Dan NICULA

ELECTRONICĂ DIGITALĂ
Carte de învăţătură 2.0

Editura Universităţii *TRANSILVANIA* din Braşov
ISBN 978-606-19-0563-8

2015

Lecţia 12

Aplicaţii cu circuite logice combinaţionale

12.1 Noţiuni teoretice

Proiectarea circuitelor logice combinaţionale complexe, cu multe intrări, nu urmează un anumit algoritm. Teoretic, se poate imagina că orice funcţie logică poate fi reprezentată într-o formă standard. Însă, pentru un număr mare de intrări, toate formele de reprezentare (expresie analitică, diagrama V-K, tabel de adevăr, forme canonice) sunt imposibil de gestionat mental.

Rămâne varianta unei proiectări pe baza comportamentului cunoscut al unor circuite combinaţionale elementare: codificator/decodificator, comparator, multiplexor/demultiplexor, sumator/scăzător, porţi logice.

12.2 Pentru cei ce vor doar să promoveze examenul

Nu este necesară parcurgerea acestui capitol.

12.3 Pentru cei ce vor să înveţe

1. Să se implementeze un circuit logic combinaţional la intrarea căruia se aplică un cuvânt M de 8 biţi şi un cuvânt N de 3 biţi. Ieşirea F a circuitului va fi activă când M este multiplu de 2^N.

   Soluţie

   Numărul binar M este multiplu al numărului 2^N dacă este îndeplinită relaţia: $M = K \times 2^N$. Observând că expresia $K \times 2^N$ este echivalentă cu *"numărul K exprimat în binar deplasat la stânga cu N poziţii"*, rezultă că relaţia este îndeplinită când ultimii N biţi ai cuvântului M sunt egali cu 0. Soluţia problemei se reduce la a determina care este poziţia primului 1 (din partea mai puţin semnificativă) a numărului M aplicat la intrare şi a o compara cu numărul N. Indexul celui mai puţin semnificativ 1 se poate determina cu un circuit codificator cu prioritate. Compararea indexului cu numărul N se poate face cu un circuit de comparare pe 3 biţi. Figura 12.1 prezintă circuitul proiectat.

2. Să se implementeze un circuit logic combinaţional care calculează relaţia: $F = [log_2 M] + 1$, pentru M număr pozitiv reprezentat pe 8 biţi. S-a notat cu [ ] partea întregă a numărului.

   Soluţie

   $F = [log_2 M] + 1 = [log_2(2 \cdot M)]$, ceea ce este echivalent cu indexul celui mai semnificativ bit al numărului M deplasat cu o poziţie spre stânga. Circuitul se poate realiza cu un codificator.

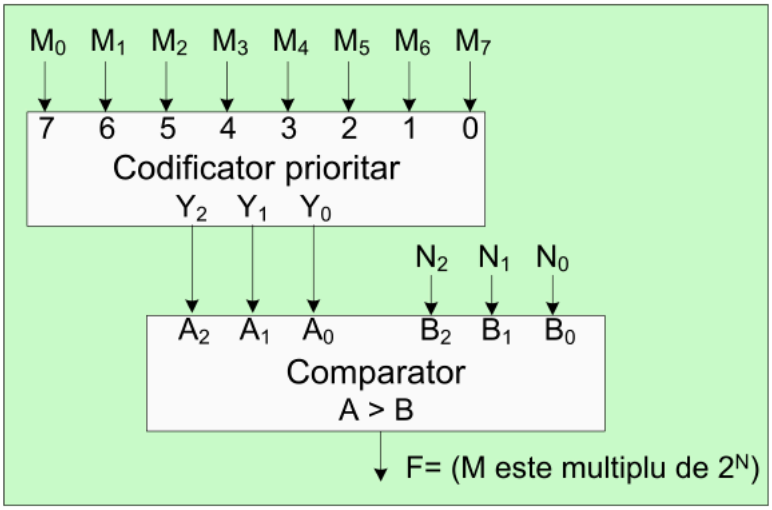


Figura 12.1 Circuit pentru implementarea relaţiei $M = K \times 2^N$, problema 1.

3. Să se proiecteze un circuit logic combinaţional cu 8 intrări I_i şi 8 ieşiri O_i, $i \in 7...0$. Circuitul activează în 1 numai ieşirea i de pe poziţia celui mai semnificativ bit 1 din cuvântul de intrare.

   Soluţie

   Codarea poziţiei celui mai semnificativ bit pe intrare se poate realiza cu un codificator prioritar iar decodarea pentru ieşire se poate realiza cu un circuit decodificator.

4. Să se implementeze un circuit logic combinaţional care activează ieşirea "1" dacă există un singur bit egal cu "1" într-un cuvânt de 8 biţi aplicat la intrare.

   Soluţie

   Rezolvarea se bazează pe observaţia că dacă la intrarea unui codificator se inversează ordinea biţilor, cel mai semnificativ devine cel mai puţin semnificativ şi invers, atunci se poate considera că la ieşirea codificatorului prioritar apare indexul celui mai puţin semnificativ bit, în cod complementat.
   Ca exemplu, se consideră numărul de intrare 0000_0100. Dacă acest cuvânt se aplică, în această ordine, la intrarea unui codificator prioritar, la ieşire se va prezenta indexul celui mai semnificativ bit, 2, combinaţia binară 010. Dacă numărul se aplică la intrarea unui codificator în mod inversat, adică 0010_0000, la ieşirea codificatorului prioritar, se va prezenta indexul bitului 5, adică 101. Prin complementarea bit cu bit, combinaţia 101 devine 010 adică indexul 2.

   În cazul unui număr de intrare având mai mult de un bit egal cu 1, la conectare directă se va codifica bitul cel mai semnificativ, iar la conectare inversată se va codifica bitul cel mai puţin semnificativ. De exemplu, dacă numărul de intrare este 0100_0100, codificatoarele vor genera 110 la conectare directă şi 101 la conectare inversată

   Se observă că dacă se însumează cele două coduri, dacă există un singur 1, atunci suma va fi întotdeauna $7_{10} = 111_2$, iar dacă există mai mult de un 1 la intrare suma celor două coduri va fi mai mare decât 7 şi va produce transport la ieşire (în cazul însumării cu sumator de 3 biţi).

   În cazul în care la intrare toţi biţii sunt 0, atunci ambele codificatoare vor prezenta la ieşire 000 iar cele două coduri însumate vor fi $0_{10} = 000_2$.

   Concluzia este că funcţia cerută a fi implementată se obţine la bitul de transport al sumatorului. Figura 12.2 prezintă circuitul proiectat.

5. Să se realizeze, folosind circuite comparator de 4 biţi o structură pentru compararea a două cuvinte de 8 biţi, respectiv 16 biţi.

   Soluţie

   Cuvintele se compară bit cu bit, începând de la bitul mai semnificativ spre cel mai puţin semnificativ. Dacă cei 2 biţi comparaţi sunt egali, se trece la compararea bitului de ordin inferior. Dacă biţii sunt diferiţi, atunci rezultatul operaţiei de comparare este determinat şi biţii mai puţini semnificativi sunt ignoraţi. Deci, există un transport de informaţie de la biţii mai semnificativi spre cei mai puţini semnificativi. În cazul utilizării circuitelor de comparare de 4 biţi, cel mai semnificativ circuit trebuie să informeze circuitul mai puţin semnificativ dacă

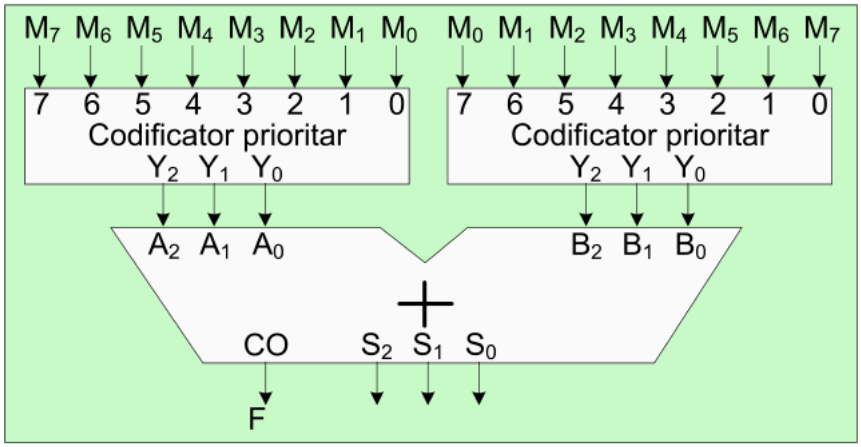


Figura 12.2 Circuit pentru detectarea prezenţei unui singur bit egal cu 1 în datele de intrare, problema 4.

rezultatul comparării a fost găsit sau partea mai semnificativă a cuvintelor de intrare este identică. În acest caz, circuitul care compară partea mai puţin semnificativă a datelor trebuie să fie activat. Circuitul este prezentat în figura 12.3. Comparatorul de 4 biţi are intrări specifice prin care transmite rezultatul comparării biţilor mai semnificativi. Comparatorul biţilor mai semnificativi are aceste intrări conectate ca şi cum biţii superiori au fost egali: $L_i = 0$, $E_i = 1$, $G_i = 0$. Ieşirile comparatorului mai semnificativ devin intrări pentru comparatorul mai puţin semnificativ.

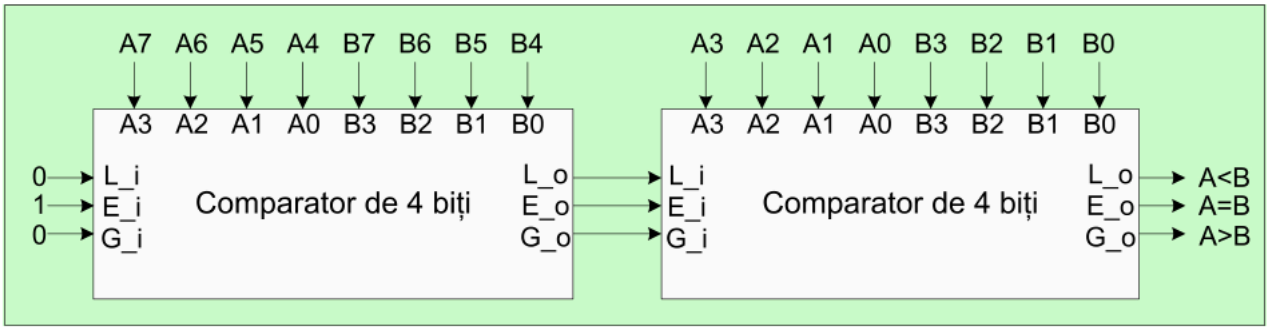


Figura 12.3 Circuit de comparare a două cuvinte de 8 biţi realizat cu două circuite comparator de 4 biţi, problema 5.

6. Să se realizeze un circuit logic combinaţional care semnalează pe ieşire identitatea celor două cuvinte de 3 biţi de la intrare. Să se implementeze circuitul cu:

   - porţi XOR sau
   - circuite multiplexor/demultiplexor.

   Soluţie

   Funcţia circuitului este egală cu 1 când cele două cuvinte de intrare sunt egale bit cu bit. Poarta logică XOR prezintă la ieşire 1 dacă biţii de intrare sunt diferiţi. În logică negată, poarta XNOR prezintă la ieşire 1 dacă biţii de intrare sunt egali. Această observaţie sugerează că circuitul de determinare a identităţii a două cuvinte se poate realiza dacă se compară cele două cuvinte bit cu bit, folosind porţi XNOR iar rezultatele acestora se unesc cu o poartă AND.

   $F(A,B) = \overline{(A_2 \oplus B_2)} \cdot \overline{(A_1 \oplus B_1)} \cdot \overline{(A_0 \oplus B_0)}$

   Utilizând teorema lui DeMorgan se poate scrie expresia funcţiei ca:

   $F(A,B) = \overline{(A_2 \oplus B_2) + (A_1 \oplus B_1) + (A_0 \oplus B_0)}$

   Circuitul este prezentat în figura 12.4-a,b.

   Funcţia de determinare a identităţii se poate realiza cu două circuite DMUX 1:8 şi MUX 8:1 dacă cele două cuvinte se conectează pe intrările de selecţie iar ieşirile de date ale demultiplexorului se conectează unu la unu

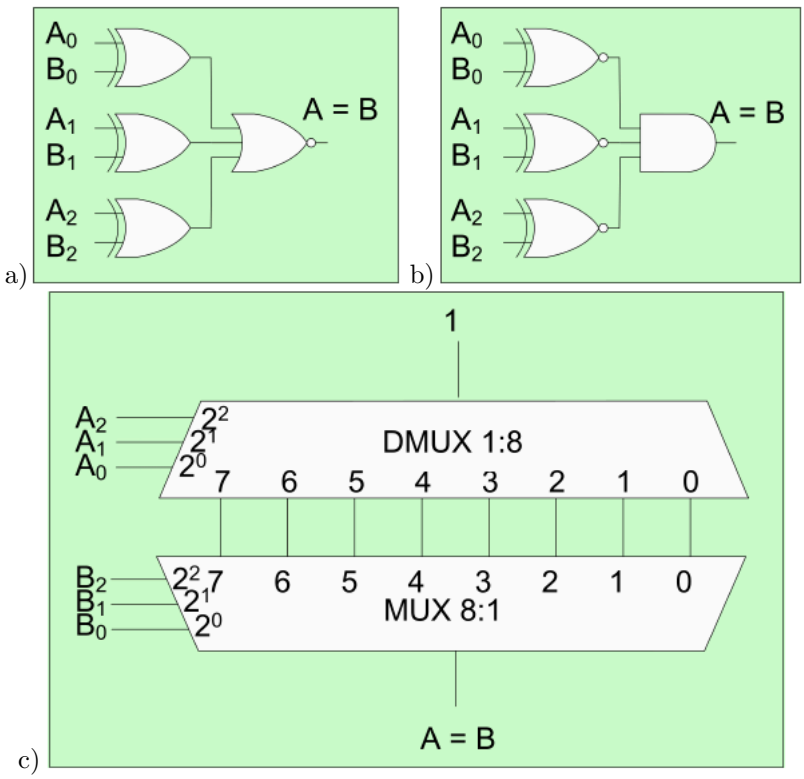


Figura 12.4 Circuit de verificare a identităţii a două cuvinte de 3 biţi, problema 6.

la intrările de date ale multiplexorului. În cazul identităţii celor două cuvinte, valoarea de 1 de la intrarea demultiplexorului se propagă la ieşirea cu indexul corespunzător primului operand şi apoi este preluată de pe intrarea cu indice corespunzător a multiplexorului. Circuitul este prezentat în figura 12.4-c.

7. Folosind circuite DMUX 1:8 şi MUX 8:1 să se realizeze un circuit logic combinaţional care semnalează pe ieşire existenţa următoarelor corespondenţe ale intrărilor (date de intrare reprezentate pe 3 biţi): $0-6$, $1-4$, $2-3$, $3-1$, $4-7$, $5-5$, $6-2$, $7-0$.

Soluţie

Soluţia se bazează pe funcţia circuitului demultiplexor de a activa ieşirea a cărei index este prezent pe intrările de selecţie şi funcţia circuitului multiplexor de a transmite la ieşire intrarea selectată. În cazul unui circuit similar cu cel prezentat în figura 12.4-c, dacă firele se conectează în ordine, se obţine un circuit care detectează identitatea intrărilor de selecţie, adică: $0-0$, $1-1$, $2-2$, $3-3$, $4-4$, $5-5$, $6-6$, $7-7$. Prin modificarea firelor se pot detecta alte corespondenţe. Corespondenţa specificată este prezentată în figura 12.5.

8. Proiectaţi un circuit care depistează o eroare într-un număr în format BCD. Circuitul semnalizează cu 0 cazurile când cei 4 biţi de la intrare nu corespund unui număr zecimal (între 0 şi 9). Adăugaţi proiectului 4 porţi cu două intrări pentru ca în cazul în care la intrare apare o combinaţie invalidă la ieşire să se transmită numărul zecimal 0.

Soluţie

Tabelul de adevăr al circuitului de semnalare a erorii este:

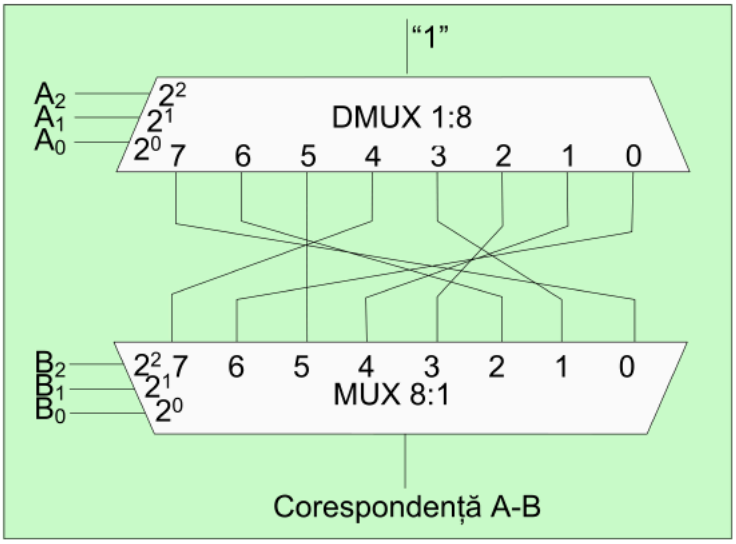


Figura 12.5 Circuit de verificare a corespondenţei unor date reprezentate pe 3 biţi, problema 7.

Nr.	B_3	B_2	B_1	B_0	$\overline{ERR}$
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

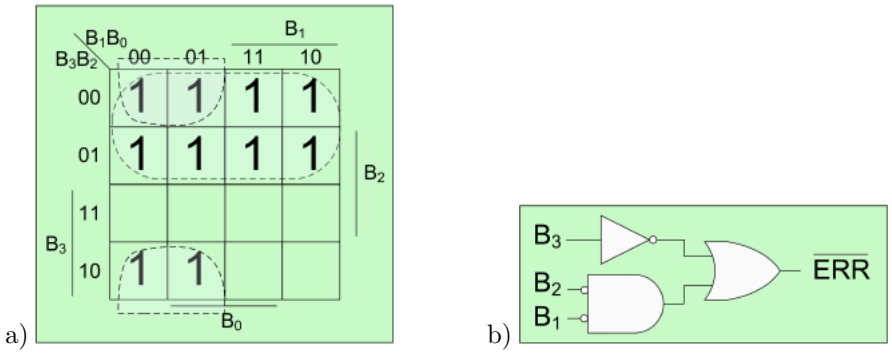


Figura 12.6 Funcţia de depistare a erorii de cod BCD: **a)** diagrama V-K, **b)** structura de porţi logice.

Diagrama V-K asociată este prezentată în figura 12.6-a. Din diagrama V-K rezultă ecuaţia:
$\overline{ERR} = \overline{B_3} + \overline{B_2} \cdot \overline{B_1}$
Circuitul care implementează funcţia $\overline{ERR}$ este prezentat în figura 12.6-b.

Pentru a evita transmiterea codurilor binare mai mari decât 9 şi înlocuirea acestora cu 0, trebuie adăugate pe cele 4 ieşiri 4 porţi AND cu două intrări. Pentru fiecare din cele 4 porţi o intrare este bitul $\overline{ERR}$. În acest fel,

dacă $\overline{ERR} = 0$ la ieşirea porţilor AND se va genera 0, indiferent de datele prezentate la intrare. Dacă $\overline{ERR} = 1$, toate porţile AND sunt deschise şi transferă la ieşire valorile de la intrarea circuitului. Circuitul astfel realizat este prezentat în figura 12.7.

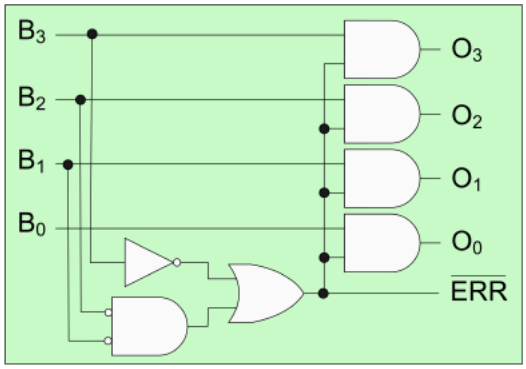


Figura 12.7 Circuit de mascare a erorilor de cod BCD.

9. Implementaţi cu porţi logice pe două nivele (fără a folosi celule semisumator sau sumator complet de 1 bit) un circuit de însumare a două numere reprezentate pe câte 2 biţi. Circuitul acceptă transport de intrare şi generează transport de ieşire.

   Soluţie

   Schema bloc a circuitului este prezentată în figura 12.8. S-au notat A_1A_0 şi B_1B_0 operatorii, S_1S_0 rezultatul, C_{in} transportul de intrare şi C_{out} transportul de ieşire. Sumatorul este un circuit combinaţional cu 5 intrări şi

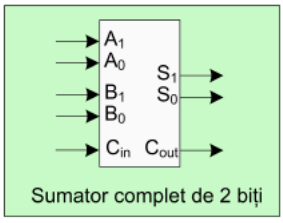


Figura 12.8 Schema bloc a sumatorului de 2 biţi referit la problema 9.

   3 ieşiri şi poate fi implementat pe două nivele de porţi logice. Ecuaţiile logice ale ieşirilor sunt:

   $S_0 = A_0 \oplus B_0 \oplus C_{in}$
   $C_0 = A_0 \cdot B_0 + A_0 \cdot C_{in} + B_0 \cdot C_{in}$
   $S_1 = A_1 \oplus B_1 \oplus C_0$
   $C_{out} = A_1 \cdot B_1 + A_1 \cdot C_0 + B_1 \cdot C_0$

   S-a notat C_0 transportul intermediar de la bitul 0 (cel mai puţin semnificativ).

10. Se doreşte transmiterea pe o distanţă mare a unui bus de 1024 biţi de date. Soluţia banală a existenţei a 1024 de conductoare este prea costisitoare. Ţinând cont că informaţia transmisă pe bus nu are o frecvenţă mare, se propune reducerea numărului de conductoare prin multiplexarea biţilor la expeditor şi demultiplexarea acestora la destinaţie. Propuneţi o schemă de implementare. Care este numărul de conductoare necesar?

    Soluţie

    Soluţia constă în multiplexarea datelor la partea de emisie şi demultiplexarea acestora la partea de recepţie. Pentru o comunicare corectă, trebuie ca multiplexorul şi demultiplexorul să selecteze simultan aceeaşi valoare. Valoarea de selecţie de 10 biţi ar trebui şi ea transmisă de la sursă la destinaţie. În total, numărul de conectoare va fi redus de la 1024 la 11 (10 pentru selecţie şi 1 pentru date). Schema de principiu este prezentată în figura 12.9.

11. Demonstraţi că orice funcţie logică de 4 intrări poate fi implementată cu următoarele resurse: un decodificator de 3 biţi (DCD 3:8), un multiplexor 2:1, două porţi OR de maxim 4 intrări şi un inversor.

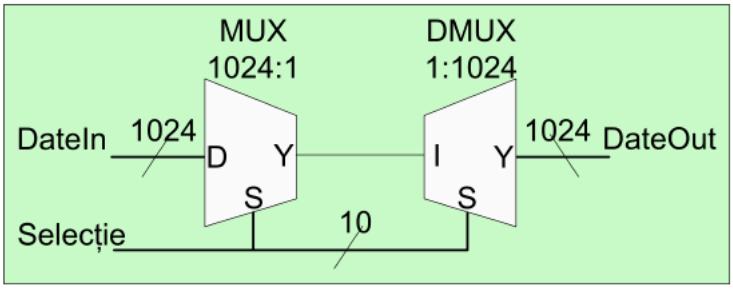


Figura 12.9 Reducerea numărului de conectoare pentru o transmisiune de date folosind circuite multiplexor/demultiplexor, problema 10.

Să se exemplifice soluţia pentru funcţiile:

a) $F = \sum(4, 5, 8, 9)$
b) $F = \sum(2, 7, 8, 9, 10, 12, 13)$
c) $F = \sum(0, 1, 3, 5, 9, 10, 11, 13, 15)$
d) $F = \sum(2, 3, 5, 6, 7, 8, 9, 10)$

Soluţie

Orice funcţie de 4 variabile se poate scrie sub o formă condensată cu o variabilă reziduu, ca:

$F(A, B, C, D) = A \cdot G_1(B, C, D) + \overline{A} \cdot G_0(B, C, D)$

Funcţiile $G_{1,0}(B, C, D)$ depind de aceleaşi 3 variabile. Acestea pot fi implementate cu un circuit decodificator de 3 biţi şi câte o poartă logică OR care reuneşte mintermii prezenţi în expresia funcţiei. În final, un multiplexor 2:1 are conectată intrarea A pe intrarea de selecţie şi funcţiile G_1 şi G_0 pe intrările de date. Numărul maxim de intrări pentru poartă OR este 4 (jumătate din numărul total de mintermi). Dacă expresia unei funcţii G_1 şi G_0 are mai mult de 4 mintermi, se implementează inversată şi cu ajutorul unui inversor se aduce la forma necesară. Dacă ambele funcţii G_1 şi G_0 au mai mult de 4 mintermi, inversorul se plasează pe ieşire, după multiplexor. Dacă ambele funcţii G_1 şi G_0 au maxim 4 mintermi, inversorul nu mai este necesar.

12. Se consideră funcţia $F(A, B, C, D) = \sum(1, 2, 3, 4, 5, 9, 13)$. Se cer:
a) Implementarea funcţiei cu porţi NAND. Evaluarea costului implementării.
b) Implementarea funcţiei cu porţi NOR. Evaluarea costului implementării.
c) Evaluarea existenţei hazardului combinaţional în implementările de la punctele **a)** sau **b)** şi implementarea funcţiei cu eliminarea hazardului combinaţional. Forme de undă pentru punerea în evidenţă a hazardului combinaţional.
d) Implementarea funcţiei cu MUX 8:1, variabilă reziduu D, A sau C.
e) Implementarea funcţiei cu MUX 4:1, variabile reziduu (C, D) sau (A, B).
f) Implementarea funcţiei cu DCD 3:8, MUX 2:1 şi porţi logice OR/NOR.
g) Implementarea funcţiei cu ROM 16x1. Implementarea funcţiei cu două circuite ROM 8x1 şi MUX 2:1.
h) Implementarea funcţiei cu circuit PLA generic.
i) Implementarea funcţiei cu circuit PAL generic.

12.4 Pentru cei ce vor să devină profesionişti

1. Proiectaţi un multiplicator binar de 2 biţi implementat cu porţi AND şi sumatoare binare.
Extindeţi multiplicatorul binar pentru a realiza înmulţiri cu numere reprezentate pe 3 biţi şi 4 biţi.

Soluţie

Algoritmul clasic de înmulţire presupune înmulţirea fiecărei cifre a unui operand cu fiecare cifră a celuilalt operand şi adunarea rezultatelor ţinând cont de ordinul acestora. Înmulţirea a două cifre binare este implementată cu o poartă AND ($0 \times 0 = 0, 0 \times 1 = 0, 1 \times 0 = 0, 1 \times 1 = 1$). Modul de plasare a rezultatelor de parţiale şi modul de adunare a acestora sunt prezentate în figura 12.10.

Figura 12.11 prezintă circuitele necesare pentru multiplicarea a două numere de câte 2 sau câte 3 biţi. Simbolurile de sumator reprezintă sumatoare complete de 1 bit.

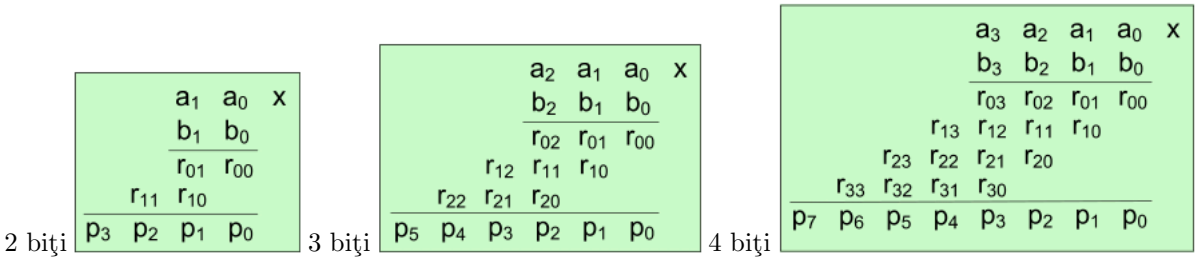


Figura 12.10 Înmulţirea numerelor de 2, 3 şi 4 biţi (problema 1)

2. Proiectaţi circuite de generare şi verificare de paritate pentru un bus de date de 8 biţi.

Soluţie

Generarea de paritate (pară sau impară) constă în determinarea unui bit suplimentar, pe baza biţilor datelor, astfel încât numărul total de biţi (biţi de date şi bit de paritate) să aibă paritatea impusă. De exemplu, în cazul parităţii *pare* suma biţilor de date plus bitul de paritate trebuie să fie un număr par. Similar, în cazul parităţii *impare*, suma biţilor de date plus bitul de paritate trebuie să fie un număr impar.

Verificarea parităţii constă în calcularea sumei tuturor biţilor plus bitul de paritate şi verificarea îndeplinirii parităţii aşteptate. În cazul apariţiei unei erori asupra biţilor transmişi, verificarea de paritate nu va mai fi satisfăcută şi se va semnala o *eroare de paritate*. Paritatea este un mod de protecţie limitată împotriva erorilor în mediul de transmisiune. Protecţia este activă doar în cazul apartiţiei unei singure erori. De exemplu, în cazul existenţei a două erori, verificarea de paritate produce un rezultat *fals pozitiv*, semnalând în mod eronat că datele sunt corecte. Schema bloc a circuitelor de generare şi verificare de paritate este prezentată în figura 12.12. În cazul parităţii pare, 8 biţi de date, ecuaţiile ieşirilor circuitelor combinaţionale sunt următoarele:

Generare de paritate:
$P_{par} = 0$, dacă suma $D_7 + D_6 + D_5 + D_4 + D_3 + D_2 + D_1 + D_0$ este un număr par
$P_{par} = 1$, dacă suma $D_7 + D_6 + D_5 + D_4 + D_3 + D_2 + D_1 + D_0$ este un număr impar

Verificare de paritate:
$ERR_{par} = 0$, dacă suma $D_7 + D_6 + D_5 + D_4 + D_3 + D_2 + D_1 + D_0 + P_{par}$ este un număr par
$ERR_{par} = 1$, dacă suma $D_7 + D_6 + D_5 + D_4 + D_3 + D_2 + D_1 + D_0 + P_{par}$ este un număr impar

Ţinând cont că poarta logică XOR poate fi interpretată ca un circuit de generare a parităţii impare (produce la ieşire 1 în cazul dacă un număr impar de intrări sunt egale cu 1), se deduc următoarele ecuaţii logice:
$P_{par} = D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0$
$ERR_{par} = D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0 \oplus P_{par}$

Similar,
$P_{impar} = \overline{D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0}$
$ERR_{impar} = \overline{D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0 \oplus P_{impar}}$

3. Proiectaţi un singur circuit care să poată fi utilizat atât pentru generarea parităţii cât şi pentru verificarea acesteia în cazul unei transmisiuni de 8 biţi de date, atât pentru paritate pară cât şi pentru paritate impară.

Soluţie

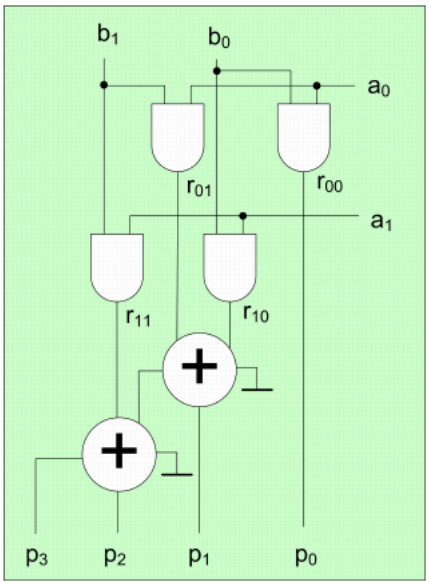
Conform problemei 2, rezultă că ecuaţiile logice pentru generare şi verificare de paritate sunt simlare: XOR cu mai multe intrări. În plus, schimbarea tipului parităţii (pară sau impară) presupune negarea expresiilor parităţii generate şi erorii determinate. Poarta XOR poate fi interpretată ca "inversor comandat".

În cazul a 8 intrări de date, expresiile menţionate la problema 2 pot fi rescrise pentru a putea fi implementate cu acelaşi circuit astfel:

$P_{par} = D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0$
$ERR_{par} = D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0 \oplus P_{par}$

Similar,
$P_{impar} = D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0 \oplus 1$
$ERR_{impar} = D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0 \oplus P_{impar} \oplus 1$

Rezultă că circuitul capabil de a genera şi verifica paritatea pară sau impară are funcţia logică XOR cu 10 intrări. În funcţie de rolul său, circuitul se va utiliza în conexiunile prezentate în tabelul următor.

2 biţi 

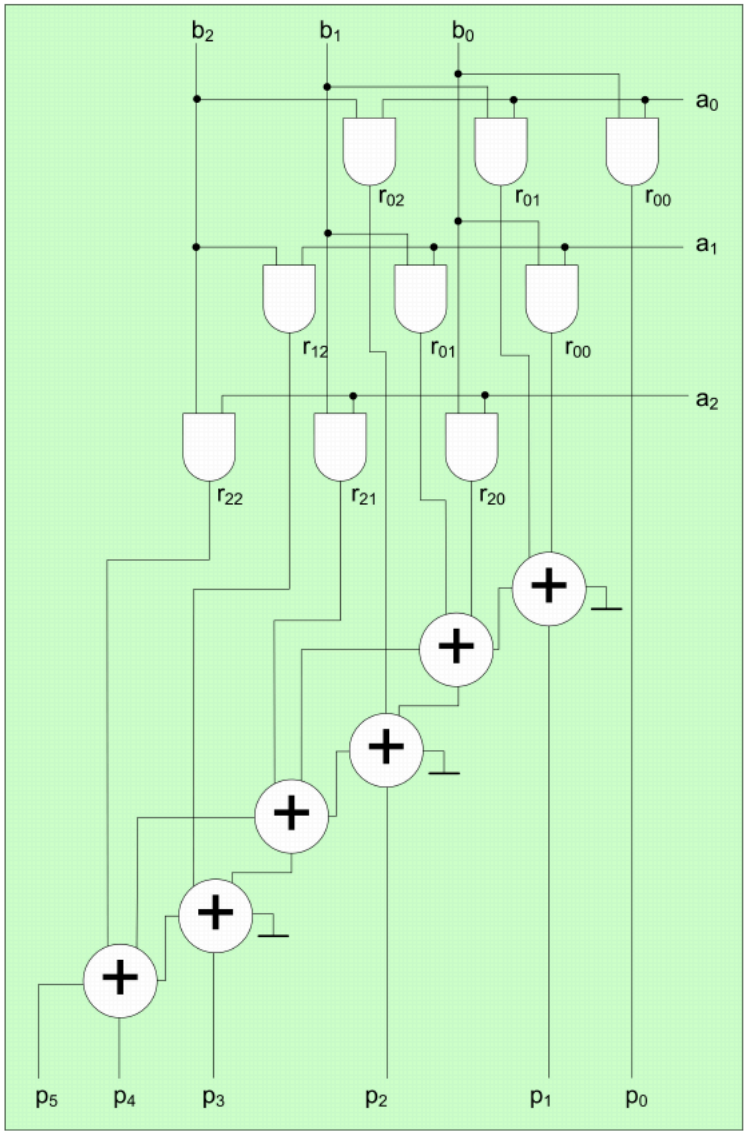
3 biţi 

Figura 12.11 Circuite de înmulţire a numerelor reprezentate pe 2 sau 3 biţi (problema 1).

Utilizare	Ieşire	I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0
Generare paritate pară	P_{par}	0	0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Verificare paritate pară	ERR_{par}	0	P_{par}	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Generare paritate impară	P_{impar}	1	0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Verificare paritate impară	ERR_{impar}	1	P_{impar}	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

4. Utilizând exclusiv sumatoare complete de un bit, proiectaţi un circuit care furnizează la ieşire numărul de biţi egali cu 1 în cadrul cuvântului de 7 biţi prezentat la intrare.

5. Utilizând exclusiv sumatoare complete de un bit, proiectaţi un circuit care furnizează la ieşire numărul de biţi egali cu 1 în cadrul cuvântului de 15 biţi prezentat la intrare.

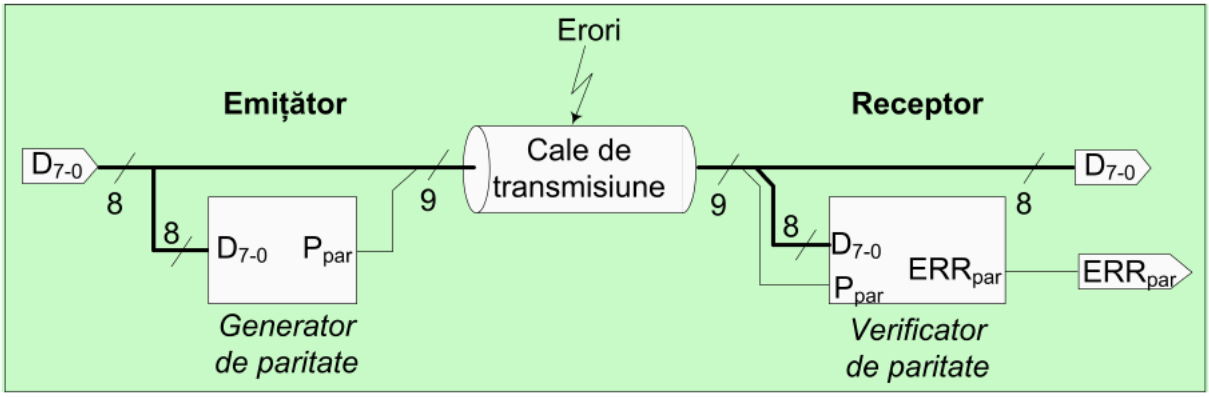


Figura 12.12 Schema bloc a circuitelor de generare şi verificare de paritate, referite la problema 2.