UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale 2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Proiectarea Logică

04. Clase de circuite combinaţionale

CLASE DE CIRCUITE COMBINATIONALE

Tehnologiile utilizate pentru producerea circuitelor integrate prezintă, adesea, particularităţi proprii tehnologiilor respective. Faptul că aceeaşi funcţie logică poate avea mai multe expresii algebrice are un impact deosebit de util atunci când particluarităţile unei tehnologii impun utilizarea unui anumit stil de proiectare.

Dar nu numai tehnologiile aduc în discuţie stilurile de proiectare.
Se întâmplă, adeseori, să să trebuiască să se utilizeze anumite circuite, pentru că sunt acestea sunt disponibile la furnizor sau sunt deja achiziţionate, ori pot apare constrângeri, din raţiuni economice, care impun utilizarea unei anumite familii tehnologice de circuite etc.

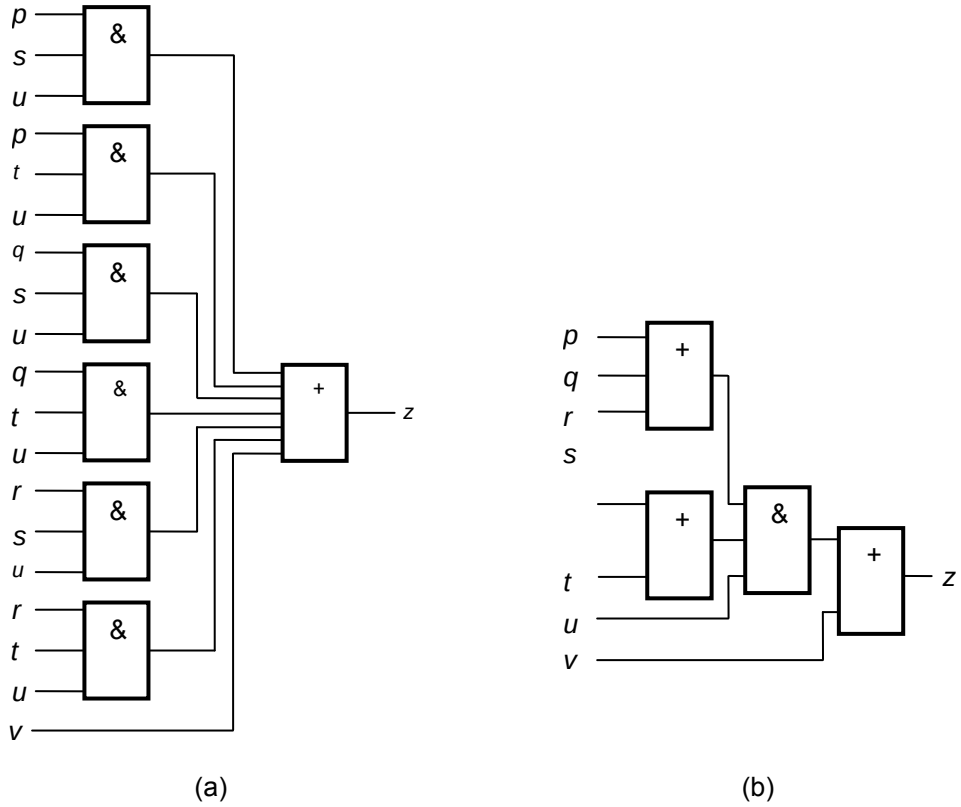

Figura 1. Circuitul exemplului 1.

Exemplul 1. Se consideră funcţia următoare:
$z(p, q, r,s,t, u, v) = psu + ptu + qsu + qtu + rsu + rtu + v$.
Această funcţie este exprimată, deja, în formă minimală printr-o sumă de produse. Implementarea acesteia, utilizând porţi ŞI, SAU, necesită şase porţi ŞI cu câte trei intrări şi o poartă SAU cu şapte intrări, aşa cum se poate urmări în figura 1(a). Costul total, în literali, al implementării acestei funcţii este 19.

Există posibilitatea ca prin factorizarea expresiei iniţiale, a funcţiei z, să se obţină o implementare cu un cost mai scăzut:

$$z = (ps + pt + qs + qt + rs + rt)u + v \quad (1)$$
$$z = ((p + q + r)s + (p + q + r)t)u + v \quad (2)$$
$$z = (p + q + r)(s + t)u + v \quad (3)$$

Notând expresiile obţinute după cum urmează:

$$w = p + q + r \quad (4)$$
$$x = s + t \quad (5)$$

Expresia (3) se poate rescrie astfel:

$$z = wxu + v \quad (6)$$

În această exprimare algebrică funcţia *z* necesită doar două porţi SAU cu câte două linii de intrare, o poartă SAU cu trei linii de intrare şi o poartă ŞI tot cu trei linii de intrare. Costul expresiei (3) este de doar nouă literali (pentru că *w* şi *x* sunt consideraţi literali independenţi, fiind funcţii intermediare). Implementarea în formă factorizată a funcţiei *z* este ilustrată în figura 1(b).

Această implementare se numeşte, generic, *circuit multi-nivel*.
Un astfel de circuit reduce mult numărul de porţi şi fire (conexiuni) dar, din cauza numărului crescut de nivele logice, timpul de întârziere al circuitului poate fi mai mare, depinzând mult de tehnologia de realizare, comparativ cu implementarea în două nivele ilustrată în figura 1(a).

1. Implementarea circuitelor combinaţionale utilizând doar porţi ŞI-NU ori doar porţi SAU-NU

Anumite tehnologii pot produce predilect fie porţi ŞI-NU, fie porţi SAU-NU. Din acest motiv implementarea unui circuit combinaţional utilizând fie doar porţi ŞI-NU, fie doar porţi SAU-NU constituie o aplicaţie practică. Este util, în abordarea care urmează, să fie reamintite teoremele DeMorgan:

$$(uv)' = u' + v', (u + v)' = u'v', \text{ şi}$$
$$u + v = (u'v')', uv = (u' + v')'$$

Utilizând aceste teoreme se poate afirma că o funcţie ŞI-NU poate fi implementată printr-o poartă SAU ale cărei intrări sunt inversate (negate). În mod similar o funcţie SAU-NU poate fi implementată printr-o poartă ŞI având intrările negate.

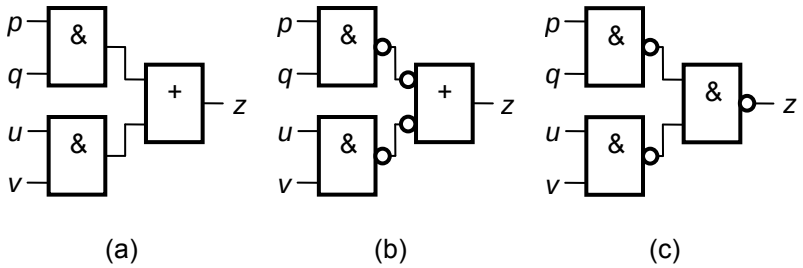

Figura 2. Conversia unui circuit ŞI - SAU într-un circuit ŞINU - ŞINU

Se consideră, pentru început, doar circuite exprimate în sume de produse numite, adeseori, circuite ŞI - SAU. Conversia unui astfel de circuit într-un circuit alcătuit exclusiv cu porţi ŞINU-ŞINU este prezentată în figura 2.

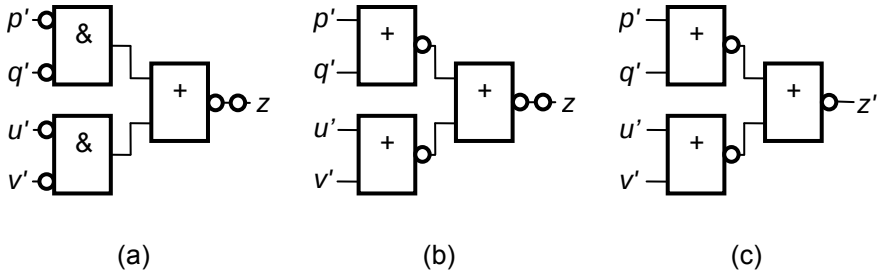

Figura 3. Conversia unui circuit ŞI-SAU într-un circuit SAU-NU – SAU-NU.

Circuitul iniţial, prezentat în figura 2(a) este transformat într-un circuit funcţional echivalent inversând atât ieşirile porţilor ŞI cât şi intrările porţii SAU, corespunzătoare (conform $(w')' = w$), aşa cum se poate vedea în figura 2(b). În continuare, se poate remarca lesne că o poartă SAU cu intrările negate este echivalentă funcţional cu o poartă ŞI–NU (figura 2(c)).

Conversia unui circuit ŞI – SAU într-un circuit SAU-NU – SAU-NU este prezentată în figura 3.

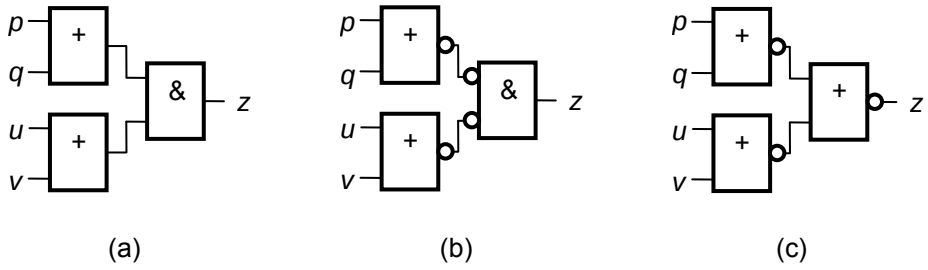

Figura 4. Conversia unui circuit SAU – ŞI într-un circuit SAUNU - SAUNU

Circuitele SAU – ŞI se convertesc utilizând un procedeu similar. Astfel, conversia unui circuit SAU – ŞI într-un circuit format doar cu porţi SAU-NU este prezentată în figura 4.
Această conversie transformă, deasemenea, circuitul iniţial (figura 4(a)) într-altul funcţional echivalent şi are ca procedeu inversarea ieşirilor porţilor SAU şi inversarea intrărilor porţii ŞI (figura 4(b)).
Având în vedere că o poartă ŞI cu intrările negate este echivalentă funcţional unei poarţi SAU-NU, se obţine circuitul echivalent constituit doar din porţi SAU-NU (figura 4(c)).

Conversia unui circuit SAU–ŞI, simplu, într-un circuit care utilizează exclusiv cu porţi ŞI-NU este prezentată în figura 5.

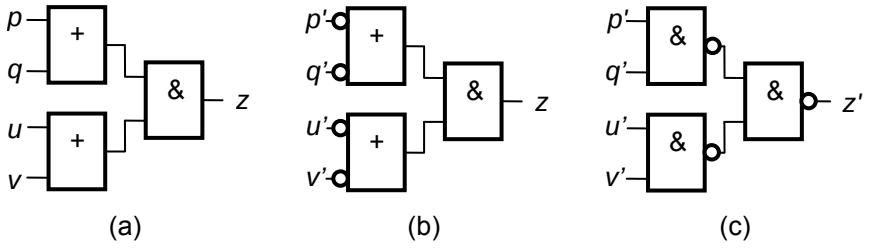

Figura 5. Conversia unui circuit SAU – ŞI într-un circuit ŞI-NU – ŞI-NU

Această conversie urmează un procedeu, similar unuia utilizat anterior, prin care în circuitul iniţial (prezentat în figura 5(a)) se inversează variabilele de intrare şi liniile de intrare în porţile SAU în (figura 5(b)). În continuare, având în vedere că o poartă SAU cu liniile de intrare negate este echivalentă funcţional unei porţi ŞI-NU şi prin negarea liniei de ieşire a porţii *z* (pentru omogenitatea transformării) se obţine conversia finală (figura 5(c)).

Procedeele utilizate pentru circuitele cu două nivele sunt aplicabile şi circuitelor multinivel atunci când se intenţionează constituirea unor circuite multinivel alcătuite doar din porţi având aceeaşi funcţionalitate (ŞI-NU ori SAU-NU).

2. Utilizarea decodoarelor şi multiplexoarelor

Decodoarele şi multiplexoarele sunt circuite combinaţionale complexe, realizate integrat, fiind utilizabile şi pentru implementarea unor circuite combinaţionale arbitrare, altele decât cele pentru care au fost proiectate. Aceste dispozitive sunt circuite integrate pe scara medie (MSI) şi pot oferi, pentru proiecte de complexitate mică şi medie, soluţii surprinzător de eficiente. Datorită complexităţii funcţionale a acestor circuite MSI costul proiectării logice se exprimă prin numărul de capsule utilizate.

Cu cât sunt utilizate mai puţine capsule în tehnologia MSI, cu atât este mai performantă proiectarea cu aceste circuite.

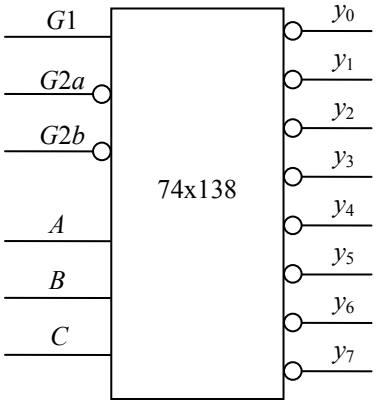

Figura 6. Diagrama generală a decodorului 74X138.

Decodoarele. Acestea mai sunt numite şi demultiplexoare, sunt circuite combinaţionale având, în principiu, n linii de intrare şi mai multe linii de ieşire. Fiecare linie de ieşire, a unui decodor, corespunde unui anumit minterm al spaţiului Boole-ean cu n variabile, $B^n = \{0, 1\}^n$.

În figura 6 este prezentat decodorul 74x138. Acesta are trei linii de intrare de validare ($G1$, $G2a$ şi $G2b$) şi trei lini de intrare de date (A, B şi C). Liniile de ieşire ale acestui decodor sunt notate y_0, y_1, … şi y_7. Acestea sunt active atunci când au valoarea 0. Liniile de validare $G1$, $G2a$ şi $G2b$ sunt, de asemenea, active atunci când au valoarea 0.
Funcţionarea liniilor de ieşire este descrisă de următoarele relaţii:

$$valid = G1 \cdot (G2a)' \cdot (G2b)', \tag{7}$$

$$\begin{aligned}
y_0' &= A' \cdot B' \cdot C' \cdot valid,\\
y_1' &= A \cdot B' \cdot C' \cdot valid,\\
y_2' &= A' \cdot B \cdot C' \cdot valid,\\
y_3' &= A \cdot B \cdot C' \cdot valid,\\
y_4' &= A' \cdot B' \cdot C \cdot valid,\\
y_5' &= A \cdot B' \cdot C \cdot valid,\\
y_6' &= A' \cdot B \cdot C \cdot valid,\\
y_7' &= A \cdot B \cdot C \cdot valid,
\end{aligned} \tag{8 – 15}$$

Atâta timp cât semnalul *valid* are valoarea 0 (ceea ce înseamnă că $G1 = 0$, $G2a = 1$ şi $G2b = 1$) liniile de ieşire, toate, vor fi inactive (au valoarea 1). Atunci când semnalul *valid* are valoarea 1, se activează o singură linie de ieşire corespunzător valorilor liniilor de intrare de date A, B şi C. Astfel, dacă $valid = 0$ şi $A = 1$, iar $B = 0$ şi $C = 0$, este activată linia de ieşire y_1 ($y_1 = 0$), iar toate celelalte linii de ieşire sunt inactive ($y_i = 1$, pentru $i = 0, 2, 3, 4, 5, 6, 7$).
Pentru astfel de decodor se utilizează notaţia, generică, decodor 3:8. Aceasta este o denumire mult răspândită, şi foarte expresivă. În această manieră se pot introduce, generic, decodoare $n : 2^n$, unde n este numărul de linii de date ale decodorului generic.
În afara acestui decodor există şi decodorul 74x139, care într-o capsulă cu 16 pini (aceeaşi capsulă este utilizată şi pentru 74x138) conţine două decodoare independente de tipul 2 : 4. Fiecare decodor 2 : 4 are o linie de intrare de validare unică, separată, activă tot prin valoarea 0.

Pare, într-un fel, neobişnuit faptul că liniile de ieşire sunt active prin valori 0. Atunci când au apărut aceste circuite (din familia MSI) industria circuitelor de comutaţie era dominată de tehnologia bipolară TTL. O caracteristică intrinsecă a acestei familii de circuite face ca viteza de comutaţie a circuitelor inversoare (ŞI-NU, spre exemplu) să fie mai mare decât a celor ne-inversoare (ŞI, spre exemplu). Cu

alte cuvinte, aceste circuite decodoare erau proiectate să se conecteze, preferenţial, la circuite inversoare (ŞI-NU, spre exemplu).

Deoarece orice funcţie booleană poate fi exprimată printr-o sumă de mintermi se poate lesne realiza că prin utilizarea unui decodor potrivit ales şi a unei porţi SAU (presupunând că liniile de ieşire ale decodorului sunt active prin valoarea 1) este fezabilă implementarea respectivei funcţii direct printr-o sumă de mintermi corespunzătoare, fără o prealabilă minimizare, în principiu. Deoarece liniile de ieşire sunt active prin valori 0, porţile SAU sunt înlocuite prin porţi SAU cu liniile intrare inversate, ceea ce este echivalentul porţilor ŞI-NU. În tehnologia TTL porţile ŞI-NU erau componenta de bază, având o viteză de comutaţie superioară.

Mai mult, orice circuit combinaţional cu *n* linii de intrare şi *m* linii de ieşire poate fi implementat direct, în principiu printr-un singur decodor, potrivit ales şi *m* porţi ŞI-NU corespunzătoare.

Procedeul implementării unei funcţii booleene printr-un decodor generic şi o poartă are drept date de intrare expresia în sumă de mintermi a respectivei funcţii. Decodificatorul este ales, ori proiectat, astfel încât să genereze toţi mintermii variabilelor de intrare.
Liniile de intrare ale porţii sunt conectate la mintermii corespunzători, după cum este definită funcţia.

Exemplul 2. Se consideră un circuit sumator pentru un bit având funcţionarea descrisă prin următorul tabel de adevăr:

Tabelul 1. Tabelul de adevăr al unui sumator binar cu un rang.

x	y	t_i	t_e	*suma*
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

Se remarcă faptul că sumatorul pentru un singur rang binar impelementează două funcţii distincte, *suma* şi transportul spre rangul superior t_e. Din tabelul 1 se deduc ecuaţiile liniilor de ieşire ale funcţiilor sumatorului:

$$suma(x, y, t_i) = \Sigma m(1,2,4,7),$$
$$t_e(x, y, t_i) = \Sigma m(3,5,6,7).$$

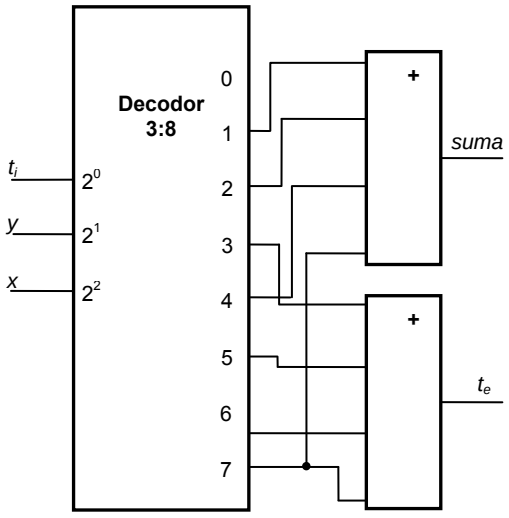

Figura 7. Implementarea unui sumator cu un rang utilizând un decodificator şi două porţi SAU.

În acest exemplu sunt trei variabile de intrare producând în total opt mintermi. În consecinţă, se alege un decodor generic cu trei linii de intrare de date şi opt linii de ieşire - similar celui prezentat anterior, 74x138. Pentru facilitarea înţelegerii exemplului nu vor fi considerate liniile de intrare de validare iar liniile de ieşire vor fi presupuse active prin valoarea 1. Schimbarea valorii active a liniilor de ieşire corespunde, în cazul practic, alegerii unor porţi SAU cu intrările negate, echivalente (în fapt) unor porţi ŞI-NU.
Implementarea generică este prezentată în figura 7. Decodorul generează toţi cei 8 mintermi pentru cele trei linii de intrare x, y şi t_i.
Poarta corespunzătoare liniei de ieşire *suma*, pentru sumatorul boolean cu un rang, calculează disjuncţia mintermilor 1, 2, 4 şi 7. Poarta corespunzătoare liniei de ieşire t_e, pentru transportul spre rangul următor, formează disjuncţia mintermilor 3, 5, 6 şi 7. Mintermul 0 este singurul minterm neutilizat în această aplicaţie.
În acest exemplu sunt trei variabile de intrare producând în total opt mintermi. În consecinţă, se alege un decodor generic cu trei linii de intrare de date şi opt linii de ieşire - similar celui prezentat anterior, 74x138. Pentru facilitarea înţelegerii exemplului nu vor fi considerate liniile de intrare de validare iar liniile de ieşire vor fi presupuse active prin valoarea 1. Schimbarea valorii active a liniilor de ieşire corespunde, în cazul practic, alegerii unor porţi SAU cu intrările negate, echivalente (în fapt) unor porţi ŞI-NU.
Implementarea generică este prezentată în figura 7. Decodorul generează toţi cei 8 mintermi pentru cele trei linii de intrare x, y şi t_i.
Poarta corespunzătoare liniei de ieşire *suma*, pentru sumatorul boolean cu un rang, calculează disjuncţia mintermilor 1, 2, 4 şi 7. Poarta corespunzătoare liniei de ieşire t_e, pentru transportul spre rangul următor, formează disjuncţia mintermilor 3, 5, 6 şi 7. Mintermul 0 este singurul minterm neutilizat în această aplicaţie.
Capacitatea de decodare depinde de numărul liniilor de intrare. Numărul liniilor de intrare poate

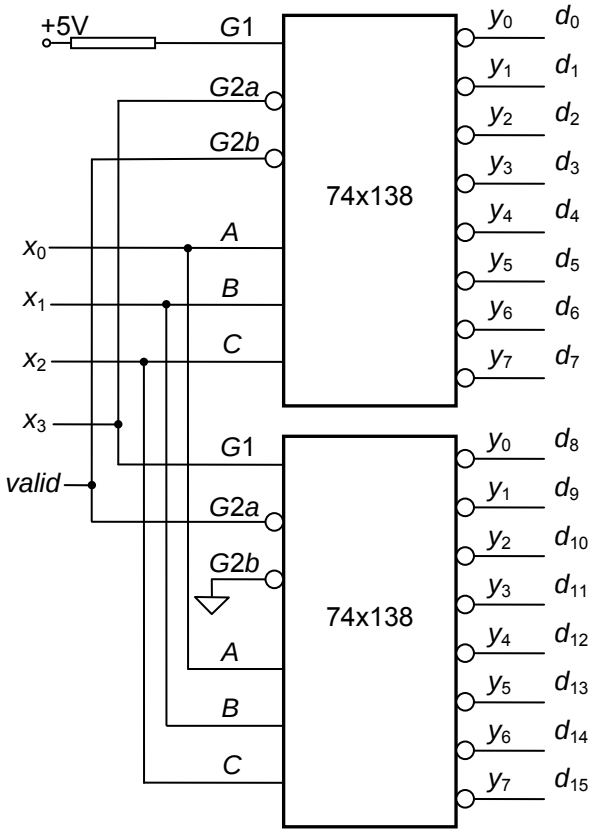

Figura 8. Implementarea unui decodor binar
cu 4 linii de date, utilizând decodoare 74x138.

introduce o limitare a utilizării decodoarelor. Atunci când este necesară o extindere a capacităţii de decodare se pot aplica procedee de conectare, în cascadă, a mai multor decodoare disponibile astfel încât să se obţină un decodor echivalent extins.

Exemplul 3. Se consideră proiectarea unui decodor având patru linii de date şi 16 linii de ieşire atunci când sunt disponibile doar decodoare 3 : 8.

Multiplexoarele. Aceste circuite mai sunt numite, adeseori, selectoare. Aceste dispozitive au două grupuri de linii de intrare şi o singură linie de ieşire furnizând valoarea asertată a ieşirii iar, la anumite modele, este furnizată şi valoarea complementară a acesteia. Primul grup de linii de intrare sunt numite *linii de date* spre deosebire de cel de-al doilea grup de linii de intrare care sunt numite *linii de selecţie*. Valorile binare plasate pe liniile de selecţie sunt interpretate drept adresa binară a uneia dintre liniile de date. Astfel, dacă un multiplexor are n linii de selecţie atunci, corespunzător, vor fi 2^n linii de date ale multiplexorului respectiv. Există, deasemenea, o linie de activare sau de validare a liniei de ieşire. Această linie de validare este activă, de regulă, prin valori 0. Atunci când linia de validare are valoarea 1 linia de ieşire are valoarea constantă indiferent de valorile liniilor de date şi/sau selecţie.

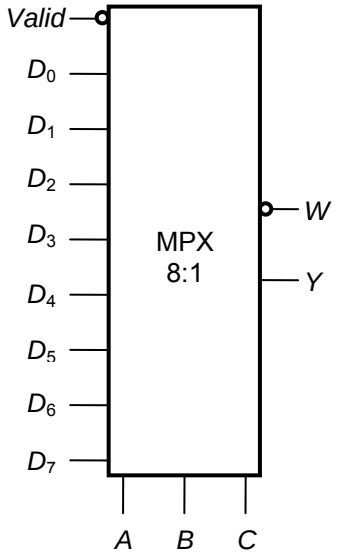

Figura 7. Multiplexor cu 8 linii de date.

Linia de ieşire, Y, satisface relaţia:

$$Y = (D_0A'B'C' + D_1A'B'C + D_2A'BC' + D_3A'BC + D_4AB'C' + D_5AB'C + D_6ABC' + D_7ABC)Valid \quad (7)$$

Utilizarea circuitului multiplexor pentru implementarea funcţiilor booleene este sugerată de ecuaţia (7). Astfel, cu un multiplexor 8:1 se poate implementa orice funcţie booleeană $f(u, v, w)$cu trei variabile. Pentru aceasta este suficient să se atribuie liniilor de date valorile constante 0 sau 1, corespunzătoare funcţiei respective:

$D_0 = f(0,0,0)$, $D_1 = f(0,0,1)$, ... şi $D_7 = f(1,1,1)$.

Se poate remarca că utilizând acelaşi multiplexor 8:1, se poate implementa orice funcţie de patru variabile deoarece o funcţie de patru variabile $f(t, u, v, w)$ poate fi scrisă astfel:

$$f(t, u, v, w) = t\, f(1, u, v, w) + t'\, f(0, u, v, w) \quad (8)$$

Deoarece $f(1, u, v, w)$ şi $f(0, u, v, w)$ sunt funcţii de trei variabile rezultă că liniilor D_i, $7 \le i \le 0$, li se vor atribui expresii constante ori care depind doar de variabila t. Deoarece expresiile în variabila t sunt extrem de simple (sunt doar două expresii netriviale: t şi t') rezultă că efortul de implementare al unei funcţii booleene cu patru variabile printr-un multiplexor 8: 1, este minim.

Se poate remarca, în acest sens, că:

- oricare patru funcţii de două variabile se pot implementa printr-o singură capsulă conţinând patru multiplexoare 2:1,
- oricare două funcţii de trei variabile se pot implementa printr-o singură capsulă conţinând două multiplexoare 4:1,
- oricare funcţie de patru variabile se poate implementa printr-o singură capsulă conţinând un multiplexor 8:1 şi
- oricare funcţie de cinci variabile se poat implementa printr-o singură capsulă conţinând un multiplexor 16:1.

Implementarea funcţiilor booleene având mai multe variabile, decât au fost amintite anterior, poate fi făcută, în anumite limite, cu un minim de efort. Abordarea teoretică face uz de o descompunere similară celei din relaţia (8) dar se factorizează două sau mai multe variabile. Expresiile rezultate în urma factorizării variabilelor se numesc *expresii reziduale* (care urmează să fie implementate la pinii liniilor de date ale multiplexorului utilizat).
Aceste expresii reziduale pot fi simplificate utilizând tehnicile cunoscute şi apoi utilizând, eventual, un număr mic de porţi suplimentare pentru implementarea expresiilor reziduale se obţine implementarea dorită.

Exemplul 3. Se consideră implementarea funcţiei majoritate utilizând un multiplexor 4:1. Tabelul de adevăr al funcţiei majoritate arată astfel:

Tabelul 2. Funcţia majoritate.

a	*b*	*c*	*f*	Remarci
0	0	0	0	$ab = 00 \rightarrow f = 0$
0	0	1	0	$ab = 00 \rightarrow f = 0$
0	1	0	0	$ab = 01 \rightarrow f = c$
0	1	1	1	$ab = 01 \rightarrow f = c$
1	0	0	0	$ab = 10 \rightarrow f = c$
1	0	1	1	$ab = 10 \rightarrow f = c$
1	1	0	1	$ab = 11 \rightarrow f = 1$
1	1	1	1	$ab = 11 \rightarrow f = 1$

Din tabelul 2, care descrie funcţiea majoritate, se poate alcătui un tabel mai aproape de scopul implementării funcţiei printr-un multiplexor 4:1. Acest tabel arată astfel:

Tabelul 3. Funcţia majoritate exprimată convenabil în raport cu multiplexorul 4:1.

a	*b*	*f*
0	0	0
0	1	*c*
1	0	*c*
1	1	1

Corespunzător tabelului 3 se poate găsi o implementare a funcţiei majoritate *f* printr-un multiplexor 4:1, aşa cum se poate vedea din figura 8.

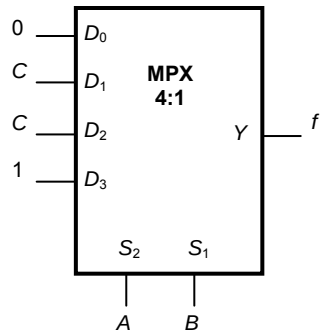

Figura 8. Implementarea funcţiei majoritate printr-un multiplexor 4 :1.

O altă cale de soluţionare a implementării funcţiilor booleene cu un număr mare de variabile este extinderea capacităţii multiplexoarelor existente, disponibile, prin interconectarea structurată a acestora, obţinând astfel multiplexoare cu un număr mare de linii de selecţie.

Exemplul 4. Un multiplexor 16:1 poate fi realizat, spre exemplu, din cinci multiplexoare 4: 1 aşa cum se arată în figura 9.

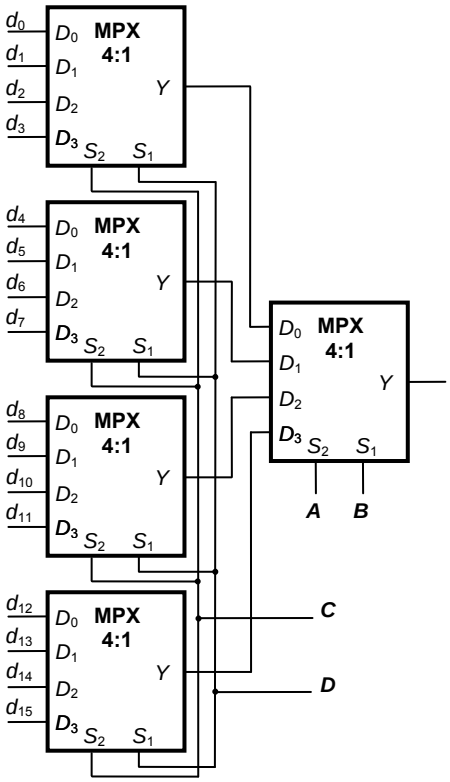

Figura 9. Multiplexor 16:1
realizat cu cinci multiplexoare 4:1.

Sunt disponibile, curent, multiplexoare cu 2, 4, 8 şi 16 linii de date. Astfel:

- 74*x*150 este un multiplexor cu 16 linii de date, patru linii de selecţii dar având disponibilă linia de ieşire doar complementată. Capsula acestui dispozitiv are 24 de pini. Multiplexoarele următoare au capsulele cu 16 pini.
- 74*x*151 este un multiplexor cu opt linii de intrare având disponibile ambele faze ale liniei de ieşire.
- 74*x*153 este un multiplexor cu patru linii de intrare, două asemenea multiplexoare în aceeaşi capsulă, având liniile de selecţie comune dar liniile de validare sunt separate, câte una pentru fiecare multiplexor.
- 74*x*157 este un multiplexor cu două linii de date şi o linie de selecţie, sunt patru asemenea multiplexoare într-o capsulă având comune linia de selecţie şi linia de validare, liniile de ieşire sunt disponibile doar asertate.
- 74*x*158 este un multiplexor similar cu 74*x*157 exceptând faptul că liniile de ieşire sunt disponibile doar complementate.

Există multiplexoare având linii de ieşire cu trei stări. La aceste multiplexoare linia de validare forţează linia de ieşire în starea de impedanţă ridicată (circuitul apare ca având, practic, linia de ieşire neconectată). Multiplexorul 74*x*251 este similar cu multiplexorul 74*x*151 ca mod de conectare la pinii capsulei şi ca funcţionalitate dar liniile de ieşire sunt cu trei stări (atât linia asertată cât şi cea complementată). Într-o situaţie similară sunt multiplexoarele 74*x*253 şi 74*x*257 versiuni, ale dispozitivelor 74*x*153 şi 74*x*157, liniile de ieşire având trei stări.

Exemplul 5. Se consideră, încă o dată, sumatorul binar cu un singur rang din exemplul 2.
Este o funcţie vectorială deoarece implementează atât suma rangului respectiv cât şi transportul spre rangul imediat următor. Procedeul de proiectare poate fi ilustrat considerând funcţia *suma*. Cele două variabile *x* şi *y* sunt conectate, în această ordine, la pinii de selecţie ai multiplexoarelor 4:1. Linia *x* este conectată la pinul S_2 iar linia *y* este conectată la pinul S_1. Valorile liniilor de date sunt determinate din

tabelul de adevăr al sumatorului. Atunci când $(x,y) = (0,0)$, ieşirea *suma* este egală cu t_i deoarece *suma* = 0 când $t_i = 0$ şi suma = 1 când $t_i = 1$. Din acest motiv trebuie ca variabila t_i să fie aplicată la pinii D_0 ai ambelor multiplexoare. Într-o manieră similară se determină că pinii D_1, D_2 şi D_3 aparţinând multiplexorului dedicat funcţiei *suma* au valoarea t_i. Similar se determină valorile celorlalţi pini aparţinând multiplexorului care calculează transportul spre rangul următor, t_e.

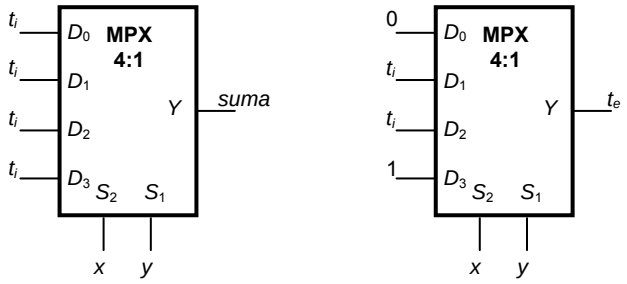

Figura 10. Implementarea sumatorului cu un rang, utilizând multiplexoare 4 :1.

Circuitele SAU-EX (suma-modulo-2) sunt adesea implementate prin multiplexoare. În figura 11 este prezentată o astfel de implementare, pentru poarta SAU-EX cu două variabile, *x* şi *y*.

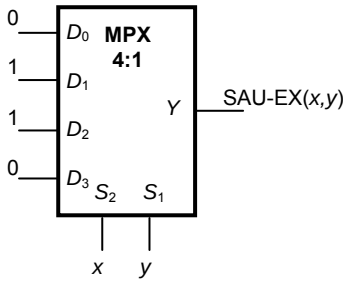

Figura 11. Implementarea cicuitului SAU-EX, utilizând un multiplexor 4 :1.

Memoriile cu conţinut fix

Memoriile cu conţinut fix, denumirea anglo-americană fiind *Read Only Memory* (abrevierea fiind ROM), sunt dispozitive matriciale conţinând valori binare adresabile printr-un vector de valori binare care conţine adresa unei locaţii. Unei locaţii i se asociază, de regulă un vector binar. Iniţial memoriile cu conţinut fix erau programate în momentul în care erau produse, în fabrică.

Mai nou aceste memorii pot fi programate, eventual re-programate, de utilizator prin mijloace proprii, specifice. Organizarea internă, generică, a unei memorii cu conţinu fix este descrisă în figura 11.

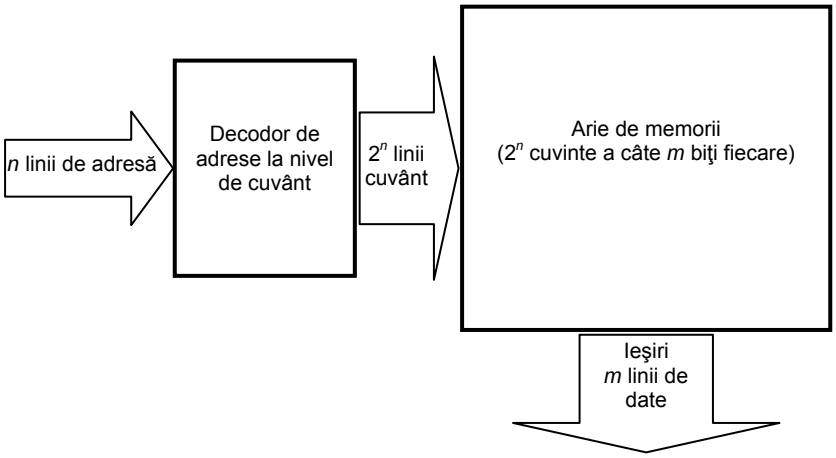

Figura 12. Organizarea internă a unei memorii cu conţinut fix.

Se consideră o memorie cu conţinut fix având 16 384 de biţi (16kb) formată dintr-o matrice de 128 x 128 de puncte care sunt sau nu sunt *conductoare*.
Circuitul, aşa cum se poate urmări în figura 12, are opt linii de ieşire şi 11 linii de adresă în total, fiind organizat în 2 048 de cuvinte de opt biţi fiecare (2 kocteţi). Pentru fiecare combinaţie de intrare este generată o combinaţie de ieşire, în conformitate cu specificaţiile utilizatorului.