 0
1 1 0	0 0 0 0 0 0 1 0
1 1 1	0 0 0 0 0 0 0 1

Din tabelul de adevăr se pot scrie ecuațiile ieşirilor:

$$\begin{aligned}
Y_0 &= \overline{A_2}\,\overline{A_1}\,\overline{A_0} \\
Y_1 &= \overline{A_2}\,\overline{A_1}A_0 \\
Y_2 &= \overline{A_2}A_1\overline{A_0} \\
Y_3 &= \overline{A_2}A_1A_0 \\
Y_4 &= A_2\overline{A_1}\,\overline{A_0} \\
Y_5 &= A_2\overline{A_1}A_0 \\
Y_6 &= A_2A_1\overline{A_0} \\
Y_7 &= A_2A_1A_0
\end{aligned} \qquad (3.3)$$

Se observă că ieşirile acestui decodificator reprezintă de fapt mintermii pentru o funcție de 3 variabile.

Schema decodificatorului este prezentată în Figura 3.4.

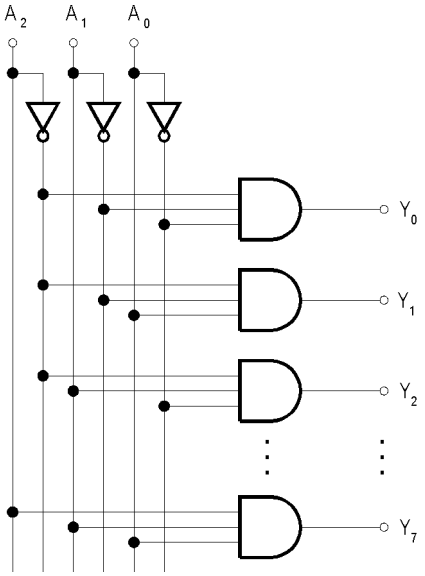

Figura 3.4. Schema logică a decodificatorului de adresă 3:8.

2.2.2. Decodificator din cod BCD în zecimal

Acest decodificator activează una din cele 10 ieşiri corespunzătoare codului BCD de la intrare. O ieşire este activă pe nivelul logic 0. Decodificatorul este prezentat în Figura 3.5.

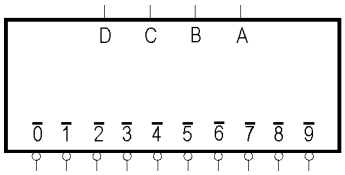

Figura 3.5. Reprezentarea decodificatorului din cod BCD în zecimal.

Pentru combinaţiile corespunzătoare valorilor între 10 şi 15 ieşirile se pot forţa la nivelul logic 1. Toate intrările sunt decodificate deci explicit. Se pot detecta astfel combinaţiile invalide de la intrare. Tabelul de adevăr este prezentat în Tabelul 3.3.

Tabelul 3.3. Tabelul de adevăr al decodificatorului din cod BCD în zecimal.

BCD	Zecimal
DCBA	$\overline{0}$ $\overline{1}$ $\overline{2}$ $\overline{3}$ $\overline{4}$ $\overline{5}$ $\overline{6}$ $\overline{7}$ $\overline{8}$ $\overline{9}$
0 0 0 0	0 1 1 1 1 1 1 1 1 1
0 0 0 1	1 0 1 1 1 1 1 1 1 1
0 0 1 0	1 1 0 1 1 1 1 1 1 1
0 0 1 1	1 1 1 0 1 1 1 1 1 1
0 1 0 0	1 1 1 1 0 1 1 1 1 1
0 1 0 1	1 1 1 1 1 0 1 1 1 1
0 1 1 0	1 1 1 1 1 1 0 1 1 1
0 1 1 1	1 1 1 1 1 1 1 0 1 1
1 0 0 0	1 1 1 1 1 1 1 1 0 1
1 0 0 1	1 1 1 1 1 1 1 1 1 0
1 0 1 0	1 1 1 1 1 1 1 1 1 1

BCD	Zecimal
DCBA	$\overline{0}\ \overline{1}\ \overline{2}\ \overline{3}\ \overline{4}\ \overline{5}\ \overline{6}\ \overline{7}\ \overline{8}\ \overline{9}$
1 0 1 1	1 1 1 1 1 1 1 1 1 1
1 1 0 0	1 1 1 1 1 1 1 1 1 1
1 1 0 1	1 1 1 1 1 1 1 1 1 1
1 1 1 0	1 1 1 1 1 1 1 1 1 1
1 1 1 1	1 1 1 1 1 1 1 1 1 1

Expresiile ieşirilor se pot scrie sub forma următoare:

$$\begin{aligned} \overline{0} &= \overline{\overline{D}\overline{C}\overline{B}\overline{A}} \\ \overline{1} &= \overline{\overline{D}\overline{C}\overline{B}A} \\ &\vdots \\ \overline{9} &= \overline{\overline{D}\overline{C}B\overline{A}} \end{aligned} \tag{3.4}$$

Schema decodificatorului este prezentată în Figura 3.6. Primul rând de inversoare complementează variabilele de intrare, iar cel de-al doilea asigură ca semnalele *D*, *C*, *B*, *A* să fie încărcate cu o singură unitate de sarcină TTL.

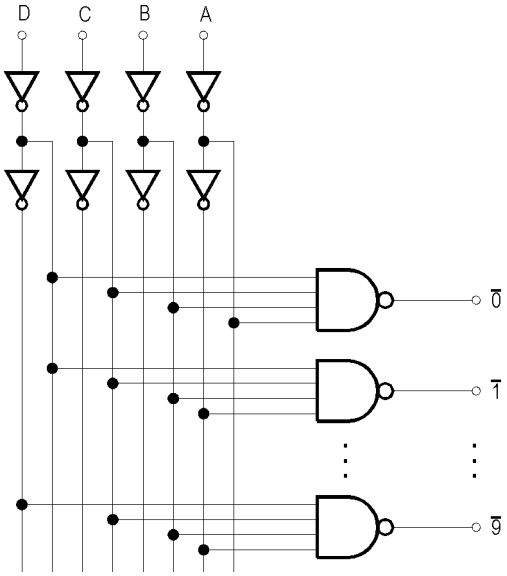

Figura 3.6. Schema logică a decodificatorului din cod BCD în zecimal.

În mod similar se poate realiza un decodificator care să nu detecteze combinaţiile invalide de la intrare. În acest caz se ţine cont de combinaţiile interzise în diagramele Karnaugh. Apariţia unei combinaţii interzise la intrare poate conduce la activarea simultană a mai multor ieşiri.

Decodificatoarele, ca şi alte circuite combinaţionale uzuale, se pot sintetiza prin porţi, sau sunt disponibile sub formă de circuite integrate pe scară medie (MSI). Câteva tipuri de decodificatoare integrate sunt următoarele:

- 7442, 7445, 74141, 74145: DCD 4:10 (convertor de cod) din BCD în zecimal
- 7443: DCD (convertor de cod) din codul exces 3 în zecimal
- 7446, 7448: DCD (convertor de cod) din BCD pentru afişajul cu 7 segmente
- 74154: DCD 4:16

2.2.3. Utilizarea decodificatoarelor pentru implementarea funcțiilor logice

Deoarece decodificatoarele generează toți termenii canonici, complementați sau nu, corespunzători variabilelor de intrare, ele se pot utiliza pentru implementarea funcțiilor logice. Pentru aceasta, funcția se aduce la forma canonică disjunctivă sau conjunctivă, iar minimizarea funcției nu mai este necesară. Implementarea unei funcții exprimate sub forma canonică disjunctivă se realizează utilizând, pe lângă circuitul decodificator, o poartă ŞI-NU cu un număr de intrări egal cu numărul de termeni ai funcției. O altă soluție constă în utilizarea, în locul porții ŞI-NU, a unei porți ŞI la care se conectează terminalele asociate termenilor canonici care nu intervin în expresia funcției.

Considerăm următoarea funcție:

$$F(A, B, C)= P_0 + P_1 + P_5 + P_6 \tag{3.5}$$

Aceasta se poate scrie sub forma:

$$F(A,B,C) = \overline{\overline{P_0} \cdot \overline{P_1} \cdot \overline{P_5} \cdot \overline{P_6}} \tag{3.6}$$

iar complementul funcției se poate scrie, ținând cont de mintermii care lipsesc:

$$\overline{F}(A,B,C) = P_2 + P_3 + P_4 + P_7 \tag{3.7}$$

Funcția inițială are expresia:

$$\overline{\overline{F}}(A,B,C) = F(A,B,C) = \overline{P_2 + P_3 + P_4 + P_7} = \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot \overline{P_7} \tag{3.8}$$

Din ecuațiile (3.6) şi (3.8) rezultă cele două implementări cu decodificatoare care au ieşirile active pe nivelul logic 0. Aceste implementări sunt prezentate în Figura 3.7.

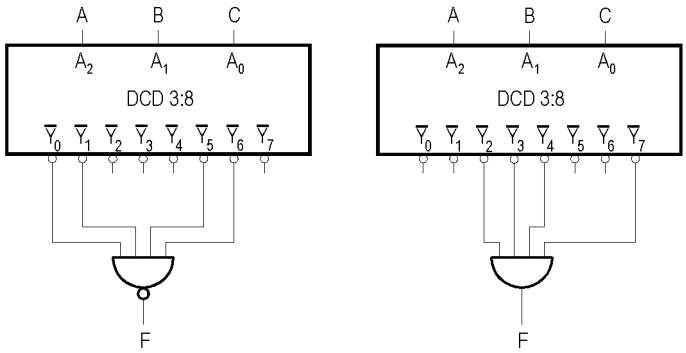

Figura 3.7. Două implementări ale unei funcții booleene prin decodificatoare cu ieşiri active în starea 0 logic.

2.3. Codificatoare

Realizează funcția inversă decodificatoarelor. Un codificator are cel mult 2^n intrări, fiecare intrare corespunzând unui anumit număr de ordine, şi *n* ieşiri. La aplicarea unui semnal logic pe o intrare se obține la ieşire un cuvânt de *n* biți, care reprezintă codul intrării activate. În mod normal, la un moment dat trebuie să fie activă o singură intrare.

De exemplu, codificarea cifrelor zecimale în cod BCD se poate realiza cu un codificator având 10 intrări şi 4 ieşiri. Considerăm un codificator cu starea activă 0 la intrare şi 1 la ieşire. Dacă se aplică un semnal logic 0 la intrarea corespunzătoare unei cifre zecimale, la ieşire se obține codul BCD al cifrei respective. Tabelul de adevăr este prezentat în Tabelul 3.4.

Tabelul 3.4. Tabelul de adevăr al codificatorului din zecimal în cod BCD.

Zecimal	BCD
$\overline{0}\ \overline{1}\ \overline{2}\ \overline{3}\ \overline{4}\ \overline{5}\ \overline{6}\ \overline{7}\ \overline{8}\ \overline{9}$	DCBA
0 1 1 1 1 1 1 1 1 1	0 0 0 0
1 0 1 1 1 1 1 1 1 1	0 0 0 1
1 1 0 1 1 1 1 1 1 1	0 0 1 0
1 1 1 0 1 1 1 1 1 1	0 0 1 1
1 1 1 1 0 1 1 1 1 1	0 1 0 0
1 1 1 1 1 0 1 1 1 1	0 1 0 1
1 1 1 1 1 1 0 1 1 1	0 1 1 0
1 1 1 1 1 1 1 0 1 1	0 1 1 1
1 1 1 1 1 1 1 1 0 1	1 0 0 0
1 1 1 1 1 1 1 1 1 0	1 0 0 1

Ieşirea *D*, de exemplu, este 1 atunci când $\overline{8} = 0$ sau $\overline{9} = 0$. Ecuaţiile ieşirilor se pot scrie astfel:

$$\begin{aligned} D &= \overline{\overline{8}} + \overline{\overline{9}} = 8 + 9 = \overline{\overline{8} \cdot \overline{9}} \\ C &= 4 + 5 + 6 + 7 = \overline{\overline{4} \cdot \overline{5} \cdot \overline{6} \cdot \overline{7}} \\ B &= 2 + 3 + 6 + 7 = \overline{\overline{2} \cdot \overline{3} \cdot \overline{6} \cdot \overline{7}} \\ A &= 1 + 3 + 5 + 7 + 9 = \overline{\overline{1} \cdot \overline{3} \cdot \overline{5} \cdot \overline{7} \cdot \overline{9}} \end{aligned} \tag{3.9}$$

Schema circuitului este prezentată în Figura 3.8.

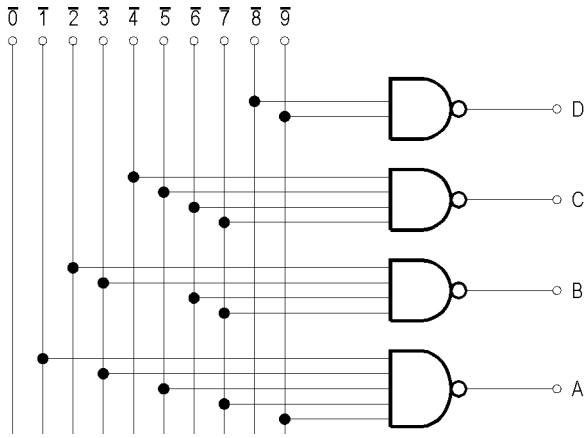

Figura 3.8. Schema logică a codificatorului din zecimal în cod BCD.

Similar se poate realiza un codificator cu starea activă 1 la intrare.

Dezavantajul schemei precedente este că atunci când se activează simultan mai multe intrări, codul de la ieşire este eronat. De exemplu, dacă se activează simultan intrările $\overline{3}$ şi $\overline{5}$, se obţine la ieşire codul 0111, care corespunde intrării $\overline{7}$, neactivate. De aceea, au fost realizate codificatoare prioritare, care generează la ieşire codul corespunzător intrării cu prioritatea cea mai mare. Astfel de codificatoare prioritare, disponibile sub formă integrată, sunt circuitele 74147, cu 9 intrări, şi 74148, cu 8 intrări.

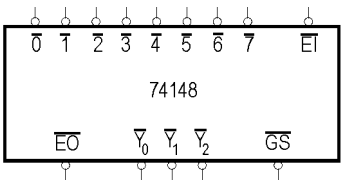

Figura 3.9. Reprezentarea codificatorului 74148.

Codificatorul 74148 (Figura 3.9) dispune de intrările $\overline{0}$, $\overline{1}$, …, $\overline{7}$, şi ieşirile $\overline{Y_0}$, $\overline{Y_1}$, $\overline{Y_2}$. Fiecare din intrări are o prioritate, care creşte cu numărul intrării. Pentru validarea circuitului s-a prevăzut intrarea $\overline{EI}$ (*Enable Input*); dacă circuitul nu este validat, ieşirile sunt inactive (1 logic). Ieşirea $\overline{GS}$ este activată (0 logic) dacă cel puţin una din intrările de date este activată, iar ieşirea $\overline{EO}$ (*Enable Output*) este activată atunci când toate intrările de date sunt inactive. Ieşirea $\overline{EO}$ este necesară atunci când pentru extensie se conectează în cascadă mai multe codificatoare, pentru validarea circuitului similar având intrări de date cu prioritate mai mică. Funcţionarea este descrisă în Tabelul 3.5.

Tabelul 3.5. Tabelul de funcţionare al codificatorului 74148.

Intrări		Ieşiri		
$\overline{EI}$	$\overline{0}$ $\overline{1}$ $\overline{2}$ $\overline{3}$ $\overline{4}$ $\overline{5}$ $\overline{6}$ $\overline{7}$	$\overline{Y_2}$ $\overline{Y_1}$ $\overline{Y_0}$	$\overline{GS}$	$\overline{EO}$
1	x x x x x x x x	1 1 1	1	1
0	1 1 1 1 1 1 1 1	1 1 1	1	0
0	0 1 1 1 1 1 1 1	1 1 1	0	1
0	x 0 1 1 1 1 1 1	1 1 0	0	1
0	x x 0 1 1 1 1 1	1 0 1	0	1
0	x x x 0 1 1 1 1	1 0 0	0	1
0	x x x x 0 1 1 1	0 1 1	0	1
0	x x x x x 0 1 1	0 1 0	0	1
0	x x x x x x 0 1	0 0 1	0	1
0	x x x x x x x 0	0 0 0	0	1

Ecuaţiile ieşirilor sunt următoarele:

$$\begin{aligned} \overline{Y_2} &= \overline{4} \cdot \overline{5} \cdot \overline{6} \cdot \overline{7} \\ \overline{Y_1} &= \overline{2} \cdot \overline{3} \cdot \overline{6} \cdot \overline{7} \\ \overline{Y_0} &= \overline{1} \cdot \overline{3} \cdot \overline{5} \cdot \overline{7} \end{aligned} \tag{3.10}$$

Codificatorul din zecimal în cod BCD prezentat în exemplul precedent poate fi realizat sub formă de codificator prioritar, utilizând circuitul 74148. Ecuaţiile ieşirilor se pot scrie astfel:

$$\begin{aligned} D &= \overline{\overline{8} \cdot \overline{9}} \\ C &= \overline{\overline{4} \cdot \overline{5} \cdot \overline{6} \cdot \overline{7}} = \overline{\overline{Y_2}} \\ B &= \overline{\overline{2} \cdot \overline{3} \cdot \overline{6} \cdot \overline{7}} = \overline{\overline{Y_1}} \\ A &= \overline{\overline{1} \cdot \overline{3} \cdot \overline{5} \cdot \overline{7} \cdot \overline{9}} = \overline{\overline{Y_0} \cdot \overline{9}} \end{aligned} \tag{3.11}$$

Schema circuitului este prezentată în Figura 3.10.

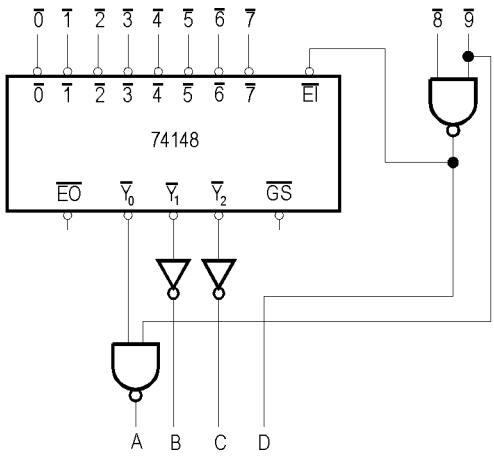

Figura 3.10. Codificator prioritar din zecimal în cod BCD realizat cu circuitul 74148.

Dacă este activă intrarea $\overline{8}$ sau $\overline{9}$ ($\overline{8}\cdot\overline{9} = 0$), trebuie să se invalideze circuitul 74148, astfel încât codul de ieşire să nu depindă decât de cele două intrări $\overline{8}$ şi $\overline{9}$. Trebuie să se aplice nivelul logic 1 pe intrarea $\overline{EI}$, deci $\overline{EI} = \overline{\overline{8}\cdot\overline{9}}$, iar ieşirea *D* trebuie să fie 1: $D = \overline{\overline{8}\cdot\overline{9}}$. În acest caz, $\overline{Y_0} = \overline{Y_1} = \overline{Y_2} = 1$. Dacă $\overline{9} = 1$, $\overline{8} = 0$, *DCBA* = 1000, iar dacă $\overline{9} = 0$, *DCBA* = 1001.

Codificatoarele prioritare se mai utilizează la realizarea sistemelor de întreruperi multiple, la care unitatea centrală va răspunde, din numărul de cereri de întrerupere activate simultan, numai la cea cu prioritatea maximă.

2.4. Multiplexoare

Multiplexorul este un circuit combinaţional care transmite un semnal de la o intrare selectată la o ieşire unică. Se mai numeşte circuit selector. În general, un multiplexor are 2^n intrări de date, $D_0, D_1, ..., D_{2^n-1}$, *n* intrări de selecţie S_0, S_1, …, S_{n-1}, şi o ieşire *Z*. Reprezentarea simbolică este prezentată în Figura 3.11.

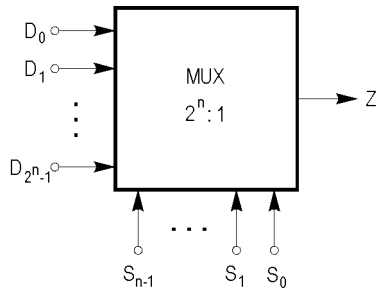

Figura 3.11. Reprezentarea simbolică a unui multiplexor.

Ieşirea *Z* are expresia:

$$Z = D_k \tag{3.12}$$

unde $k = (S_{n-1}\ S_{n-2}\ \ldots\ S_0)$ este echivalentul zecimal al numărului binar reprezentat de intrările de selecţie.

Considerăm un multiplexor cu 4 intrări de date şi 2 intrări de selecţie (Figura 3.12).

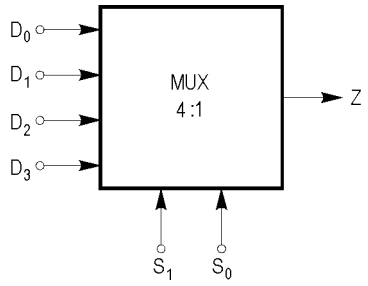

Figura 3.12. Reprezentarea unui multiplexor 4:1.

Tabelul de adevăr al acestui multiplexor este prezentat în Tabelul 3.6.

Tabelul 3.6. Tabelul de adevăr al unui multiplexor 4:1.

S_1S_0	$D_0D_1D_2D_3$	Z
0 0	0 x x x	0
0 0	1 x x x	1
0 1	x 0 x x	0
0 1	x 1 x x	1
1 0	x x 0 x	0
1 0	x x 1 x	1
1 1	x x x 0	0
1 1	x x x 1	1

Ieşirea *Z* are ecuația următoare:

$$Z = \overline{S_1}\,\overline{S_0}D_0 + \overline{S_1}S_0D_1 + S_1\overline{S_0}D_2 + S_1S_0D_3 \tag{3.13}$$

Schema logică este prezentată în Figura 3.13.

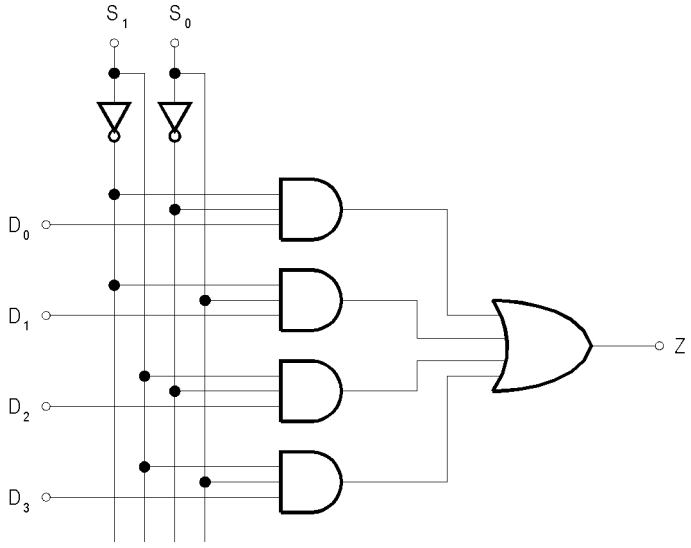

Figura 3.13. Schema logică a unui multiplexor 4:1.

Multiplexoarele integrate posedă, în general, două ieşiri complementare *W* şi $\overline{W}$, şi o intrare de validare a ieşirii sau selecția circuitului $\overline{E}$. Exemple de asemenea circuite sunt următoarele:

- 74150: 16 intrări de date, o intrare de validare $\overline{E}$ şi o ieşire $\overline{W}$.

- 74151: 8 intrări de date, o intrare de validare $\overline{E}$ şi două ieşiri, W şi $\overline{W}$.
- 74152: 8 intrări de date, fără intrare de validare şi o singură ieşire $\overline{W}$.
- 74157: 4 multiplexoare cu câte 2 intrări de date, cu logică de selecţie şi validare comună (o linie de selecţie S şi una de validare $\overline{E}$), şi câte o ieşire necomplementată $1Y$, $2Y$, $3Y$, $4Y$.
- 74158: circuit similar cu 74157, dar cu câte o ieşire complementată.

Reprezentarea simplificată a circuitului 74151 este prezentată în Figura 3.14.

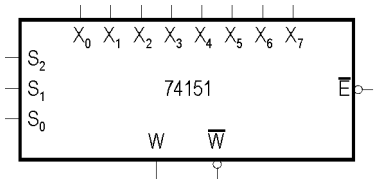

Figura 3.14. Reprezentarea multiplexorului 74151.

Extinderea capacităţii de multiplexare se poate realiza prin conectarea în paralel a mai multor multiplexoare. De exemplu, pentru realizarea unui multiplexor 16:1, se pot conecta două multiplexoare 8:1, ca în Figura 3.15.

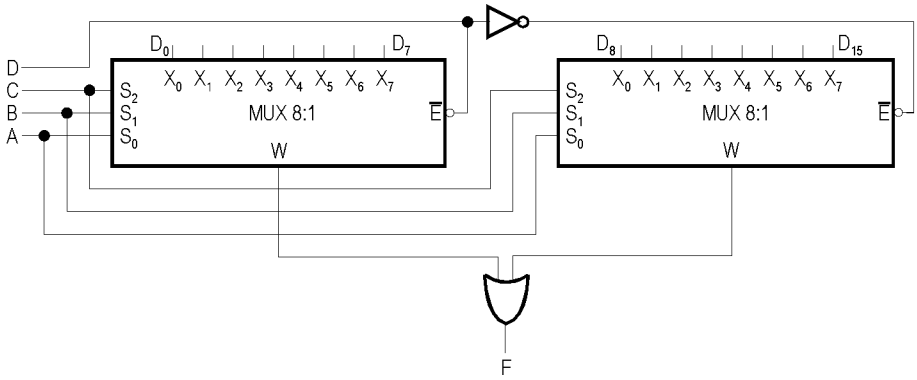

Figura 15. Multiplexor 16:1 realizat prin utilizarea a două multiplexoare 8:1.

Bitul c.m.s. din cuvântul de selecţie se utilizează pentru validare. Primul circuit este selectat de un cuvânt de forma 0××× (de la 0 la 7), iar al doilea de un cuvânt de forma 1××× (de la 8 la 15).

Dacă sunt necesare mai multe linii de selecţie suplimentare, ca de exemplu pentru o extindere de la 8 la 32 de biţi, se poate utiliza un decodificator cu un număr de ieşiri egal cu numărul de multiplexoare utilizate. În general, pentru extinderea multiplexării de 2^k ori sunt necesare k linii de selecţie suplimentare, deci un decodificator cu 2^k linii de ieşire.

De exemplu, pentru extinderea de la 8 la 32 de biţi (de 4 ori) sunt necesare două linii de selecţie suplimentare. Schema unui asemenea circuit este prezentată în Figura 3.16.

Există mai multe utilizări ale multiplexoarelor în cadrul sistemelor de calcul, de exemplu:

- Pentru comutarea mai multor surse de informaţie către o singură destinaţie;
- Pentru realizarea magistralelor de transmitere a informaţiilor;
- Pentru conversia paralel-serie a datelor, aplicând datele în paralel la intrările de date şi modificând succesiv codul de selecţie;
- Pentru implementarea circuitelor combinaţionale.

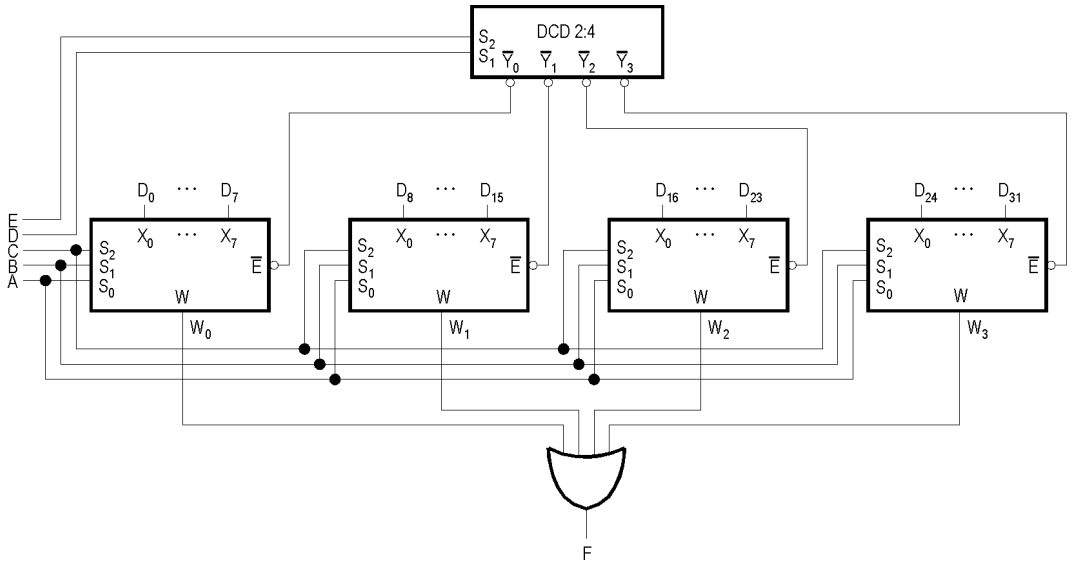

Figura 3.16. Multiplexor 32:1 realizat cu patru multiplexoare 8:1 şi un decodificator.

Din relația de definiție rezultă că multiplexorul generează toți termenii canonici minimali ai variabilelor de selecție, multiplicați cu variabila de intrare. Dacă toate intrările de date sunt la 1 logic, se generează toți mintermii. Eliminarea unora din acestea se realizează conectând la 0 logic intrările de date corespunzătoare.

De exemplu, implementarea funcției de 3 variabile:

$$F(A, B, C) = P_0 + P_1 + P_5 + P_6 \tag{3.14}$$

se realizează aplicând valoarea 1 logic pe intrările X_0, X_1, X_5 şi X_6, şi valoarea 0 logic pe intrările X_2, X_3, X_4 şi X_7 (Figura 3.17).

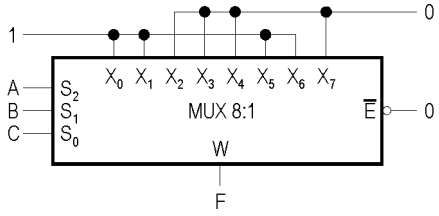

Figura 3.17. Implementarea unei funcții de 3 variabile cu un multiplexor 8:1.

În exemplul anterior, funcția de 3 variabile s-a implementat cu un multiplexor având $2^3 = 8$ intrări. Aceeaşi funcție se poate implementa cu un multiplexor având 4 intrări, dacă una din variabile se aplică pe liniile de intrare, iar celelalte două variabile pe liniile de selecție. Funcția se poate scrie sub forma următoare:

$$\begin{aligned} F(A,B,C) &= \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + A\,\overline{B}\,C + A\,B\,\overline{C} \\ &= \overline{A}\,\overline{B}\cdot 1 + A\,\overline{B}\cdot C + A\,B\cdot\overline{C} \\ &= P_0\cdot 1 + P_1\cdot 0 + P_2\cdot C + P_3\cdot\overline{C} \end{aligned}$$

Această implementare este prezentată în Figura 3.18.

În general, o funcție de *n* variabile se poate implementa cu un multiplexor având 2^{n-1} intrări de date.

La implementarea unui circuit combinațional cu mai multe ieşiri se preferă utilizarea unui decodificator şi a unor porți ŞI-NU sau ŞI, față de utilizarea câte unui multiplexor pentru fiecare ieşire a

circuitului. Dacă, în cazul a *n* variabile nu se dispune de un multiplexor cu 2^{n-1} intrări, ci cu mai puține, implementarea se poate realiza cu mai multe etaje (nivele) de multiplexoare.

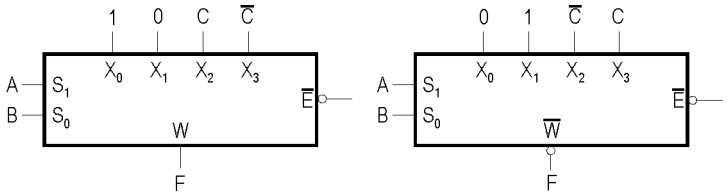

Figura 3.18. Implementarea unei funcții de 3 variabile cu două multiplexoare 4:1.

2.5. Demultiplexoare

Demultiplexorul este un circuit combinațional care dispune, în cazul general, de o intrare de date *D*, *n* intrări de selecție S_0, S_1, …, S_{n-1}, şi 2^n ieşiri $Z_0, Z_1, ..., Z_{2^n-1}$. Intrările de selecție determină apariția semnalului de la intrare la una din cele 2^n ieşiri. Se mai numeşte circuit distribuitor. Reprezentarea simbolică este prezentată în Figura 3.19.

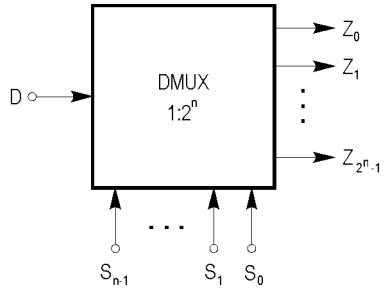

Figura 3.19. Reprezentarea simbolică a unui demultiplexor.

Ieşirea Z_j are ecuația:

$$Z_j = \begin{Bmatrix} 0, & j \neq k \\ D, & j = k \end{Bmatrix} \tag{3.15}$$

pentru $j = 0, 1, 2^{n-1}$, iar $k = (S_{n-1} \ldots S_1 S_0)$ este echivalentul zecimal al numărului binar reprezentat de intrările de selecție.

Considerăm un demultiplexor cu 2 intrări de selecție şi 4 ieşiri (Figura 3.20).

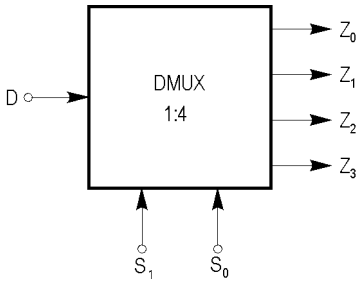

Figura 3.20. Reprezentarea unui demultiplexor 1:4.

Tabelul de adevăr al acestui demultiplexor este prezentat în Tabelul 3.7.

Tabelul 3.7. Tabelul de adevăr al unui demultiplexor 1:4.

S_1S_0	$Z_0Z_1Z_2Z_3$
0 0	D 0 0 0
0 1	0 D 0 0
1 0	0 0 D 0
1 1	0 0 0 D

Ecuațiile ieşirilor sunt următoarele:

$$\begin{aligned} Z_0 &= \overline{S_1}\,\overline{S_0} \cdot D \\ Z_1 &= \overline{S_1} S_0 \cdot D \\ Z_2 &= S_1 \overline{S_0} \cdot D \\ Z_3 &= S_1 S_0 \cdot D \end{aligned} \tag{3.16}$$

Schema logică este prezentată în Figura 3.21.

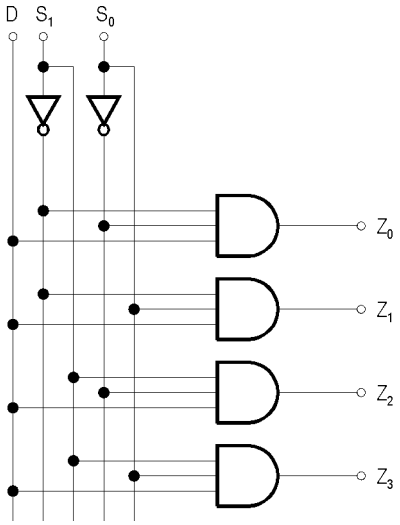

Figura 3.21. Schema logică a unui demultiplexor 1:4.

Un caz particular al demultiplexorului îl constituie decodificatorul, la care intrarea de date lipseşte sau este permanent în starea logică 1.

Circuitele demultiplexoare integrate au, în general, intrările şi ieşirile active pe nivelul logic 0. Exemple de asemenea circuite sunt următoarele:

- 74154: DMUX 1:16, cu 4 intrări de selecție;
- 74155: două DMUX 1:4, cu posibilitatea de a se utiliza ca un DMUX 1:8 (cu 3 intrări de selecție).

Fiecare demultiplexor al circuitului 74155 dispune de o poartă la care una din intrări se poate utiliza ca intrare de date, iar a doua ca intrare de validare (strobare). Tabelul 3.8 prezintă tabelul de funcționare pentru un circuit DMUX 1:4 al circuitului 74155.

Tabelul 3.8. Tabelul de funcționare al circuitului 74155.

Selecție	Strobare	Date	Ieşiri
B A	1G	1C	$\overline{1Y_0}\ \overline{1Y_1}\ \overline{1Y_2}\ \overline{1Y_3}$
x x	1	x	1 1 1 1
0 0	0	1	0 1 1 1
0 1	0	1	1 0 1 1
1 0	0	1	1 1 0 1
1 1	0	1	1 1 1 0
x x	x	0	1 1 1 1

Pentru realizarea unui circuit DMUX 1:8, terminalele 1*C* şi 2*C* se utilizează pentru selecție, iar 1*G* şi 2*G* pentru strobare sau pentru date (Figura 3.22).

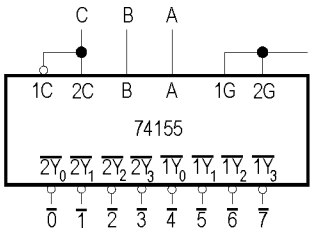

Figura 3.22. Utilizarea circuitului 74155 ca demultiplexor 1:8.

Pentru utilizarea ca decodificator, intrarea de date 1*C* se conectează la 1 logic. Numărul liniilor de ieşire se poate extinde prin conectarea în cascadă.

Deoarece la ieşirea unui demultiplexor se obțin toți termenii canonici minimali ai variabilelor de selecție multiplicați cu variabilele de intrare, cu ajutorul unui circuit DMUX 1:2^n se poate implementa orice funcție logică de *n* variabile sub forma canonică disjunctivă. Termenii canonici sunt introduşi într-o poartă SAU.

Demultiplexoarele se utilizează pentru transmiterea informației de la intrare la o anumită adresă, de exemplu la înscrierea informației în memorie.

2.6. Memorii ROM

Memoriile ROM (*Read Only Memory*) reprezintă o altă categorie de circuite care se pot utiliza pentru implementarea unui set de funcții logice. Aceste memorii conțin un set de locații utilizate pentru stocarea informației binare. Conținutul acestor locații este fixat în momentul fabricației, aceste memorii fiind programate prin măşti. Pe lângă acestea, există şi memorii ROM programabile care utilizează fuzibile, numite PROM (*Programmable* ROM). Acestea pot fi înscrise de către utilizator, cu ajutorul unui echipament de programare corespunzător. De asemenea, există memorii PROM care pot fi şterse cu ajutorul razelor ultraviolete, şi apoi pot fi reprogramate. Acestea se numesc EPROM (*Erasable Programmable* ROM). În fine, memoriile ROM care utilizează o tehnologie de ştergere prin impulsuri electrice se numesc EEPROM sau E^2PROM (*Electrically Erasable Programmable ROM*). Diferitele variante de memorii ROM sunt nevolatile, deci îşi păstrează conținutul şi după întreruperea tensiunii de alimentare.

O memorie ROM are *k* intrări şi *n* ieşiri. Intrările furnizează adresa pentru memorie, iar ieşirile furnizează conținutul cuvântului selectat de adresa de la intrare. Numărul cuvintelor dintr-o memorie ROM este determinat de faptul că prin *k* linii de adresă se pot selecta 2^k cuvinte. Memoria ROM nu are date de intrare, deoarece nu este posibilă operația de scriere. Memoriile ROM integrate dispun de una sau mai multe intrări de validare şi ieşiri cu trei stări pentru a permite realizarea unor memorii de dimensiuni mai mari.

Considerăm, de exemplu, o memorie ROM de 32×8, care conţine 32 de cuvinte de 8 biţi fiecare. Există cinci linii de intrare, care permit specificarea adreselor cuprinse între 0 şi 31. Structura internă a acestei memorii este indicată în Figura 3.23. Intrările sunt decodificate în 32 ieşiri distincte cu ajutorul unui decodificator 5:32. Fiecare ieşire a decodificatorului reprezintă o adresă de memorie. Cele 32 de ieşiri sunt conectate prin intermediul unor conexiuni programabile la fiecare din cele opt porţi SAU. Fiecare poartă SAU trebuie considerată ca având 32 de intrări. Pentru aceste porţi s-au utilizat simboluri simplificate. În locul unor linii de intrare multiple la fiecare din aceste porţi, s-a figurat o singură linie. Liniile de intrare sunt figurate perpendicular pe această linie şi sunt conectate în mod selectiv la porţile respective. O conexiune este indicată printr-un × la intersecţia a două linii. Fiecare ieşire a decodificatorului este conectată prin intermediul unui fuzibil la una din intrările fiecărei porţi SAU. Deoarece fiecare poartă SAU are 32 de conexiuni interne programabile, şi deoarece există opt porţi SAU, această memorie ROM conţine 32×8 = 256 de conexiuni programabile. În general, o memorie ROM cu capacitatea de $2^k \times n$ are un decodificator intern k:2^k şi n porţi SAU. Fiecare poartă SAU are 2^k intrări, care sunt conectate prin conexiuni programabile la fiecare din ieşirile decodificatorului.

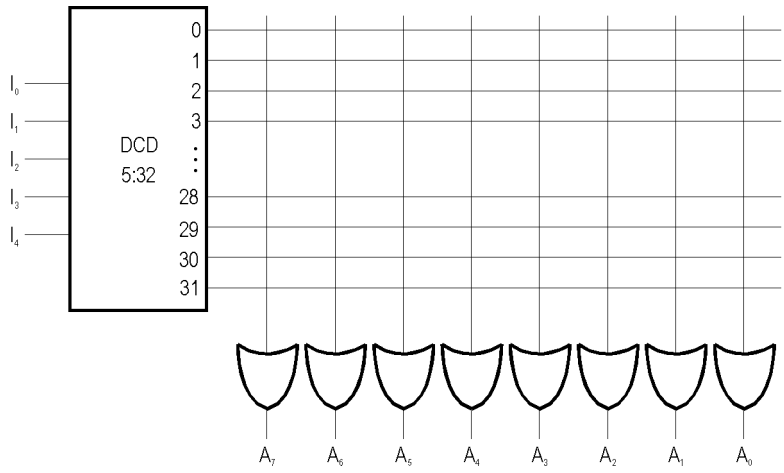

Figura 3.23. Logica internă a unei memorii ROM cu capacitatea de 32 × **8.**

Conţinutul locaţiilor unei memorii ROM poate fi specificat printr-o tabelă de adevăr. De exemplu, conţinutul unei memorii ROM cu dimensiunea de 32×8 poate fi specificat printr-o tabelă de adevăr similară cu cea din Tabelul 3.9. Fiecare combinaţie a intrărilor din tabel specifică adresa unui cuvânt de 8 biţi, a cărei valoare este indicată în coloanele ieşirilor. Tabelul 3.9 prezintă numai primele patru şi ultimele patru cuvinte ale memoriei ROM; tabelul complet trebuie să conţină lista tuturor celor 32 de cuvinte.

Tabelul 3.9. Tabelul de adevăr al unei memorii ROM (parţial).

Intrări	Ieşiri
I_4 I_3 I_2 I_1 I_0	A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0
0 0 0 0 0	1 0 1 1 0 1 1 0
0 0 0 0 1	0 0 0 1 1 1 0 1
0 0 0 1 0	1 1 0 0 0 1 0 1
0 0 0 1 1	1 0 1 1 0 0 1 0
⋮	⋮
1 1 1 0 0	0 0 0 0 1 0 0 1
1 1 1 0 1	1 1 1 0 0 0 1 0
1 1 1 1 0	0 1 0 0 1 0 1 0
1 1 1 1 1	0 0 1 1 0 0 1 1

Programarea memoriei ROM conform tabelului de adevăr de mai sus conduce la configurația din Figura 3.24. Fiecare valoare 0 din tabelul de adevăr specifică un circuit deschis, şi fiecare valoare 1 specifică un circuit închis. De exemplu, tabelul specifică un cuvânt cu valoarea 11000101 la adresa 00010. Cei patru biți de 0 din cadrul cuvântului sunt programați prin deschiderea conexiunilor dintre ieşirea 2 a decodificatorului şi intrările porților SAU asociate cu ieşirile A_5, A_4, A_3 şi A_1. Cei patru biți de 1 din cadrul cuvântului sunt marcați cu un × în schemă, pentru a indica un circuit închis. Atunci când intrarea memoriei ROM este 00010, ieşirea 2 a decodificatorului va fi la 1 logic, celelalte ieşiri fiind la 0 logic. Semnalul cu nivelul 1 logic de la ieşirea 2 a decodificatorului se propagă prin circuitele închise şi porțile SAU la ieşirile A_7, A_6, A_2 şi A_0, iar celelalte ieşiri rămân la 0 logic. Rezultatul este că la ieşirile de date se aplică cuvântul cu valoarea 11000101.

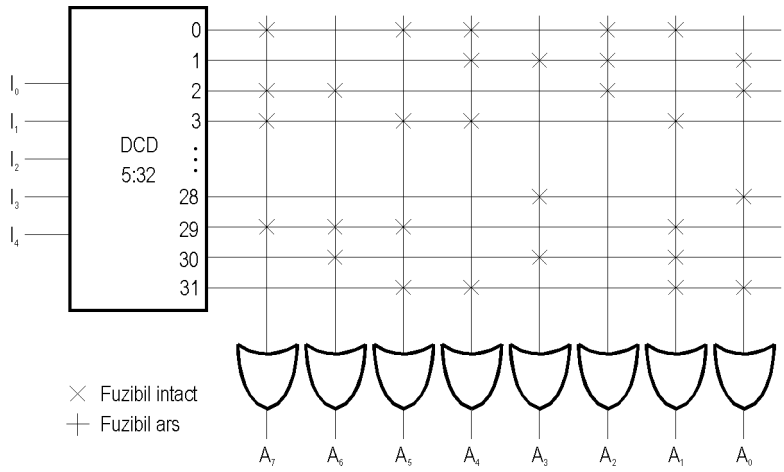

Figura 3.24. Programarea memoriei ROM conform Tabelului 3.9.

După cum s-a arătat anterior, un decodificator generează toți mintermii variabilelor de intrare. Prin inserarea unor porți SAU pentru însumarea mintermilor funcțiilor booleene, se poate implementa orice circuit combinațional. O memorie ROM conține atât un decodificator, cât şi porți SAU în cadrul aceluiaşi circuit. Prin închiderea conexiunilor pentru mintermii incluşi în cadrul funcției, ieşirile memoriei ROM pot fi programate pentru a reprezenta funcțiile booleene ale variabilelor de intrare dintr-un circuit combinațional.

Din punctul de vedere al unui circuit care implementează o funcție booleană, fiecare terminal de ieşire al unei memorii ROM este considerat separat ca ieşire a funcției booleene exprimate ca o sumă de mintermi. De exemplu, memoria ROM din Figura 3.24 poate fi considerată ca un circuit combinațional cu opt ieşiri, fiecare fiind o funcție a celor cinci variabile de intrare. Ieşirea A_7 poate fi exprimată ca o sumă a mintermilor, în felul următor (punctele reprezintă mintermii de la 4 la 27, care nu apar în figură):

$$A_7 (I_4, I_3, I_2, I_1, I_0) = \Sigma (0, 2, 3, \ldots, 29)$$

Memoriile ROM sunt utilizate pentru implementarea circuitelor combinaționale complexe direct din tabelul de adevăr. Aceste memorii sunt utile pentru conversia dintr-un cod în altul, pentru generarea operațiilor aritmetice complexe, ca înmulțirea sau împărțirea, şi în general, pentru aplicații care necesită un număr moderat de intrări şi un număr mare de ieşiri.

În practică, la implementarea unui circuit combinațional printr-o memorie ROM, nu este necesar să se deseneze schema logică sau să se indice conexiunile interne din cadrul memoriei. Trebuie să se specifice doar o anumită memorie ROM prin numărul circuitului şi tabelul de adevăr al memoriei. Tabelul de adevăr conține toate informațiile necesare programării memoriei.

De exemplu, se consideră proiectarea unui circuit combinațional care acceptă la intrare un număr de 3 biți şi generează la ieşire un număr binar egal cu pătratul numărului de la intrare. Prima etapă pentru proiectarea circuitului este întocmirea tabelului de adevăr. În cele mai multe cazuri,

aceasta este singura operație necesară. În alte cazuri, se poate utiliza un tabel de adevăr parțial pentru memoria ROM ținând cont de anumite proprietăți ale variabilelor de ieşire. Tabelul 3.10 reprezintă tabelul de adevăr pentru circuitul proiectat.

Tabelul 3.10. Tabelul de adevăr pentru circuitul care generează pătratele valorilor de la intrare.

Intrări	Ieşiri	
A_2 A_1 A_0	B_5 B_4 B_3 B_2 B_1 B_0	Zecimal
0 0 0	0 0 0 0 0 0	0
0 0 1	0 0 0 0 0 1	1
0 1 0	0 0 0 1 0 0	4
0 1 1	0 0 1 0 0 1	9
1 0 0	0 1 0 0 0 0	16
1 0 1	0 1 1 0 0 1	25
1 1 0	1 0 0 1 0 0	36
1 1 1	1 1 0 0 0 1	49

Pentru circuitul proiectat, sunt necesare trei intrări şi şase ieşiri. Se observă că ieşirea B_0 este egală întotdeauna cu intrarea A_0, astfel încât nu este necesară generarea acestei ieşiri cu ajutorul memoriei ROM. De asemenea, ieşirea B_1 este întotdeauna 0. Astfel, trebuie să se genereze doar patru ieşiri cu memoria ROM. Memoria trebuie să aibă minimum trei intrări şi patru ieşiri. Cele trei intrări specifică opt cuvinte, deci dimensiunea minimă a memoriei necesare este de 8×4. Schema circuitului este prezentată în Figura 3.25.

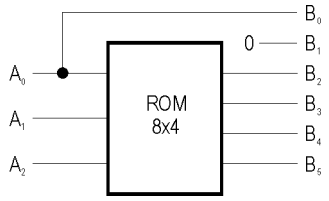

Figura 3.25. Implementarea cu o memorie ROM a circuitului care generează pătratele valorilor de la intrare.

2.7. Rețele logice programabile

O rețea logică programabilă PLA (*Programmable Logic Array*) este similară ca şi concept cu o memorie ROM, cu excepția faptului că nu realizează decodificarea completă a variabilelor şi nu generează toți mintermii. Decodificatorul este înlocuit cu o rețea de porți ŞI care poate fi programată pentru a genera termenii produs ai variabilelor de intrare. Termenii produs sunt apoi conectați în mod selectiv cu porți SAU pentru a genera suma termenilor produs pentru funcțiile booleene necesare. Structura de bază a unui circuit PLA este prezentată în Figura 3.26.

Un circuit PLA poate implementa în mod direct un set de funcții logice exprimate printr-un tabel de adevăr. Fiecare intrare pentru care valoarea funcției este adevărată necesită un termen produs, şi acestuia îi corespunde o linie de porți ŞI din primul etaj al circuitului PLA. Fiecare ieşire corespunde la o linie de porți SAU din al doilea etaj al circuitului. Numărul de porți SAU corespunde cu numărul de intrări din tabela de adevăr pentru care ieşirea este adevărată. Dimensiunea totală a circuitului PLA este egală cu suma dintre dimensiunea rețelei de porți ŞI şi dimensiunea rețelei de porți SAU. Din Figura 3.26 se observă că dimensiunea rețelei de porți ŞI este egală cu numărul de intrări multiplicat cu numărul diferiților termeni produs, iar dimensiunea rețelei de porți SAU este egală cu numărul de ieşiri multiplicat cu numărul termenilor produs.

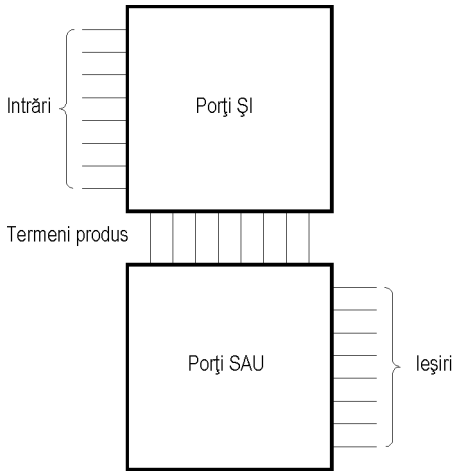

Figura 3.26. Structura generală a unui circuit PLA.

Un circuit PLA are două caracteristici care determină o implementare eficientă a unui set de funcţii logice. Prima este că singurele intrări din tabelul de adevăr care necesită porţi logice sunt cele cărora le corespunde o valoare adevărată pentru cel puţin o ieşire. A doua este că un anumit termen produs va avea o singură intrare în circuitul PLA, chiar dacă termenul produs este utilizat în mai multe ieşiri.

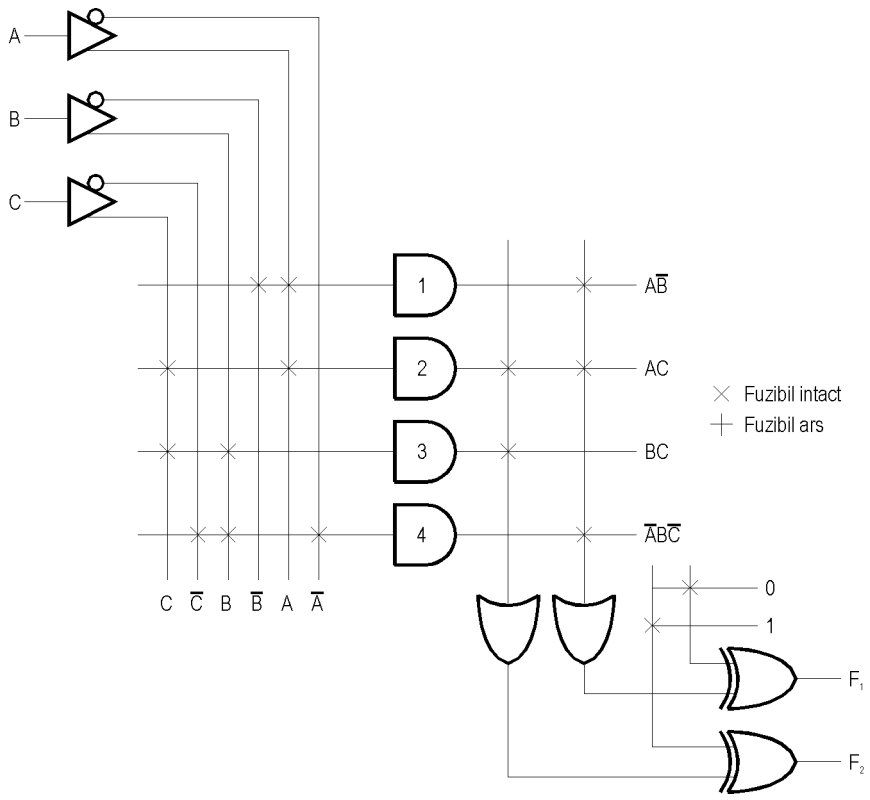

Figura 3.27. Structura internă a unui circuit PLA cu trei intrări, patru termeni produs şi două ieşiri.

Considerăm implementarea următoarelor funcţii booleene într-un circuit PLA:

$$F_1 = A\overline{B} + AC + \overline{A}B\overline{C}$$
$$F_2 = \overline{AC + BC} \tag{3.17}$$

Structura internă a circuitului PLA care implementează aceste funcţii este prezentată în Figura 3.27. Fiecare intrare trece printr-un buffer şi un inversor. Există conexiuni programabile de la fiecare intrare şi complementul acesteia la intrările fiecărei porţi ŞI, indicate prin intersecţiile dintre liniile verticale şi orizontale. Ieşirile porţilor ŞI au conexiuni programabile la intrările fiecărei porţi SAU. Ieşirea fiecărei porţi SAU constituie intrare într-o poartă SAU EXCLUSIV, iar cealaltă intrare poate fi programată fie la 1 logic, fie la 0 logic. Ieşirea este inversată dacă această intrare este 1 logic (deoarece $X \oplus 1 = \overline{X}$), şi rămâne neschimbată dacă această intrare este 0 logic (deoarece $X \oplus 0 = X$).

Modul de programare a circuitului PLA se poate specifica sub formă tabelară. De exemplu, pentru circuitul din Figura 3.27 programarea este specificată în Tabelul 3.11. Tabelul constă din trei secţiuni. Prima secţiune listează numerele termenilor produs. A doua secţiune specifică conexiunile necesare dintre intrări şi porţile ŞI. A treia secţiune specifică conexiunile dintre porţile ŞI şi porţile SAU. Fiecare variabilă de ieşire poate fi adevărată (A) sau complementată (C), ceea ce se controlează prin poarta SAU EXCLUSIV. Pentru fiecare termen produs, intrările sunt marcate cu 1, 0 sau –. Dacă o variabilă dintr-un termen produs apare sub formă necomplementată, variabila este marcată cu 1. Dacă o variabilă dintr-un termen produs apare sub formă complementată, ea este marcată cu 0. Dacă variabila lipseşte din termenul produs, ea este marcată cu –. Variabilele de ieşire sunt marcate cu 1 pentru termenii produs care sunt incluşi în cadrul funcţiei. O ieşire marcată cu A indică faptul că cealaltă intrare a porţii SAU EXCLUSIV corespunzătoare trebuie conectată la 0, iar o ieşire marcată cu C indică o conectare la 1.

Tabelul 3.11. Tabelul de programare pentru circuitul PLA din Figura 3.27.

	Termen produs	Intrări	Ieşiri
		A B C	(A) (C) F_1 F_2
$A\overline{B}$	1	1 0 –	1 –
AC	2	1 – 1	1 1
BC	3	– 1 1	– 1
$\overline{A}\overline{B}C$	4	0 1 0	1 –

Dimensiunea unui circuit PLA este specificată prin numărul intrărilor, numărul termenilor produs şi numărul ieşirilor. Un circuit PLA tipic are 16 intrări, 48 de termeni produs şi 8 ieşiri. Pentru n intrări, k termeni produs şi m ieşiri, logica internă a circuitului PLA constă din n buffere-inversoare, k porţi ŞI, m porţi SAU, şi m porţi SAU EXCLUSIV. Există $2n \times k$ conexiuni programabile între intrări şi porţile ŞI, $k \times m$ conexiuni programabile între porţile ŞI şi porţile SAU, şi m conexiuni programabile asociate cu porţile SAU EXCLUSIV.

Pentru proiectarea unui sistem digital cu un circuit PLA, nu este necesar să se indice conexiunile interne ale circuitului, ci trebuie să se specifice doar tabela de programare. Ca şi în cazul memoriilor ROM, circuitele PLA pot fi programate prin măşti sau programate de către utilizator. Circuitele PLA programate de către utilizator se numesc FPLA (*Field Programmable Logic Array*).

La implementarea funcţiilor logice cu ajutorul circuitelor PLA, trebuie să se reducă numărul termenilor produs în scopul reducerii complexităţii circuitului. Pentru aceasta trebuie simplificate funcţiile booleene astfel încât acestea să aibă un număr minim de termeni. Numărul de literale dintr-un termen este mai puţin important, deoarece toate variabilele de intrare sunt disponibile. Trebuie simplificată atât forma necomplementată, cât şi cea complementată a fiecărei funcţii pentru a se determina care din acestea se poate exprima cu ajutorul unui număr mai mic de termeni produs şi care utilizează termeni produs care sunt utilizaţi şi de alte funcţii.

3. Desfăşurarea lucrării

3.1. Se va realiza sinteza prin porţi logice a unui convertor din codul exces 3 în BCD.

3.2. Se va realiza sinteza prin porţi logice a unui convertor din cod binar în codul Gray.

3.3. Se va implementa cu un decodificator şi porţi externe circuitul combinaţional definit prin următoarele funcţii booleene:

$$F_1(A,B,C) = \overline{A}\overline{B} + AB\overline{C}$$
$$F_2(A,B,C) = \overline{A} + C$$
$$F_3(A,B,C) = AB + \overline{A}\overline{B}$$

3.4. Se va realiza un codificator prioritar de 16 biţi cu două circuite 74148.

3.5. Se va implementa cu un multiplexor 8:1 următoarea funcţie booleană:

$$F\,(A, B, C, D) = \Sigma\,(2, 3, 5, 6, 8, 9, 12, 14)$$

3.6. Se va întocmi tabelul de adevăr pentru o memorie ROM cu dimensiunea 8×4 care implementează următoarele funcţii booleene:

$$W\,(A, B, C) = \Sigma\,(3, 6, 7)$$
$$X\,(A, B, C) = \Sigma\,(0, 1, 4, 5, 6)$$
$$Y\,(A, B, C) = \Sigma\,(2, 3, 4)$$
$$Z\,(A, B, C) = \Sigma\,(2, 3, 4, 7)$$

3.7. Se va întocmi tabelul de programare al unui circuit PLA pentru cele patru funcţii booleene de la punctul 3.6. Se va minimiza numărul termenilor produs şi se va încerca partajarea acestor termeni între mai multe funcţii.

3.8. Se va întocmi tabelul de programare al unui circuit PLA pentru circuitul combinaţional care generează pătratul unui număr de 3 biţi. Se va minimiza numărul termenilor produs.

3.9. Se vor implementa următoarele funcţii booleene cu ajutorul unui circuit PLA:

$$F_1\,(A, B, C) = \Sigma\,(0, 1, 2, 4)$$
$$F_2\,(A, B, C) = \Sigma\,(0, 5, 6, 7)$$