Cursul 4

3.2 Automatul de ordinul zero. Circuite logice combinaţionale

Circuitul logic combinaţional constituie modelul fizic al automatului finit de ordinul zero. Aşa cum s-a arătat acest automat este caracterizat de faptul că variabilele de ieşire sunt independente de timp, (automatul nu are în structura sa elemente de memorie care să conţină informaţii legate de evoluţia în timp a intrărilor), valoarea lor fiind determinată doar de valoarea variabilelor de intrare la momentul respectiv. În figura 3.15 este dată schema bloc a unui circuit logic combinaţional cu "n" intrări şi "m" ieşiri.

Figura 3.15

Pentru acest circuit se poate scrie setul de funcţii:

$$y_1 = f_1(u_1, u_2, \cdots, u_n)$$
$$y_2 = f_2(u_1, u_2, \cdots, u_n)$$
$$\vdots$$
$$y_m = f_m(u_1, u_2, \cdots, u_n)$$

unde $f_1, f_2, \cdots, f_m$ sunt funcţii logice.

Legat de realizabilitatea fizică a circuitelor logice combinaţionale se pot formula două tipuri de probleme: de sinteză şi de analiză. În continuare se vor dezvolta aceste probleme.

3.2.1 Sinteza şi analiza circuitelor logice combinaţionale

Problemele de sinteză ale circuitelor logice combinaţionale se definesc în modul următor: cunoscând semnalele de ieşire corespunzătoare diferitelor combinaţii ale semnalelor de intrare se cere să se stabilească structura circuitului logic combinaţional.

În general, sinteza unui circuit logic combinaţional presupune parcurgerea următoarelor etape:

1) Pe baza enunţului temei de proiectare se procedează la completarea tabelului de adevăr;
2) Se realizează minimizarea funcţiilor logice prin una din metodele cunoscute;

3) Se implementează funcţiile logice obţinute cu circuite logice;
4) Se analizează circuitul obţinut pentru a vedea dacă corespunde condiţiilor impuse iniţial.

În cadrul problemelor de analiză se cunoaşte structura logică a circuitului combinaţional şi se cere să se stabilească valorile posibile ale semnalelor de ieşire pentru toate combinaţiile posibile de valori ale semnalelor de intrare. Problemele de analiză se soluţionează căutând expresiile funcţiilor logice corespunzătoare semnalelor de ieşire pentru ca ulterior să se poată determina valorile acestor funcţii pentru toate combinaţiile posibile de valori ale semnalelor de intrare. Pentru a putea determina valorile acestor funcţii este necesar să se găsească termenii canonici ai funcţiei. Etapele ce trebuiesc parcurse în vederea analizei unui circuit combinaţional sunt:

1) Pe baza structurii logice a circuitului logic combinaţional se determină, din aproape în aproape funcţiile logice ale semnalelor de ieşire în formă normală;
2) Se dezvoltă aceste funcţii logice în formă canonică;
3) Folosind tabelul de adevăr se stabilesc valorile funcţiilor corespunzătoare semnalelor de ieşire pentru toate combinaţiile posibile de valori ale semnalelor de intrare;
4) O etapă neobligatorie, dar uneori interesantă, constă în stabilirea de expresii minimale ale funcţiilor logice în formă canonică. Această etapă vizează realizarea unui circuit echivalent cu circuitul analizat, dar mai economic. Nu trebuie omis însă faptul că în procesul de sinteză au putut interveni şi alte criterii decât cel de economicitate (de ex. funcţionare fără hazard).

3.2.2 Circuite logice combinaţionale sintetizate cu porţi logice

Sinteza circuitelor logice combinaţionale cu porţi logice (adică cu acele circuite logice care materializează fizic funcţiile logice elementare) presupune ca etapă obligatorie minimizarea funcţiilor logice în formă normală disjunctivă sau conjunctivă. Operaţia de implementare a funcţiilor logice în scheme cu porţi logice constituie o problemă de optimizare a cărei rezolvare depinde de experienţa proiectantului în condiţiile existenţei unor restricţii privind: numărul de porţi logice într-o capsulă de circuit integrat; numărul de intrări ale fiecărei porţi logice; tipul de poartă logică care asigură utilizarea numărului minim de capsule integrate; etc.

Observaţie: la sinteza circuitelor logice combinaţionale cu mai multe variabile de ieşire se poate obţine o reducere a numărului de porţi logice folosite dacă se identifică termeni comuni în expresiile funcţiilor de ieşire.

3.2.2.1 Sinteza cu porţi logice NU, SI, SAU

La baza sintezei circuitelor combinaţionale cu porţi logice NU, SI, SAU stă forma normală minimă disjunctivă a funcţiei logice, atunci când variabilele de intrare ale circuitului combinaţional se aplică la intrările unor porti logice SI, iar ieşrile se culeg de la ieşirile unor porţi logice de tip SAU, respectiv formă normală minimă conjunctivă a funcţiei, atunci când variabilele de intrare ale circuitului combinaţional se aplică la intrările unor porţi logice SAU, iar ieşirile se culeg de la ieşirile unor porţi logice SI. Se mai spune că circuitele combinaţionale ce implementează funcţii logice în forma normală disjunctivă au pe nivelul de intrare porţi logice de tip SI respectiv pe nivelul de ieşire porţi logice de tip SAU şi respectiv invers pentru circuitele combinaţionale ce implementează funcţii logice în formă normală conjunctivă. Formele normale ale funcţiilor logice descriu structural schema logică a circuitului ce trebuie realizat.

În practică, sinteza cu porţi logice NU, SI, SAU, dispuse pe două nivele, ridică probleme numai atunci când porţile logice folosite au un număr de intrări mai mic decât cel necesar, unele sunt încărcate peste limita de sarcină admisă, respectiv unele ieşiri nu pot realiza comanda tuturor circuitelor care s-ar impune conform expresiei logice.

În practică sinteza cu porţi logice SI, SAU, NU este folosită arareori întrucât în familia de circuite integrate TTL, care este familia cea mai utilizată, există o gamă restrânsă de circuite integrate care implementează funcţiile logice SI, respectiv SAU.

Din acelaşi motiv se practică sinteza utilizând porţile logice SI-NU, respectiv SAU-NU.

3.2.2.2 Sinteza cu porti logice SI-NU, respectiv SAU-NU

Funcţiile logice NU, SI, SAU formează un sistem logic complet întrucât orice funcţie logică poate fi implementată cu circuite care materializează funcţiile logice NU, SI, SAU.

Există şi alte sisteme logice complete, dintre acestea mai frecvent utilizate fiind sistemele bazate pe functiile logice SI-NU respectiv SAU-NU. Pentru a demonstra că şi aceste funcţii formează un sistem logic complet

este suficient să demonstrăm că ele pot implementa funcţiile logice NU, SI, SAU.

Observaţii: Dacă aplicăm teoremele lui De Morgan pentru cele două funcţii logice rezultă:

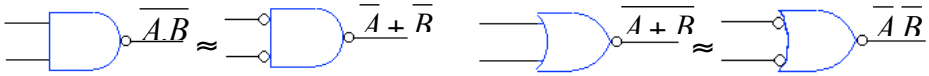


Figura 3.16

1) Un circuit SI-NU poate fi interpretat ca fiind compus dintr-un circuit SI urmat de un inversor sau ca un circuit SAU la care se aplică variabilele negate.
2) Un circuit SAU-NU poate fi interpretat ca fiind un circuit SAU urmat de un inversor sau ca un circuit SI la care se aplică variabilele negate.

Pornind de la aceste două observaţii, rezultă următoarele modalităţi de implementare a funcţiilor logice NU, SI, SAU cu circuite ce materializează fizic funcţiile SI-NU, respectiv SAU-NU.

Funcţia logică NU poate fi implementată cu un circuit SI-NU dacă la intrările neutilizate se aplică semnalul logic "1", respectiv cu un circuit SAU-NU dacă la intrările neutilizate se aplică "0" logic. De asemenea funcţia NU poate fi implementată cu oricare din aceste circuite dacă varibila ce trebuie negată se aplică la toate intrările circuitului SI-NU, respectiv SAU-NU (fig. 3.17).

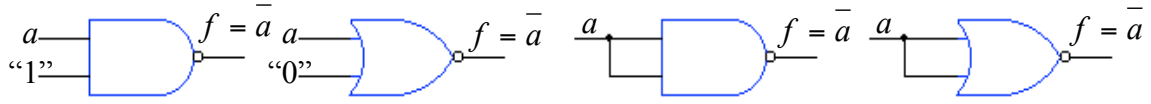


Figura 3.17

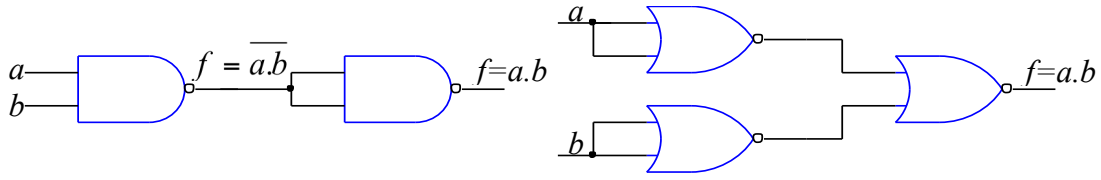
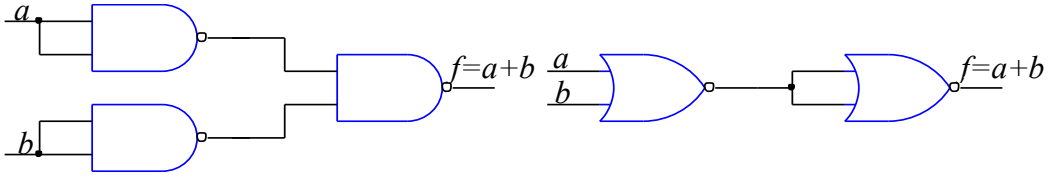


Figura 3.18



Funcţia logica SI poate fi implementată cu un circuit SI-NU urmat de un inversor sau cu un circuit SAU-NU la intrările căruia se aplică varibilele negate (fig.3.18).

Funcţia logică SAU poate fi implementată cu un circuit SI-NU la intrările căruia se aplică variabilele negate sau cu un circuit SAU-NU urmate de un inversor.

Rezultă deci că orice funcţie logică poate fi implementată folosind numai circuite SI-NU, respectiv SAU-NU.

Pentru sinteza circuitelor logice combinaţionale cu circuite SI-NU se parcurg următoarele etape:

(a) – se obţine expresia minimă a funcţiei în formă normală disjunctivă;

(b) – se realizează o schemă logică cu circuite SI-NU plasate pe două nivele;

(c) – variabilele care apar singure în sumă se aplică la intrarea porţii ce generează semnalul de ieşire în formă complementată.

Exemplu:

$f = \bar{a}.\bar{b}.\bar{c} + b.c + \bar{c}$ este implementată de schema din fig. 3.20:

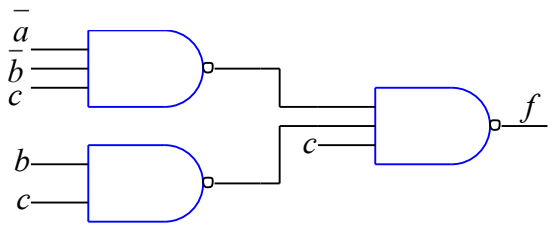


Figura 3.20

Pentru sinteza circuitelor logice combinaţionale cu circuite SAU-NU se parcurg următoarele etape:

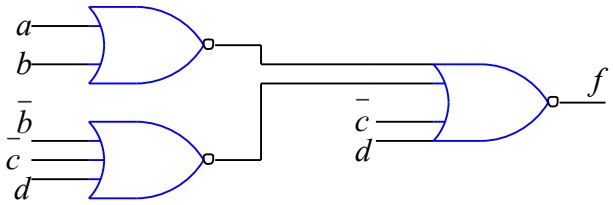
(a) – se obţine expresia minimă a funcţiei în formă normală conjunctivă;

(b) – se realizează o schemă logică cu circuite SAU-NU plasate pe două nivele;

(c) – variabilele care apar singure în produs se aplică la intrarea porţii ce generează semnalul de ieşire în formă complementată.

Exemplu:

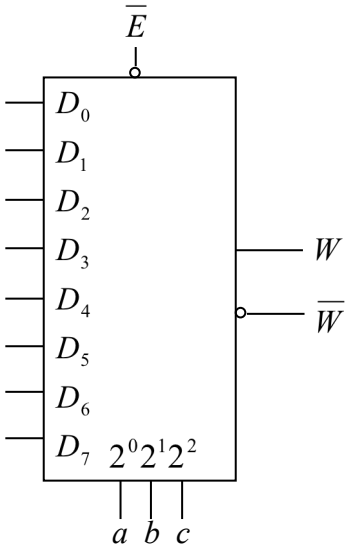
$f = (a + b)(\bar{b} + \bar{c} + d).c.\bar{d}$ este implementată de schema din fig.3.21:

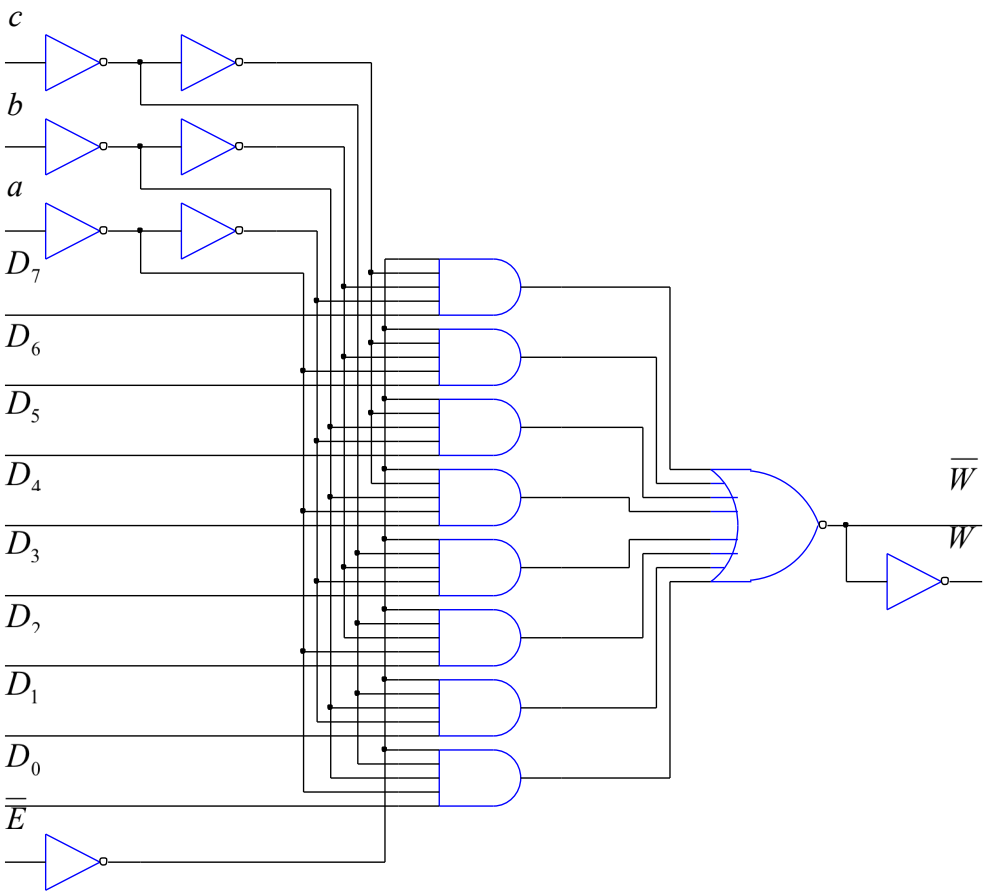


Observaţie: Metodele prezentate în acest paragraf şi în paragraful precedent sunt utilizate în general pentru implementarea funcţiilor logice simple. Pentru funcţii mai complicate se utilizează circuite integrate pe scară medie (ex. multiplexor, decodificator) sau largă (ex. memoria), circuite care realizează, în general, funcţii logice bine determinate. Pentru utilizarea lor la implementarea unor funcţii logice oarecare sunt necesare si porţi logice.

3.2.3 Multiplexorul

Multiplexorul este un circuit logic combinaţional, integrat pe scară medie, obţinut printr-o conexiune (extensie) de tip serie a unor porţi logice ce constituie elementul tipic de circuit logic combinaţional. Din punct de vedere funcţional multiplexorul este un selector ce conectează la ieşire intrarea adresată. Circuitul are în general 2^n intrări de date şi "n" intrări de adresă. Valoarea ieşirii este determinată de valoarea intrării selectate prin adresă.





INTRARI												IESIRI	
c	b	a	$\overline{E}$	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	W	$\overline{W}$
0	0	0	0	D_0	*	*	*	*	*	*	*	D_0	$\overline{D_0}$
0	0	1	0	*	D_1	*	*	*	*	*	*	D_1	$\overline{D_1}$
0	1	0	0	*	*	D_2	*	*	*	*	*	D_2	$\overline{D_2}$
0	1	1	0	*	*	*	D_3	*	*	*	*	D_3	$\overline{D_3}$
1	0	0	0	*	*	*	*	D_4	*	*	*	D_4	$\overline{D_4}$
1	0	1	0	*	*	*	*	*	D_5	*	*	D_5	$\overline{D_5}$
1	1	0	0	*	*	*	*	*	*	D_6	*	D_6	$\overline{D_6}$
1	1	1	0	*	*	*	*	*	*	*	D_7	D_7	$\overline{D_7}$
*	*	*	1	*	*	*	*	*	*	*	*	0	1

Figura 3.22

Sub formă integrată se produc mai multe tipuri de multiplexoare dintre care amintim multiplexorul cu 16 intrări de date şi 4 intrări de adresare, 8 intrări de date şi 3 de adresare şi 4 intrări de date şi 2 de adresare. În fig. 3.22 este dată schema desfăşurată, schema bloc şi tabelul de funcţionare ale multiplexorului cu 8 intrări de date, 3 intrări de adresare, o

intrare de validare a funcţionării (activă pe 0) şi două ieşiri complementare W şi $\overline{W}$.

Funcţia logică pe care o implementează acest multiplexor este:

$$W = D_0.\overline{a}.\overline{b}.\overline{c} + D_1.a.\overline{b}.\overline{c} + D_2.\overline{a}.b.\overline{c} + D_3.a.b.\overline{c} + D_4.\overline{a}.\overline{b}.c + D_5.a.\overline{b}.c + D_6.\overline{a}.b.c + D_7.a.b.c = D_0.P_0 + D_1.P_1 + D_2.P_2 + D_3.P_3 + D_4.P_4 + D_5.P_5 + D_6.P_6 + D_7.P_7$$

Dintre aplicaţiile posibile ale circuitelor multiplexoare amintim selecţia secvenţială, conversia paralel-serie a datelor şi transmisia multiplexată a informaţiilor de pe mai multe linii pe o singură linie.

Pe lângă aceste aplicaţii multiplexoarele pot fi utilizate şi pentru implementarea funcţiilor logice. Astfel cu un multiplexor cu 2^n intrari de date poate fi implementată orice funcţie logică de "n" variabile aplicând la intrările de date valorile funcţiei, iar la intrările de adresare cele "n" variabile. De exemplu funcţia de trei variabile

$$f_1 = \overline{a}.\overline{b}.\overline{c} + \overline{a}.b.c + a.\overline{b}.\overline{c} + a.b.\overline{c}$$

este implementată de multiplexorul din fig. 3.23.

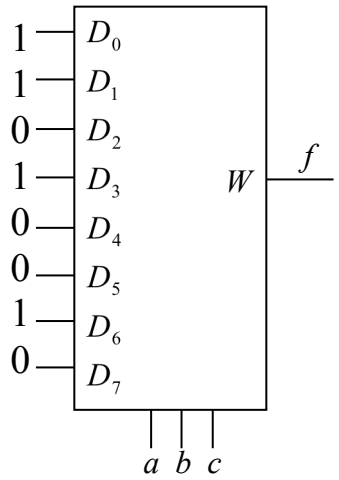
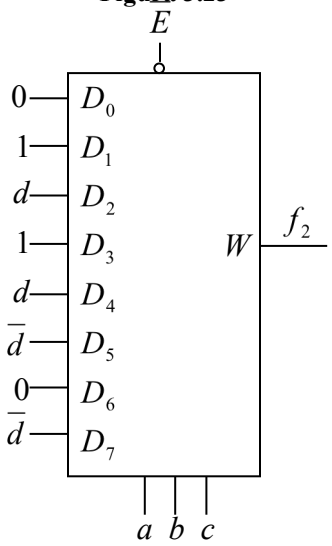


Figura 3.23

	a	b	c	d	f_2
D_0	0	0	0	0	0
	0	0	0	1	0
D_1	0	0	1	0	1
	0	0	1	1	1
D_2	0	1	0	0	0
	0	1	0	1	1
D_3	0	1	1	0	1
	0	1	1	1	1
D_4	1	0	0	0	0
	1	0	0	1	1
D_5	1	0	1	0	1
	1	0	1	1	0
D_6	1	1	0	0	0
	1	1	0	1	0
D_7	1	1	1	0	1
	1	1	1	1	0

Figura 3.24



Un multiplexor cu 2^n intrări de date poate implementa şi funcţii de mai mult de "n" variabile dacă se utilizează şi porţi logice suplimentare. Pentru funcţii de "n+1" variabile se aleg "n" variabile care se aplică la intrările de adresare, iar a "n+1" variabilă se aplică în formă normală sau complementată la unele intrări de date.

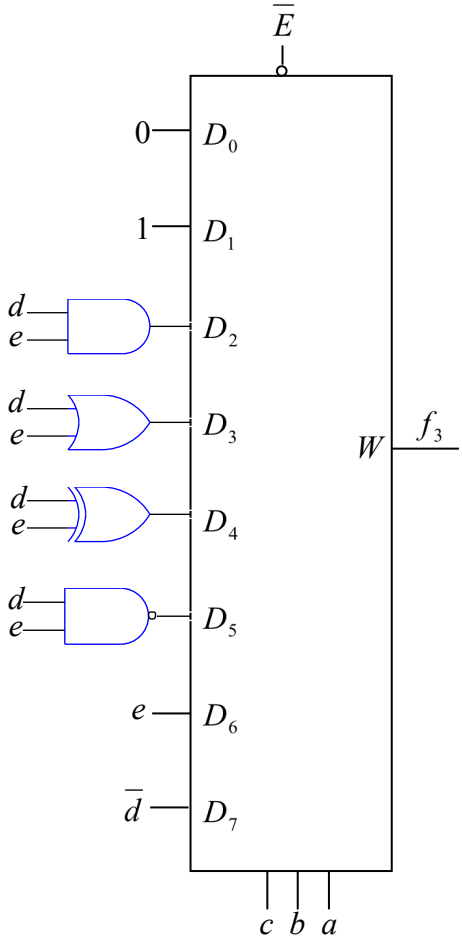
Ex. funcţia f_2 definită prin tabelul şi având reprezentarea în figura 3.24

Observaţie: Pentru a uşura determinarea funcţiilor de intrare din tabelul de definiţie al funcţiei logice este recomandabil ca variabila reziduală să fie cea mai puţin semnificativă.

În general prin utilizarea circuitelor multiplexoare se reduce numărul variabilelor de care depinde funcţia logică. Astfel, daca funcţia de implementat depinde de "n" variabile, prin utilizarea unui multiplexor cu 2^m intrari de date (m<n) funcţiile care trebuiesc implementate cu porţi logice depind de "n-m" variabile. Implementarea funcţiei de "n" variabile se reduce în acest fel la implementarea a 2^m functii de "n-m" variabile.

Pentru exemplificare se consideră funcţia f_3 de cinci variabile la implementarea căreia se foloseşte un multiplexor cu 8 intrări de date (fig.3.25).

a	b	c	d	e	f_3	
0	0	0	0	0	0	
0	0	0	0	1	0	$D_0 = 0$
0	0	0	1	0	0	
0	0	0	1	1	0	
0	0	1	0	0	1	
0	0	1	0	1	1	$D_1 = 1$
0	0	1	1	0	1	
0	0	1	1	1	1	
0	1	0	0	0	0	
0	1	0	0	1	0	$D_2 = d.e$
0	1	0	1	0	0	
0	1	0	1	1	1	
0	1	1	0	0	0	
0	1	1	0	1	1	$D_3 = d + e$
0	1	1	1	0	1	
0	1	1	1	1	1	
1	0	0	0	0	0	
1	0	0	0	1	1	$D_4 = d \oplus e$
1	0	0	1	0	1	
1	0	0	1	1	0	
1	0	1	0	0	1	$D_5 = \overline{d\,e}$



1	0	1	0	1	1	
1	0	1	1	0	1	
1	0	1	1	1	0	
1	1	0	0	0	0	
1	1	0	0	1	1	
1	1	0	1	0	0	$D_6 = e$
1	1	0	1	1	1	
1	1	1	0	0	1	
1	1	1	0	1	1	
1	1	1	1	0	0	$D_7 = \overline{d}$
1	1	1	1	1	0	

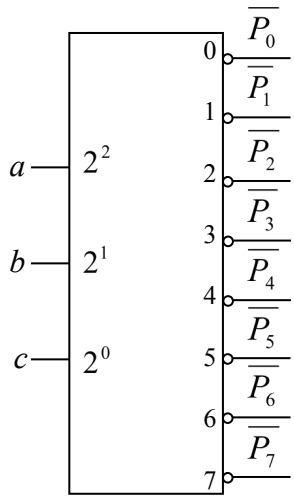
Figura 3.25

3.2.4 Decodificatorul si demultiplexorul

Decodificatorul este un circuit logic combinational, integrat pe scară medie, obţinut printr-o extensie de tip paralel a unor porţi logice şi are "n" intrări de adresare şi 2^n iesiri. Din punct de vedere funcţional decodificatorul activează una din cele 2^n ieşiri ale sale, funcţie de codul aplicat la cele "n" intrări.

Din punct de vedere constructiv se produc mai multe tipuri de circuite decodificatoare dintre care amintim decoficatorul zecimal cu 4 intrări şi 10 ieşiri, decodificatorul binar dublu cu patru ieşiri şi două intrări de adresare comune, decodificatorul binar cu 16 ieşiri, decodificator pentru comanda dispozitivelor de afisaj cu 7 segmente. În fig. 3.26 este dată schema desfăşurată, schema bloc şi tabelul de funcţionare ale decodificatorului binar cu 3 intrări şi 8 ieşiri.

INTRARI			IESIRI							
a	b	c	$\overline{P_0}$	$\overline{P_1}$	$\overline{P_2}$	$\overline{P_3}$	$\overline{P_4}$	$\overline{P_5}$	$\overline{P_6}$	$\overline{P_7}$
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0



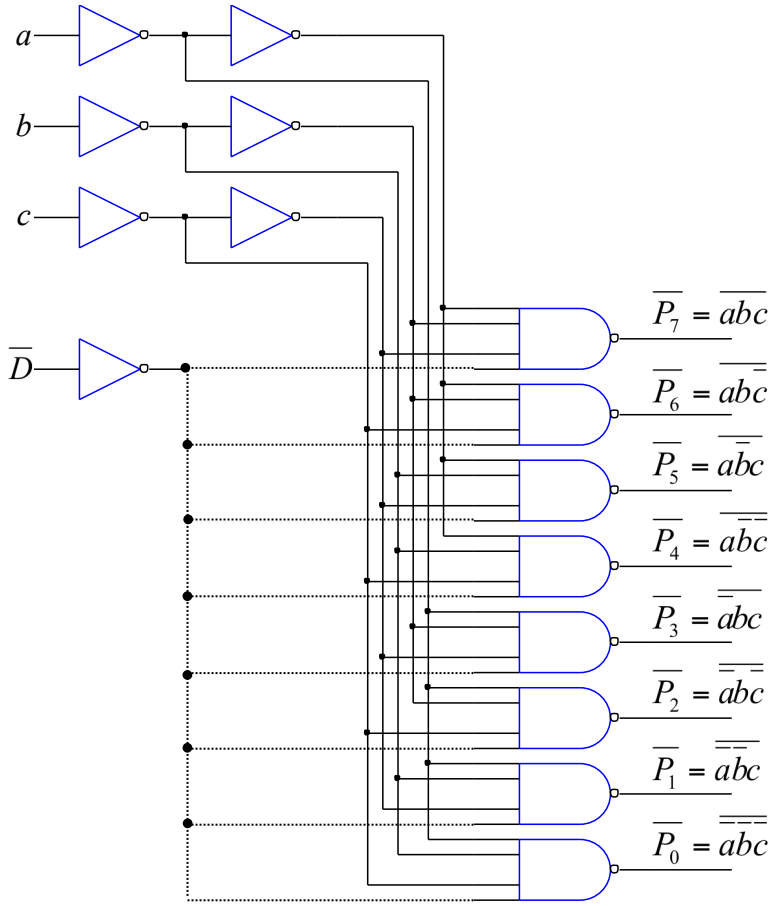


Figura 3.26

Circuitul decodificator din fig.3.26 se transformă în circuit demultiplexor dacă schema se completează cu traseele cu linie întreruptă.

Din punct de vedere funcţional demultiplexorul asigură transmiterea semnalului aplicat la intrarea sa prin una din cele 2^n ieşiri, ieşire selectată prin codul aplicat la cele "n" intrări de adresare.

Tabelul de funcţionare şi schema bloc ale demultiplexorului a cărui schemă logică desfăşurată este dată în fig. 3.26 (inclusiv traseul cu linie intrerupta) sunt prezentate in fig.3.27.

INTRARI				IESIRI							
a	b	c	$\overline{D}$	0	1	2	3	4	5	6	7
0	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1
0	1	0	0	1	1	0	1	1	1	1	1
0	1	1	0	1	1	1	0	1	1	1	1
1	0	0	0	1	1	1	1	0	1	1	1

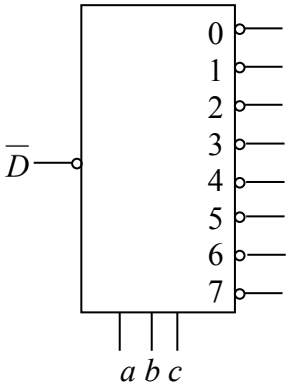


Figura 3.27

1	0	1	0	1	1	1	1	1	0	1	1
1	1	0	0	1	1	1	1	1	1	0	1
1	1	1	0	1	1	1	1	1	1	1	0
*	*	*	1	1	1	1	1	1	1	1	1

Din tabel se observă că demultiplexorul funcţionează ca un decodificator cu 8 ieşiri dacă la intrarea de date $\overline{D}$ se aplică semnal logic zero, deci practic un demultiplexor poate fi folosit ca decodificator. Si reciproca este valabilă, adică un decodificator poate fi utilizat ca demultiplexor. În cazul de faţă, decodificatorul din fig. 3.26 poate fi utilizat ca demultiplexor cu patru ieşiri (primele patru) dacă intrarea cu ponderea cea mai mare se consideră ca fiind intrare de date.

Multiplexorul şi demultiplexorul pot fi utilizate pentru transmiterea mai multor semnale pe o singură linie.

Circuitele decodificatoare sunt foarte utile la implementarea sistemelor de funcţii. Exemplu: să se sintetizeze un circuit logic combinaţional care să facă conversia din cod binar natural în cod Gray de trei variabile folosind un circuit decodificator cu 8 ieşiri.

	a	b	c	f_1	f_2	f_3
P_0	0	0	0	0	0	0
P_1	0	0	1	0	0	1
P_2	0	1	0	0	1	1
P_3	0	1	1	0	1	0
P_4	1	0	0	1	1	0
P_5	1	0	1	1	1	1
P_6	1	1	0	1	0	1
P_7	1	1	1	1	0	0

$$f_1 = P_4 + P_5 + P_6 + P_7$$
$$f_2 = P_2 + P_3 + P_4 + P_5$$
$$f_3 = P_1 + P_2 + P_5 + P_6$$

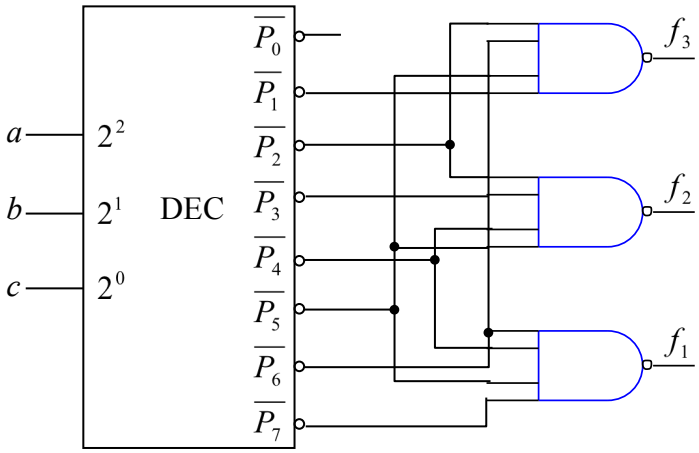


Figura 3.28