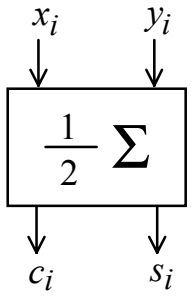
5 CIRCUITE COMBINAȚIONALE

5.1 Sisteme de funcții binare

Un circuit combinațional este o structură de funcții binare, implementate într-un circuit ale cărui ieşiri depind numai de valorile logice aplicate pe intrări. În acest capitol ne ocupăm de circuitele combinaționale de complexitate mică şi medie, şi care nu sunt programabile. Memoriile ROM şi structurile programabile vor fi prezentate într-un capitol separat.

Dacă structura combinațională are două sau mai multe ieşiri, atunci avem de-a face cu un **sistem de funcții binare**. Ne vom ocupa pentru început de sinteza unui **sumator** pentru operanzi de un bit. Sumatorul este un circuit care realizează **suma aritmetică** a operanzilor aplicați pe intrări. Atragem atenția să nu se confunde această operație cu operația logică SAU, numită de multe ori şi **sumă logică** (confuzia poate apare datorită aceleiaşi notații: simbolul +).

Sumatorul este format din două **semisumatoare**. Semisumatorul de rang i pentru operanzi de un bit are două intrări la care se aplică operanzii, notate aici cu x_i şi y_i, şi două ieşiri: suma, notată cu s_i, şi transportul (carry), notat cu c_i. Reprezentarea grafică şi tabelul de adevăr sunt date în figura 5.1. Sinteza schemei logice a circuitului este dată în figura 5.2. Cele două funcții binare s_i şi c_i s-au minimizat cu ajutorul diagramelor Veitch-Karnaugh pentru funcții de două variabile.



x_i	y_i	c_i	s_i
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Fig. 5.1 *Schema bloc şi tabelul de adevăr*

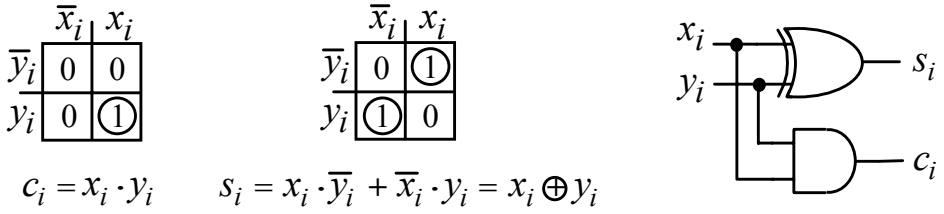
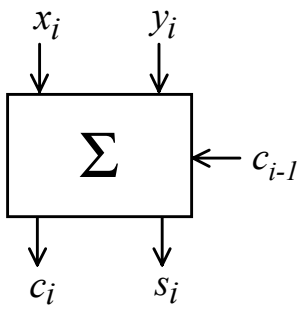


Fig. 5.2 *Sinteza schemei logice cu număr minim de porți*



x_i	y_i	c_{i-1}	c_i	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Fig. 5.3 *Schema bloc şi tabelul de adevăr*

Reprezentarea sumatorului complet pentru operanzi de 1 bit şi tabelul lui de adevăr sunt date în figura 5.3. Observăm că există o intrare suplimentară, notată cu c_{i-1}, care reprezintă transportul de la sumatorul de rang i-1. Figura 5.4 arată modul în care s-a făcut minimizarea funcțiilor s_i şi c_i. Cu ajutorul ecuațiilor obținute s-a construit schema logică din partea stângă a figurii 5.5. Şi totuşi, dacă analizăm schema din partea dreaptă a figurii, constatăm că face acelaşi lucru, dar cu număr mai mic de porți logice.

Fig. 5.4 *Minimizarea funcțiilor binare* c_i *şi* s_i

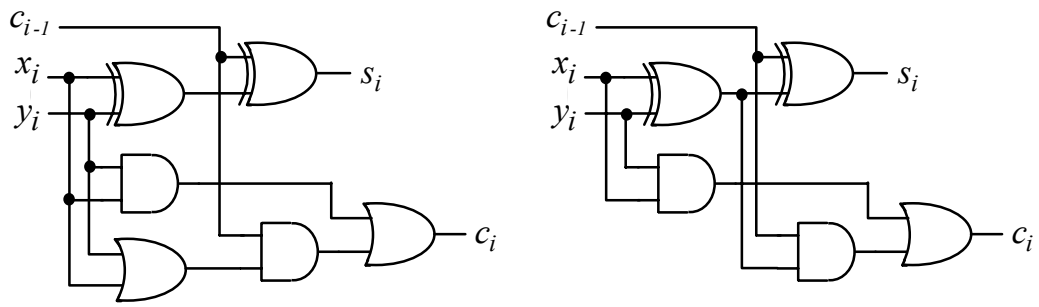


Fig. 5.5 *Schema logică a sumatorului complet de 1 bit*

Ce se întâmplă? Am greşit undeva? Într-un fel, da. Minimizarea separată a funcţiilor s_i şi c_i este corectă, dar aici avem un sistem de două funcţii binare, care ar trebui minimizat global. Ar trebui să încercăm să găsim unele porţi logice comune pentru cele două funcţii.

Concluzia este că forma minimă a unui sistem de funcţii binare poate să difere de suma formelor minime ale fiecărei funcţii în parte, dacă anumite porţi din structură pot fi folosite în comun pentru implementarea mai multor funcţii.

Există o metodă de sinteză a sistemelor de funcţii binare. Algoritmul presupune parcurgerea următoarelor etape:

1. se stabilesc implicanţii primi ai funcţiei rezultate prin intersecţia tuturor celor *n* funcţii din sistem şi apoi a tuturor funcţiilor rezultate prin intersecţia a *n*-1 funcţii din sistem, şi aşa mai departe, până ce obţinem implicanţii primi ai fiecărei funcţii luate separat. Fiecare implicant prim este luat în considerare o singură dată.
2. se construieşte un tabel al implicanţilor primi şi se stabileşte suma minimală neredundantă care acoperă complet toţi mintermenii fiecărei funcţii din sistem.

Exemplul 5.1

Să se implementeze cu număr minim de porţi sistemul:

$$f_1 = P_2 + P_4 + P_{10} + P_{11} + P_{12} + P_{13}$$
$$f_2 = P_4 + P_5 + P_{10} + P_{11} + P_{13}$$
$$f_3 = P_1 + P_2 + P_3 + P_{10} + P_{11} + P_{12}$$

Se stabilesc implicanţii primi ai funcţiilor rezultate prin toate intersecţiile posibile. Figura 5.6 indică toţi aceşti implicanţi primi prin vizualizarea lor pe diagrame Veitch-Karnaugh. Fiecare implicant prim este luat în considerare o singură dată. Construim în continuare un tabel al implicanţilor primi, asemănător cu cel construit la minimizarea prin metoda Quine-McCluskey. Coloanele sunt definite de mintermenii funcţiilor din sistem, iar liniile, de toţi implicanţii primi găsiţi mai înainte.

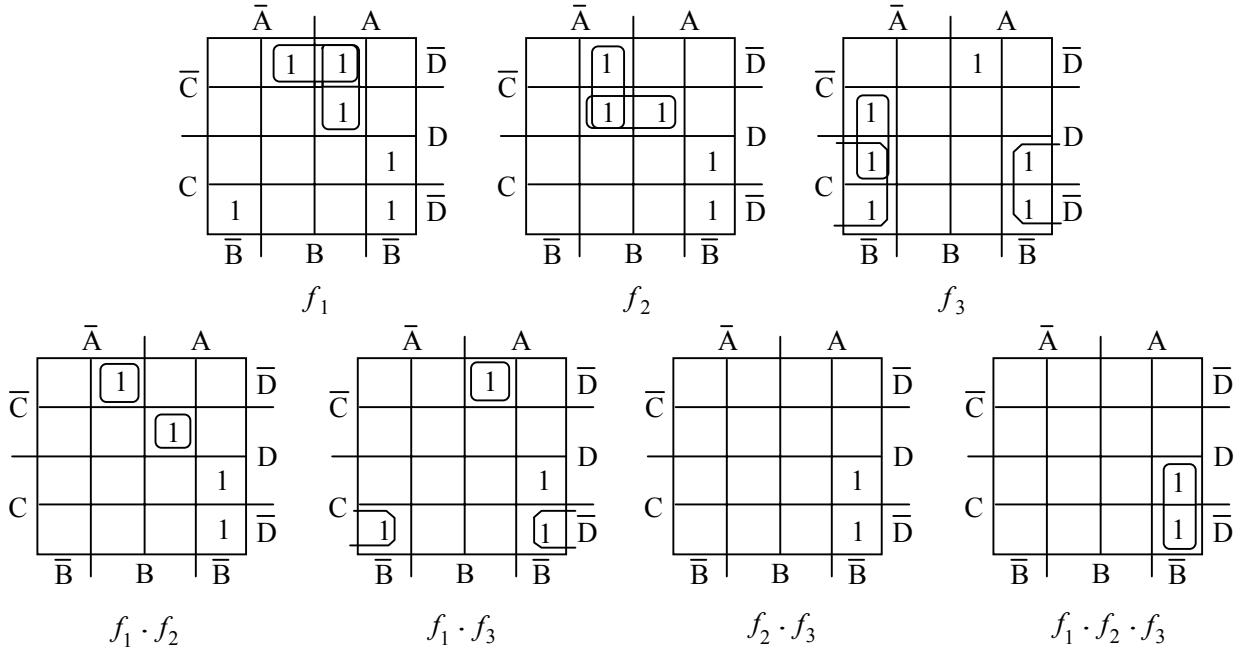


Fig. 5.6 *Stabilirea implicanţilor primi*

Mintermeni / Implicanți primi	f_1						f_2					f_3					
	P_2	P_4	P_{10}	P_{11}	P_{12}	P_{13}	P_4	P_5	P_{10}	P_{11}	P_{13}	P_1	P_2	P_3	P_{10}	P_{11}	P_{12}
$A\overline{B}C$			√	√					√	√					√	√	
$\overline{A}B\overline{C}\overline{D}$		√					√										
$AB\overline{C}D$						√					√						
$AB\overline{C}\overline{D}$					√												√
$\overline{B}C\overline{D}$	√		√										√		√		
$\overline{B}C$													√	√	√	√	
$B\overline{C}\overline{D}$		√			√												
$AB\overline{C}$					√	√											
$\overline{A}B\overline{C}$							√	√									
$B\overline{C}D$								√			√						
$\overline{A}\overline{B}D$												√		√			

Fig. 5.7 *Tabelul implicanților primi*

Se urmăresc coloanele marcate o singură dată şi se stabilesc implicanții primi esențiali: $\overline{B}C\overline{D}$ şi $A\overline{B}C$ pentru f_1, $A\overline{B}C$ pentru f_2, $\overline{A}\overline{B}D$ şi $AB\overline{C}\overline{D}$ pentru f_3. Reuniunea lor formează setul implicanților primi esențiali pentru sistemul de funcții.

Au rămas de acoperit numai termenii P_4 şi P_{13} din f_1 şi termenii P_4, P_5 şi P_{13} din f_2. Termenii din f_3 sunt complet acoperiți de implicanții primi esențiali.

Construim în continuare tabelul redus al implicanților primi, trecând pe linii numai implicanții primi conținuți de mintermenii ce trebuie să fie acoperiți. Acest tabel este reprezentat în figura 5.8. Folosind notațiile din tabel, scriem condiția de acoperire completă a sistemului de funcții: $(a+c)\cdot(b+d)\cdot(a+e)\cdot(e+f)\cdot(b+f)=1$. Efectuăm calculele şi rezultă: $(a+ce)\cdot(b+d)\cdot(eb+f)=1$, sau $abe+bce+abf+adf+cdef=1$.

Dintre toți aceşti termeni, numai *bce* şi *adf* conțin implicanți care au un singur marcaj pe linie. Oricare dintre ei formează un set de implicanți primi neesențiali care, împreună cu cei esențiali, acoperă sistemul de funcții. Dacă alegem *adf*, rezultă: $f_1 = AB\overline{C} + \overline{B}C\overline{D} + A\overline{B}C + \overline{A}B\overline{C}\overline{D}$, $f_2 = A\overline{B}C + B\overline{C}D + \overline{A}B\overline{C}\overline{D}$, iar ultima funcție este $f_3 = \overline{A}\overline{B}D + \overline{B}C\overline{D} + A\overline{B}C + AB\overline{C}\overline{D}$. Circuitul optim este reprezentat în figura 5.8.

Mintermeni / Implicanți primi	f_1		f_2			Notație
	P_4	P_{13}	P_4	P_5	P_{13}	
$\overline{A}B\overline{C}\overline{D}$	√		√			*a*
$AB\overline{C}D$		√			√	*b*
$B\overline{C}\overline{D}$	√					*c*
$AB\overline{C}$		√				*d*
$\overline{A}B\overline{C}$			√	√		*e*
$B\overline{C}D$				√	√	*f*

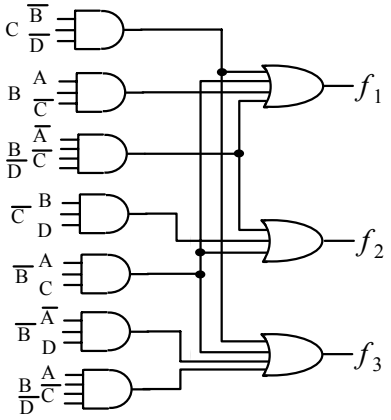


Fig. 5.8 *Tabelul redus al implicanților primi şi circuitul optim*

5.2 Circuite de uz general

Circuitele de uz general sunt structuri combinaționale care pot fi definite recursiv şi care pot atinge dimensiuni mari, limitate numai de restricții tehnologice. De aceea, ele nu vor mai fi văzute ca structuri de porți logice, ci mai degrabă ca structuri care implementează funcții complexe: **decodificarea**, care presupune recunoaşterea unei configurații binare, **demultiplexarea**, care permite distribuirea comandată a unor semnale, **multiplexarea**, care realizează selecția comandată a unui semnal şi **codificarea**, o funcție inversă decodificării.

Decodificatorul (notat prescurtat DCD) este un circuit care identifică un cod de intrare prin activarea unei singure linii de ieşire. Dacă circuitul are *n* variabile binare de intrare, atunci numărul liniilor de ieşire este 2^n. Figura 5.9 arată structura circuitului pentru $n = 2$. Intrările porților ŞI-NU sunt astfel conectate încât la ieşiri se obțin complementele tuturor mintermenilor funcției de două variabile, A şi B. Perechile de inversoare pe intrări sunt necesare pentru ca fiecare intrare în circuitul integrat să fie văzută ca intrare într-o singură poartă logică. În acest fel se respectă caracteristicile de intrare standard pentru familia logică respectivă.

Demultiplexorul (notat prescurtat DMUX) este un circuit construit pe structura decodificatorului, care permite transmiterea datelor de pe o singură cale de intrare pe una dintre cele 2^n căi de ieşire. Selecția liniilor de ieşire se face prin aplicarea unui cod binar pe *n* linii de intrare, care devin acum intrări de selecție. Structura demultiplexorului pentru $n = 2$ este dată în figura 5.10. Intrarea $\overline{\text{ENABLE}}$ (permite) este activă pe 0 logic şi permite accesul datelor de la intrarea $\overline{\text{I}}$ la una din ieşirile stabilite prin codul binar aplicat pe intrările de selecție A şi B.

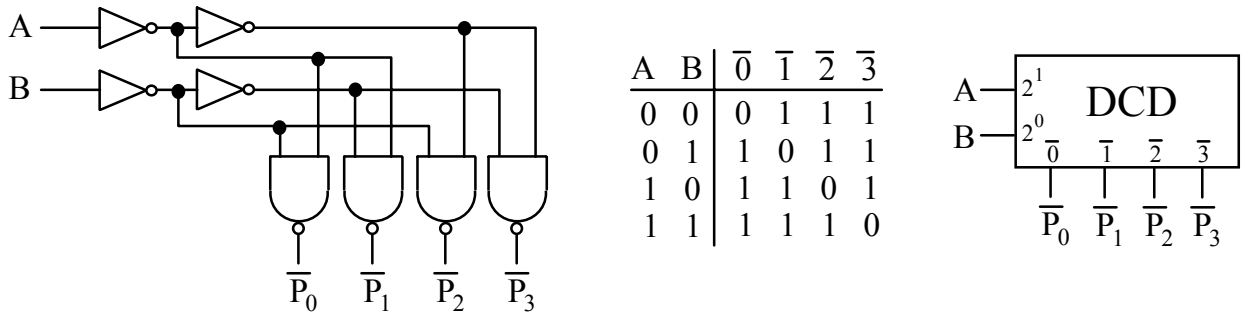


Fig. 5.9 *Structura, tabelul de adevăr şi reprezentarea decodificatorului pentru n=2*

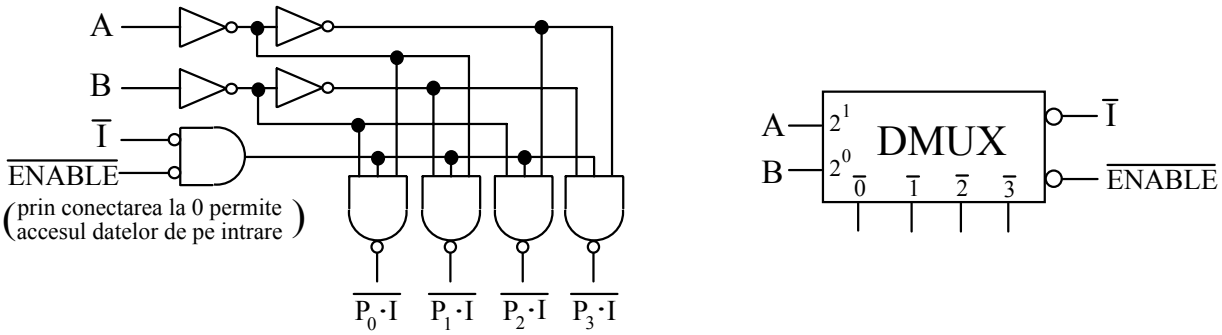


Fig. 5.10 *Structura şi reprezentarea demultiplexorului pentru n=2*

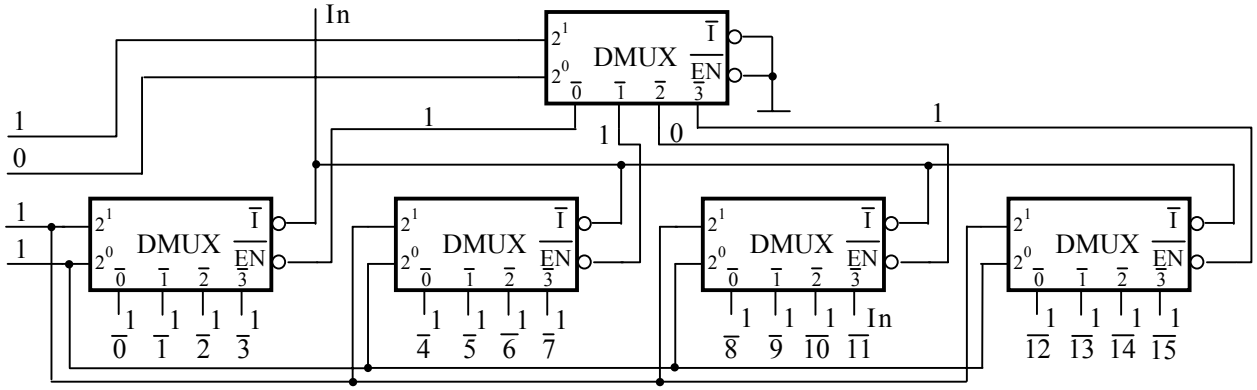


Fig. 5.11 *Extinderea capacității de demultiplexare pentru n=2*

Se construiesc circuite integrate cu funcție de demultiplexor pentru n = 1, 2, 3 sau 4. Pentru un număr n mai mare, implementarea structurii într-un singur circuit integrat nu mai este economică, pentru că numărul mare de terminale ar impune utilizarea unei capsule de microprocesor pentru o simplă funcție definită recursiv.

Soluția constă în utilizarea mai multor circuite integrate pentru a face o **extindere a capacității de decodificare/demultiplexare**. În cazul general, pentru extinderea capacității de decodificare/demultiplexare de la 2^n linii de ieşire la $\left(2^n\right)^2$ linii de ieşire se folosesc 2^n+1 demultiplexoare. Figura 5.11 exemplifică această extindere pentru n = 2. Aplicarea numărului binar 1011 pe intrările de selecție determină transferul datelor de la intrarea "In" la ieşirea $\overline{11}$ a circuitului. Biții mai semnificativi 10 selectează ieşirea $\overline{2}$ a decodificatorului de pe primul nivel, activând intrarea $\overline{EN}$ a celui de-al treilea demultiplexor de pe al doilea nivel, iar biții mai puțin semnificativi 11 selectează ieşirea $\overline{3}$ a acestui demultiplexor. Celelalte ieşiri au valoarea logică 1.

Multiplexorul (notat prescurtat MUX) realizează funcția inversă demultiplexorului, adică permite transmiterea datelor de la una din cele 2^n căi de intrare la o singură cale de ieşire. Selecția unei anumite intrări de date se face prin aplicarea unui cod binar pe n linii de intrare, linii care sunt intrările de selecție. Structura multiplexorului pentru n = 2 este dată în figura 5.12. Activarea intrării $\overline{E}$ (Enable) pe 0 logic permite accesul datelor de la una din cele 4 intrări de date I_i, i = 0, 1, 2 sau 3 la ieşirile $\overline{W}$ sau W. Selecția intrării se face printr-un cod binar aplicat pe intrările de selecție A şi B.

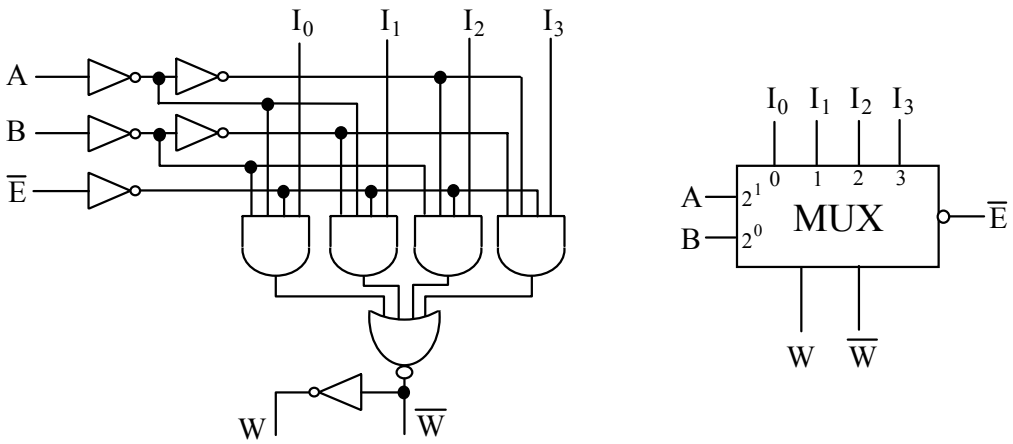


Fig. 5.12 *Structura şi reprezentarea multiplexorului pentru n=2*

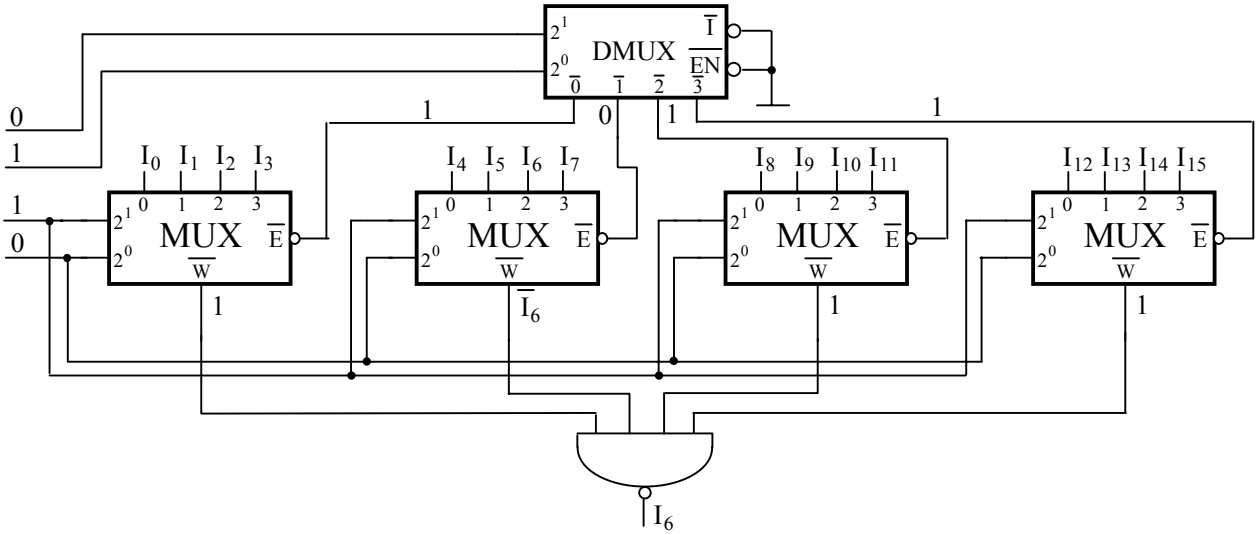


Fig. 5.13 *Extinderea capacității de multiplexare pentru n=2*

Extinderea capacității de multiplexare de la 2^n linii de intrare la $\left(2^n\right)^2$ linii de intrare se face prin utilizarea unui număr de 2^n multiplexoare şi a unui decodificator cu 2^n linii de ieşire care validează accesul la multiplexoare. Figura 5.13 exemplifică această extindere pentru n = 2. Aplicarea numărului binar 0110 pe intrările de selecție conectează intrarea I_6 la ieşirea circuitului. Biții mai semnificativi 01 selectează ieşirea $\overline{1}$ a decodificatorului de pe primul nivel, activând intrarea $\overline{E}$ a celui de-al doilea multiplexor de pe al doilea nivel, iar biții mai puțin semnificativi 10 selectează intrarea $\overline{2}$ a acestui multiplexor.

Codificatorul este un circuit care asociază unui semnal de un bit aplicat pe una dintre cele 2^n intrări un cod binar de n biți la ieşiri. El realizează într-un anumit sens funcția inversă decodificării, cu următoarele observații:

- trebuie semnalizată situația în care nu există nici un semnal aplicat pe intrare (toate intrările au valoarea logică 0)
- trebuie soluționată situația conflictuală în care două sau mai multe intrări ale circuitului sunt activate simultan; pentru a genera un singur cod la ieşire trebuie impus un criteriu de alegere a intrării active.

Prima condiție se realizează prin introducerea unei ieşiri suplimentare, notată cu Z, care indică dacă pe toate intrările avem 0 logic. A doua condiție justifică denumirea circuitului, care nu este un codificator simplu, ci un **codificator prioritar**, în sensul că intrarea activă de rang maxim are prioritate (figura 5.14 reprezintă cazul n = 3).

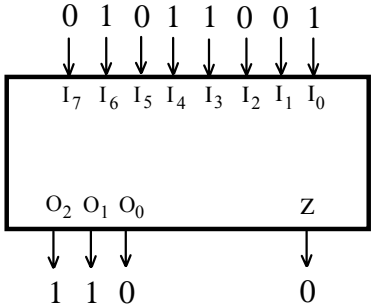


Fig. 5.14 *Codificatorul prioritar cu 8 intrări*

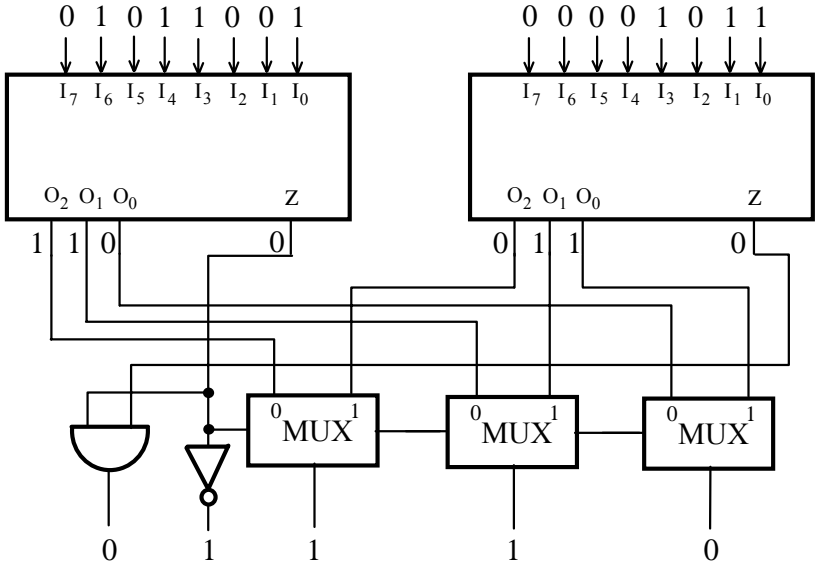


Fig. 5.15 *Dublarea numărului de intrări la codificatorul prioritar*

Există şi o interpretare numerică a funcționării codificatorului prioritar. Circuitul calculează partea întregă a logaritmului în bază 2 din numărul de opt biți aplicat la intrare. Dacă toate intrările sunt pe 0 logic, atunci operația nu are sens şi ieşirea Z indică acest lucru prin $Z = 1$. Desenul din figura 5.14 ilustrează şi un exemplu numeric: $\lceil \log_2 89 \rceil = 6$.

Extinderea capacității de codificare prioritară de la 2^n linii de intrare la $\left(2^n\right)^2$ linii de intrare se face prin utilizarea unui număr de 2^n codificatoare şi a unor circuite combinaționale suplimentare. Figura 5.15 exemplifică cum se face dublarea numărului de intrări la codificatorul prioritar cu 8 linii de intrare. Nu este singura soluție. Dacă se utilizează de exemplu circuite care dispun şi de intrarea de activare ENABLE, accesul datelor la ieşiri poate fi controlat şi logica externă s-ar putea simplifica (se pot utiliza porți SAU cu 2 intrări în loc de multiplexoare).

Toate aceste circuite combinaționale descrise aici se pot utiliza la implementarea funcțiilor binare, fără a mai fi necesare operații de minimizare.

Exemplul 5.2

Să se implementeze următoarea funcție binară folosind demultiplexor cu 3 intrări de selecție şi multiplexor cu 3 intrări de selecție:

$$f = P_0 + P_1 + P_2 + P_6 + P_7 = \overline{\overline{P}_0 \cdot \overline{P}_1 \cdot \overline{P}_2 \cdot \overline{P}_6 \cdot \overline{P}_7}$$

Prin conectarea la 0 logic a intrărilor $\overline{\mathrm{I}}$ şi $\overline{\mathrm{E}}$ circuitul este transformat într-un decodificator. La ieşirile decodificatorului sunt disponibile complementele tuturor mintermenilor funcției. De aceea se aplică legile lui De Morgan şi în expresia funcției sunt puşi în evidență termenii $\overline{P}_i$. Acum avem două posibilități de implementare: fie conectăm ieşirile $\overline{0}$, $\overline{1}$, $\overline{2}$, $\overline{6}$ şi $\overline{7}$ la intrările unei porți ŞI-NU, conform expresiei de mai sus, fie implementăm complementul funcției, după care îl negăm, adică conectăm celelalte ieşiri la intrările unei porți ŞI. Pentru implementarea cu multiplexor, fiecare intrare corespunde unui mintermen. Deci intrările care corespund mintermenilor existenți se conectează la 1 logic, iar celelate la 0 logic. Schemele din figura 5.16 prezintă aceste implementări.

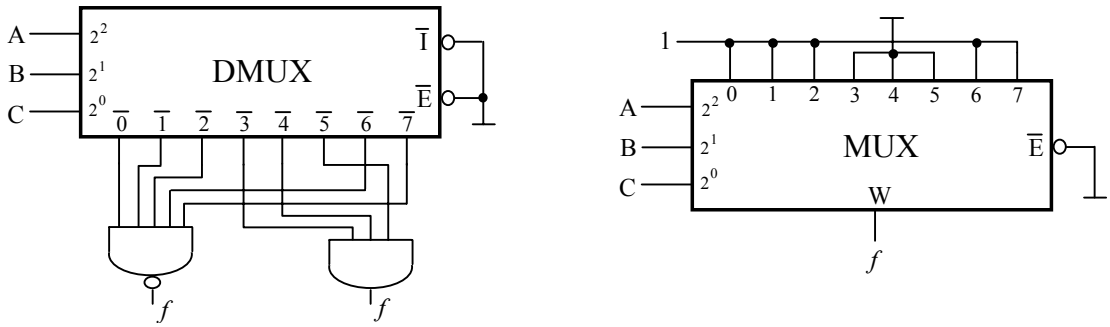


Fig. 5.16 *Implementare cu MUX/DMUX cu 3 intrări de selecţie*

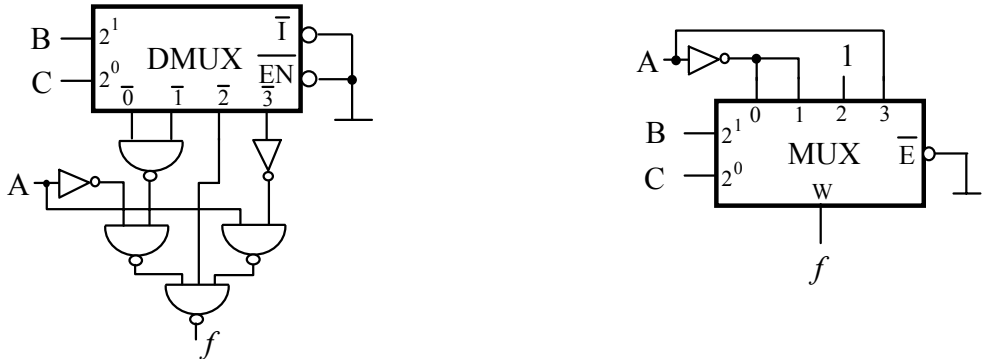


Fig. 5.17 *Implementare cu MUX/DMUX cu 2 intrări de selecţie*

Să se implementeze aceeaşi funcţie binară folosind demultiplexor cu două intrări de selecţie şi multiplexor cu două intrări de selecţie.

De data aceasta nu mai avem la dispoziţie orice termen produs fundamental al funcţiei. Pe cei pe care îi avem, îi vom pune în evidenţă construind produse de două variabile, la care vom adăuga alte circuite combinaţionale, conform expresiei algebrice care rezultă în urma transformării. Aplicând variabilele *B* şi *C* pe intrările de selecţie, putem scrie următoarele relaţii:

$$f = P_0 + P_1 + P_2 + P_6 + P_7 = \overline{A}\left(\overline{B}\,\overline{C} + \overline{B}C + B\overline{C}\right) + A\left(B\overline{C} + BC\right) =$$

$$= \overline{A}\left(P_0' + P_1' + P_2'\right) + A\left(P_2' + P_3'\right) = \overline{A}\left(P_0' + P_1'\right) + P_2' + AP_3' = \overline{\overline{A} \cdot \overline{P_0'} \cdot \overline{P_1'} \cdot \overline{P_2'} \cdot \overline{A \cdot \overline{P_3'}}}$$

Notaţia P_i' indică că avem de-a face cu alţi termeni produs, de numai două variabile. Evident că aceste variabile pot fi alese arbitrar, deci problema are multe soluţii.

Mai trebuie să facem o observaţie importantă. La implementările cu porţi, ne-am obişnuit ca variabila *A* să fie cea mai semnificativă. Am păstrat şi aici această convenţie, dar trebuie să reţinem că la majoritatea circuitelor reale lucrurile stau exact invers. Consultarea datelor de catalog este foarte importantă, atunci când folosim circuite integrate reale.

În concluzie, circuitele folosite implementează toţi termenii produs fundamentali ai unei funcţii cu număr de variabile mai mic sau egal cu numărul de intrări de selecţie *n*. Din acest motiv, implementarea funcţiilor binare nu necesită operaţii de minimizare, ci numai alegerea corectă a conexiunilor. Dacă numărul de variabile ale funcţiei este mai mic sau egal cu numărul de intrări de selecţie *n*, atunci implementarea se face direct. În caz contrar mai sunt necesare o serie de transformări algebrice pentru a găsi o altă structură adiţională formată de obicei din porţi, care să completeze lipsa intrărilor de selecţie de la multiplexor sau demultiplexor.

5.3 Circuite aritmetice

Circuitele aritmetice mai sunt numite **circuite specializate**, fiind clasificate în **circuite cu funcții dedicate** şi **circuite cu funcții selectabile** ([Ştefan, 1993]). Dintre circuitele cu funcții dedicate vom discuta despre **sumator**, **comparator**, **circuite de deplasare** şi **multiplicator**, iar dintre cele cu funcții selectabile vom prezenta **unitatea logico-aritmetică**.

Sumatorul calculează suma aritmetică a unor operanzi binari. Dacă operanzii au *n* biți, atunci circuitul se compune din *n* sumatoare complete de 1 bit. Sumatorul complet pentru operanzi de 1 bit a fost prezentat la începutul capitolului. Dezavantajul acestor circuite, numite şi **sumatoare cu transport înlănțuit**, este timpul mare de execuție a operației, pentru că semnalele de depăşire trebuie să se propage prin toate cele *n* sumatoare complete. Circuitul integrat din figura 5.18 este un sumator pentru cuvinte de 4 biți.

Pentru a creşte viteza de execuție a operației s-a implementat o tehnică de **propagare rapidă a transportului** în **sumatorul cu transport anticipat**. Dacă x_i şi y_i sunt cei doi operanzi de 1 bit pentru celula de sumare de rang *i*, atunci observăm că:

- semnalul de generare a transportului este $g_i = x_i \cdot y_i$, adică transportul este generat dacă ambii operanzi au valoarea logică 1
- semnalul de propagare a transportului este $p_i = x_i + y_i$, adică propagarea transportului are loc dacă cel puțin unul din operanzi are valoarea logică 1
- ieşirea de carry a etajului *i* devine: $c_{i+1} = g_i + p_i \cdot c_i$.

Pentru toate cele 4 celule de sumare obținem:

$$c_1 = g_0 + p_0 \cdot c_0$$

$$c_2 = g_1 + p_1 \cdot c_1 = g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot c_0$$

$$c_3 = g_2 + p_2 \cdot c_2 = g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot c_0$$

$$c_4 = g_3 + p_3 \cdot c_3 = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0$$

Fiecare ecuație este implementată folosind numai 3 nivele de logică, deci timpul de propagare final este minim. Propagarea transportului s-a transformat într-un calcul anticipat, fiecare transport se poate calcula în paralel, fără a mai aştepta calcule de la rangurile inferioare.

Aceste soluții reprezintă totuşi soluții limită. Prima este minimală ca dimensiune, iar a doua maximală ca viteză. Ambele soluții par a fi departe de optim. Un prim pas către o soluție optimă constă în îmbinarea celor două soluții. Se obține **sumatorul cu transport anticipat pe blocuri** ([Ştefan, 2000]).

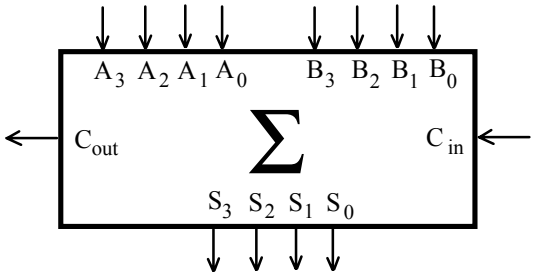


Fig. 5.18 *Sumator binar de 4 biți*

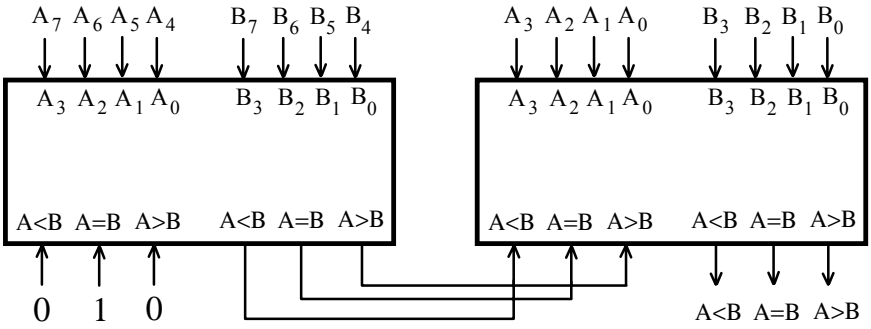


Fig. 5.19 *Comparator de 8 biţi*

Comparatorul este un circuit integrat care realizează compararea a două cuvinte de 4 biţi. Cele două cuvinte notate cu *A* şi *B* se aplică pe intrări, rezultatul comparării fiind furnizat la cele 3 ieşiri, $A < B$, $A = B$, $A > B$. Există şi 3 intrări, denumite la fel, care sunt necesare pentru cascadarea mai multor circuite, în scopul comparării unor operanzi mai mari. Figura 5.19 exemplifică extinderea capacităţii de comparare la cuvinte de 8 biţi, folosind comparatoare pe 4 biţi. Acest transport de informaţie se face de la rangurile superioare spre cele inferioare, spre deosebire de sumator, unde depăşirea se propagă spre rangurile binare mai mari.

Funcţia de deplasare spre stânga sau spre dreapta se poate realiza fie combinaţional, fie secvenţial. Soluţia combinaţională presupune utilizarea unor circuite de multiplexare. Circuitul din figura 5.20 este un caz particular de **circuit de deplasare** spre dreapta, în care nici unul din biţii de intrare nu se pierde. Avem de-a face de fapt cu o **operaţie de rotire**, iar circuitul se mai numeşte **rotitor combinaţional**. Conexiunile intrărilor sunt făcute în aşa fel încât pentru secvenţa s_1s_2 = 00, 01, 10, 11 aplicată pe intrările de selecţie, vom avea următoarea secvenţă de ieşire: $I_3I_2I_1I_0$, $I_0I_3I_2I_1$, $I_1I_0I_3I_2$ şi $I_2I_1I_0I_3$.

Circuitele de deplasare spre stânga permit realizarea multiplicării cu puterile întregi ale lui 2, iar cele de deplasare spre dreapta permit împărţirea cu puterile întregi ale lui 2. Pentru a realiza orice înmulţire sunt necesare însă circuite mai elaborate. Există soluţia combinaţională, care oferă varianta cea mai rapidă, soluţii secvenţiale, sau soluţii programate. Soluţia combinaţională foloseşte **multiplicatorul elementar** care realizează înmulţirea a două cuvinte de un bit. Structura internă şi simbolul logic pentru acest circuit sunt prezentate în figura 5.21. Poarta logică ŞI realizează înmulţirea biţilor *A* şi *B*, iar sumatorul complet de 1 bit adună produsul cu cel realizat de bitul mai puţin semnificativ.

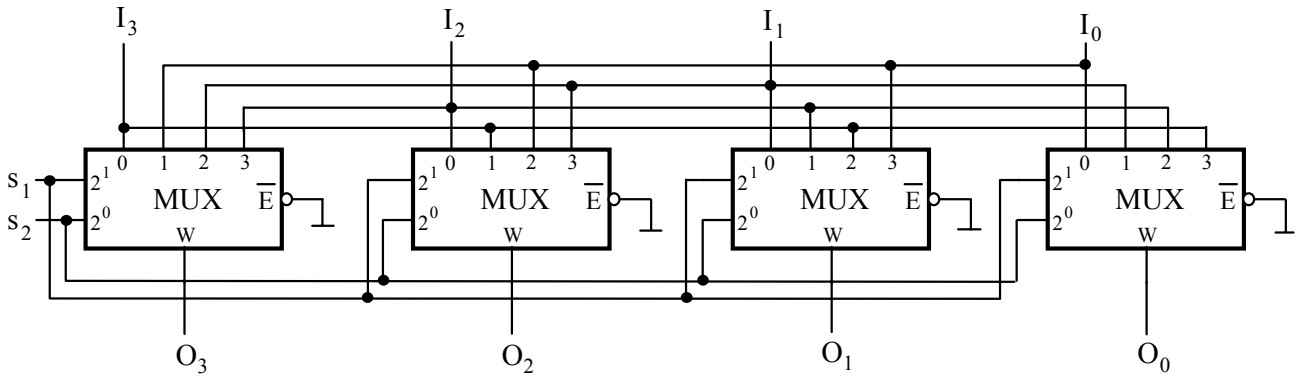


Fig. 5.20 *Rotitorul combinaţional*

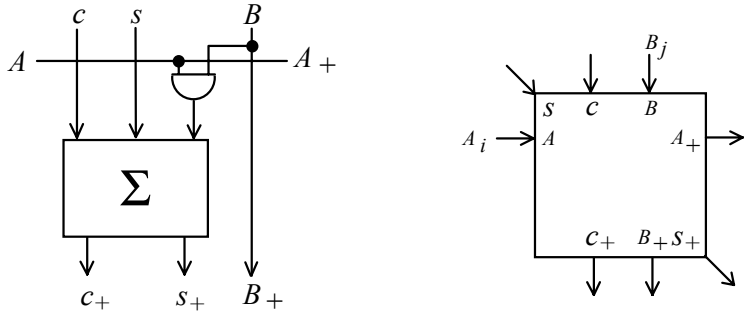


Fig. 5.21 *Structura internă şi simbolul logic pentru multiplicatorul elementar*

Pentru a realiza un **circuit de înmulțire** combinațional pentru cuvinte de n biți trebuie să interconectăm n^2 multiplicatoare elementare într-o matrice bidimensională, sugerată de algoritmul de înmulțire (cazul $n = 4$):

				A_3	A_2	A_1	A_0
				B_3	B_2	B_1	B_0
				A_3B_0	A_2B_0	A_1B_0	A_0B_0
			A_3B_1	A_2B_1	A_1B_1	A_0B_1	
		A_3B_2	A_2B_2	A_1B_2	A_0B_2		
	A_3B_3	A_2B_3	A_1B_3	A_0B_3			
X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0

Sumarea produselor A_iB_j se face ținând cont şi de transportul intermediar c_+.

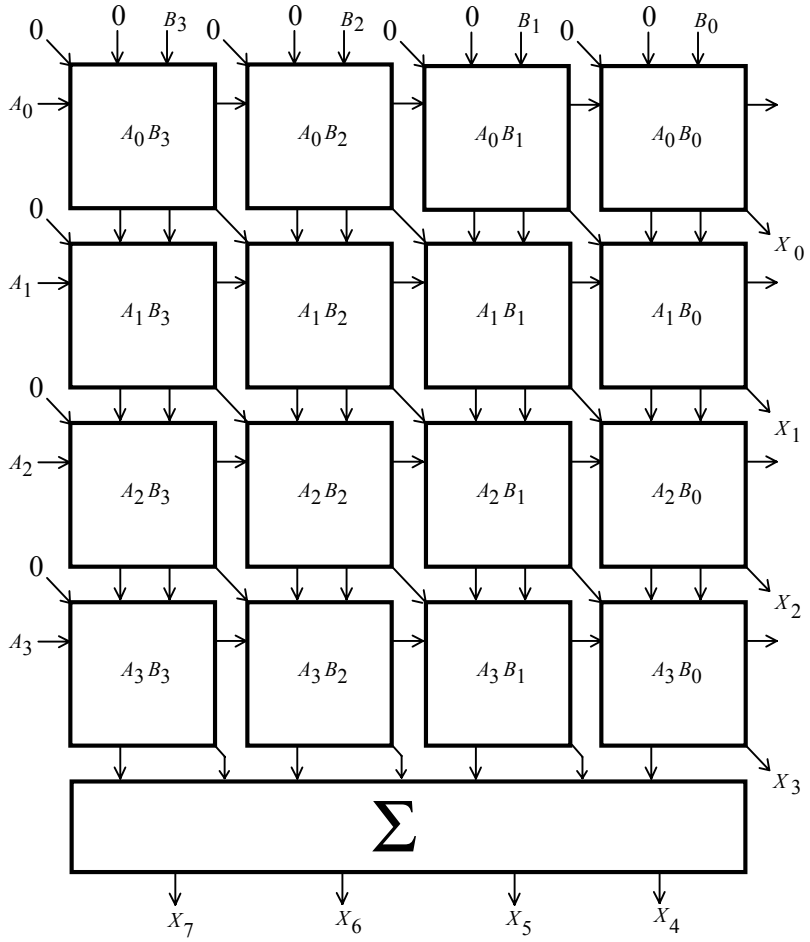


Fig. 5.22 *Înmulțitor combinațional de 4 biți*

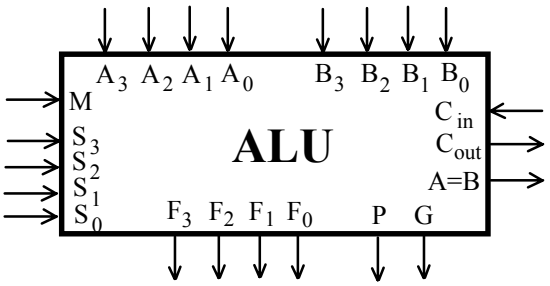


Fig. 5.23 *Unitate logico-aritmetică de 4 biți*

Unitatea logico-aritmetică (prescurtat ULA sau ALU – Arithmetic and Logic Unit) este un circuit care realizează prelucrări diverse asupra aceloraşi variabile aplicate intrărilor. Exemplul din figura 5.23 este un circuit ALU pentru operanzi de 4 biți. Operandul stâng are intrările $A_3A_2A_1A_0$, iar operandul drept are intrările $B_3B_2B_1B_0$. Intrarea M stabileşte tipul operației efectuate (aritmetică sau logică), iar cele 4 intrări de selecție aleg una din cele 16 operații posibile. Rezultatul operației apare la ieşirile $F_3F_2F_1F_0$.

Există însă şi o serie de semnale cu semnificație independentă, numite **indicatori**, sau **flag**-uri. Ele caracterizează relații între operanzi sau rezultatul operației efectuate. Printre operațiile aritmetice care pot fi efectuate de ALU este şi adunarea celor doi operanzi. Există deci o intrare de transport de la rangul inferior, C_{in} , dar şi o ieşire, C_{out}, care se activează atunci când rezultatul depăşeşte numărul de biți disponibili. O altă ieşire semnalizează identitatea celor doi operanzi, iar altele furnizează semnalele P şi G pentru transportul anticipat.

Fiind un circuit cu mai multe funcții, bănuim că şi **complexitatea** lui este mai mare. Ea este totuşi greu de evaluat, dacă privim circuitul numai din punct de vedere funcțional. În funcție de performanțele impuse, de restricțiile tehnologice sau experiența subiectivă a proiectantului se pot concepe diverse organizări distincte. Circuitul ALU de 4 biți de tipul SN74181, realizat în tehnologie TTL, de exemplu, conține 63 de porți logice şi 153 intrări.

5.4 Complexitatea structurilor combinaționale

Complexitatea unui circuit combinațional oferă o imagine asupra mărimii circuitului, asupra efortului structural depus pentru realizarea lui. Dacă notăm cu *n* numărul variabilelor de intrare, atunci vom nota cu $S(n)$ (de la *size*) complexitatea circuitului. Ea reprezintă, prin definiție, numărul total de intrări în porțile logice ale circuitului ([Ştefan, 1993]).

Adâncimea circuitului reprezintă numărul maxim de nivele logice (porți logice) parcurse de un semnal de la intrare până la ieşirea circuitului. Se notează de obicei cu $D(n)$, (de la *depth*). De obicei, un circuit realizat cu adâncime minimă tinde spre o complexitate maximă şi invers.

Pentru că discutăm despre circuite, trebuie să punem în evidență un caz particular de circuit, numit **formulă**. **Formula** se defineşte ca o rețea de porți logice, cu proprietatea că fiecare poartă are un fan-out egal cu 1. **Circuitul** este o rețea de porți logice care conține cel puțin o poartă cu fan-out mai mare sau egal cu 2. Termenii de formulă şi circuit indică faptul că o subexpresie este scrisă de câte ori apare în expresia unei funcții binare, în timp ce într-un circuit o poartă se poate implementa o singură dată şi conecta la intrarea mai multor porți.

Exemplul 5.3

Să exemplificăm definițiile anterioare folosind circuitele din figura 5.24. Circuitul din figura 5.24a este o formulă de complexitate $S(4) = 6$ şi adâncime $D(4) = 3$. Circuitul din figura 5.24f este tot o formulă de aceeaşi complexitate, dar de adâncime mai mică, $D(4) = 2$. În figura 5.24e avem un circuit de complexitate $S(4) = 12$ şi adâncime $D(4) = 4$.

Se constată uneori că anumite moduri de definire a funcțiilor binare, uneori cele recursive, impun structuri de complexitate mică, dar de adâncime foarte mare, lucru care afectează timpul de propagare prin circuit. Dacă folosim **O-notația** din teoria complexității temporale a algoritmilor, putem enunța **teorema lui Spira**, care permite reducerea adâncimii unei formule: Orice structură combinațională de tip formulă cu $D(n) = O(n)$ poate fi transformată în alta cu $D(n) = O(\log n)$.

Demonstrația ne oferă şi procedeul practic de realizare a transformării. Ea poate fi urmărită mai uşor pe exemplul din figura 5.24, unde $n = 4$. Dacă $n-1$ porți logice formează o formulă, atunci ieşirea primelor $n/2 - 1$ porți generează semnalul de selecție pentru un MUX cu două intrări; ultimele $n/2$ circuite sunt dublate, iar pe intrarea ce provenea de la primele $n/2 - 1$ porți se aplică 1 logic la grupul conectat la intrarea I_1 şi 0 logic la cel de la intrarea I_0. Formula a devenit mai complexă, dar adâncimea s-a redus de la $O(n)$ la $O(n/2)$. Regula se poate repeta pentru fiecare grup de porți logice, reducând de fiecare dată adâncimea până ce se ajunge la $D(n) = O(\log n)$.

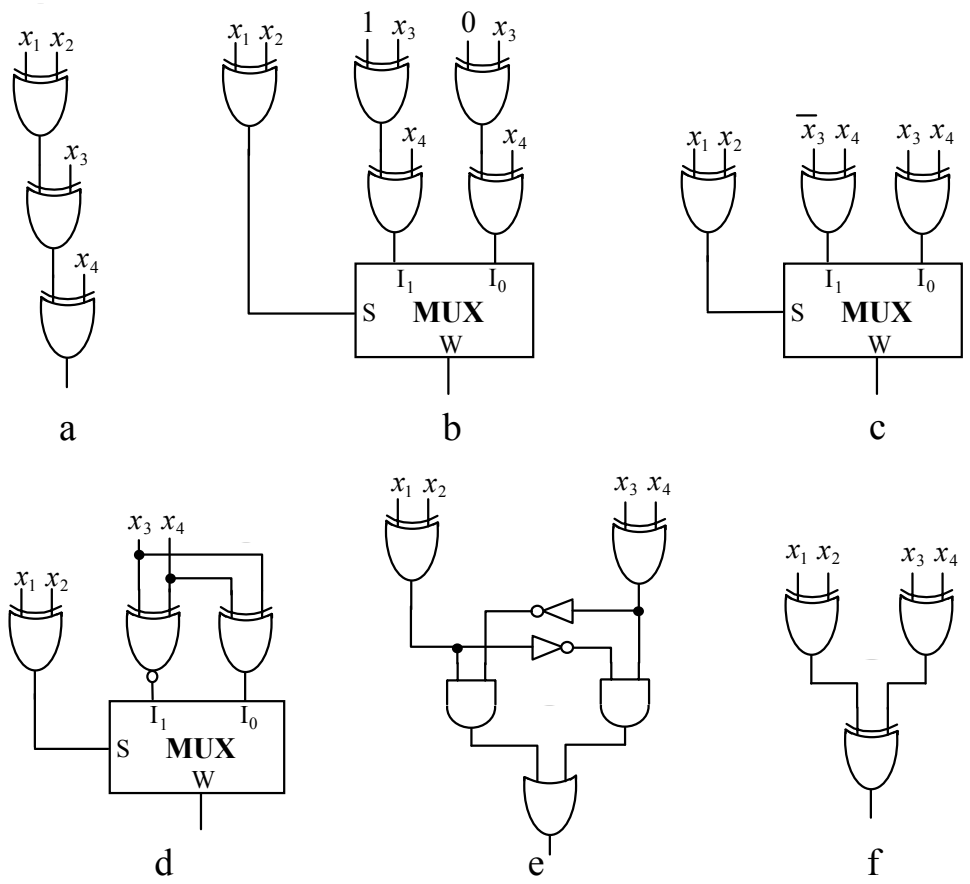


Fig. 5.24 *Exemplificarea teoremei lui Spira*

Probleme

5.1 Să se implementeze un sumator complet de 1 bit, folosind:

a) multiplexoare cu 4 căi de intrare;

b) decodificator cu 4 căi de ieşire şi porţi;

Comparaţi cele două soluţii din punct de vedere al numărului de circuite integrate şi al timpului de propagare.

5.2 Să se implementeze un convertor de cod, din cod binar în cod Gray, pentru numere reprezentate pe 3 biţi, folosind:

a) un decodificator şi porţi;

b) multiplexoare cu 4 căi de intrare;

c) număr minim de circuite;

Comparaţi cele 3 soluţii obţinute.

([Ştefan, 1992])

5.3 Să se implementeze cu un număr minim de porţi ŞI-NU circuitul logic cu ieşiri multiple descris de următoarele funcţii:

$$F = P_2 + P_3 + P_{10} + P_{13} + P_{15}$$
$$G = P_4 + P_5 + P_{10} + P_{13} + P_{15}$$
$$H = P_8 + P_9 + P_{10}$$

([Mureşan, 1996])

5.4 Să se extindă capacitatea de multiplexare/demultiplexare la 16 căi de intrare/ieşire, folosind multiplexoare/demultiplexoare cu 8 căi de intrare/ieşire. Analizaţi posibilităţile de implementare atât în tehnologie TTL, cât şi în tehnologie CMOS.

5.5 Circuitul logic combinaţional considerat posedă 4 intrări, 2 ieşiri şi funcţionează astfel încât:

- dacă $F = 00$, atunci $Q = I$;
- dacă $F = 01$, atunci $Q = I + 1 \pmod 4$;
- dacă $F = 10$, atunci $Q = I - 1 \pmod 4$;
- dacă $F = 11$, atunci $Q = \bar{I}$,

unde F, Q şi I sunt cuvinte de câte doi biţi.

a) Să se scrie funcţiile logice Q_1 şi Q_0 asociate ieşirilor circuitului.

b) Să se implementeze circuitul anterior descris.

([Ştefan, 1992])

5.6 La intrarea unui circuit se aplică doi operanzi de câte 1 bit, notaţi cu A şi B. Un cod de 2 biţi, notat cu F, stabileşte ieşirea Y_1 a circuitului, după cum urmează: dacă $F = 00$, atunci $Y_1 = \overline{A \cdot B}$; dacă $F = 01$, atunci $Y_1 = A + B$; dacă $F = 10$, atunci $Y_1 = \overline{B}$; iar dacă $F = 11$, atunci $Y_1 = A \oplus B$. Circuitul mai are o ieşire, notată cu Y_2, care se activează prin 1 logic dacă $A = B$. Să se implementeze circuitul.

5.7 Se consideră un circuit care generează la ieşire $M = \lceil \log_2 N \rceil + 1$, pentru N reprezentat binar aplicat la intrare, $N \in (0,255]$. Se cer:

a) funcția logică a circuitului;

b) implementarea circuitului.

([Ştefan, 1992])

5.8 Să se proiecteze un sumator-scăzător pentru cuvinte de patru biți reprezentând numere pozitive.

([Ştefan, 1992])

5.9 Să se proiecteze un sumator pentru două numere BCD de câte o cifră.

([Ştefan, 1992])

5.10 Aveți la dispoziție un decodificator BCD-7segmente care afişează cifrele 6 şi 9 aşa cum se vede în partea stângă a figurii de mai jos. Proiectați un circuit care să utilizeze decodificatorul dat, şi care să afişeze cifrele menționate ca în partea dreaptă a figurii de mai jos.

5.11 Proiectați un rotitor combinațional cu 16 intrări şi 16 ieşiri, care are 4 intrări de control. Cuvântul furnizat la ieşire este cel aplicat pe intrare, dar rotit cu un număr de poziții dat de cuvântul de control. De exemplu, pentru cuvântul de intrare ABCDEFGHIJKLMNOP (fiecare literă reprezintă un bit) şi cuvântul de selecție 0101 (5), atunci cuvântul la ieşire va fi FGHIJKLMNOPABCDE. Nu desenați schema completă a circuitului, numai schițați şi descrieți circuitul în termeni generali, arătând ce tipuri de circuite integrate folosiți şi care este numărul lor.

([Wakerly, 1990])

5.12 Proiectați un circuit combinațional care are ca intrări doi operanzi întregi fără semn pe câte 8 biți, X şi Y, şi un semnal de control MIN/MAX. Ieşirea circuitului este un întreg fără semn pe 8 biți, Z, astfel încât $Z = 0$ dacă $X = Y$; în caz contrar, $Z = \min(X,Y)$ dacă MIN/MAX = 1, şi $Z = \max(X,Y)$ dacă MIN/MAX = 0.

([Wakerly, 1990])