

Dan NICULA

# ELECTRONICĂ DIGITALĂ

## Carte de învățatură 2.0



Editura Universității *TRANSILVANIA* din Brașov  
ISBN 978-606-19-0563-8

2015

## Lecția 10

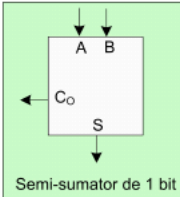
# Circuite aritmetice

### 10.1 Noțiuni teoretice

Circuitele logice combinaționale implementează funcții logice. Însă funcțiile logice pot fi interpretate ca funcții aritmetice aplicate asupra unor date binare ce codifică valori numerice.

*Semi-sumatorul de 1 bit* este un circuit logic combinațional cu două intrări și două ieșiri ce poate fi considerat ca având funcția aritmetică de a aduna 2 biți (operanzii) și a genera rezultatul pe un bit ( $S$ ) și transportul la bitul de ordin superior ( $C_O$ ). Simbolul bloc și tabelul de adevăr ale semi-sumatorului de 1 bit sunt prezentate în figura 10.1.

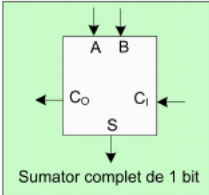
Intrări		Ieșiri	
$A$	$B$	$C_O$	$S$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



**Figura 10.1** Circuitul semi-sumator de 1 bit: tabelul de adevăr și simbolul bloc.

*Sumatorul complet de 1 bit* este un circuit logic combinațional cu trei intrări și două ieșiri ce poate fi considerat ca având funcția aritmetică de a aduna 2 biți (operanzii) plus transportul de la bitul inferior ( $C_I$ ) și a genera rezultatul pe un bit ( $S$ ) și transportul la bitul de ordin superior ( $C_O$ ). Simbolul bloc și tabelul de adevăr ale sumatorului complet de 1 bit sunt prezentate în figura 10.2.

Intrări			Ieșiri	
$A$	$B$	$C_I$	$C_O$	$S$
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



**Figura 10.2** Circuitul sumator complet de 1 bit: tabelul de adevăr și simbolul bloc.

Valorile numerice sunt codate în binar, pe unul sau mai mulți biți. Această lecție introduce noțiunile asociate operațiilor aritmetice de adunare, scădere și comparare cu operatori numere întregi. Numerele întregi pozitive sunt codate în binar. Numerele întregi negative sunt codate în complement față de 2.



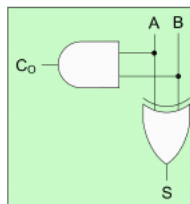
## 10.2 Pentru cei ce vor doar să promoveze examenul

1. Implementați cu porți logice un circuitul semi-sumator de un bit.

*Soluție*

Circuitul semi-sumator de 1 bit are două intrări asociate celor 2 biți de date  $A$ ,  $B$  și generează două ieșiri: suma  $S$  și transportul de ieșire  $C_O$ . Tabelul de adevăr asociat semi-sumatorului de 1 bit și structura celulei semi-sumator sunt prezentate în figura 10.3.

Intrări		Ieșiri	
$A$	$B$	$C_O$	$S$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



**Figura 10.3** Celula semi-sumator de 1 bit: tabelul de adevăr și structura.

Funcțiile ieșirilor sunt:

$$S = A \oplus B$$

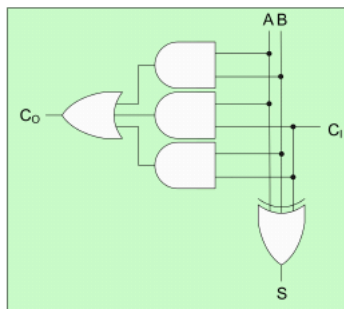
$$C_O = A \cdot B$$

2. Implementați cu porți logice circuitul sumator complet de un bit.

*Soluție*

Circuitul sumator complet de 1 bit are trei intrări asociate celor 2 biți de date  $A$ ,  $B$  și transportului de intrare  $C_I$ . Celula generează două ieșiri: suma  $S$  și transportul de ieșire  $C_O$ . Tabelul de adevăr asociat sumatorului de 1 bit și structura acestuia sunt prezentate în figura 10.4.

Intrări			Ieșiri	
$A$	$B$	$C_I$	$C_O$	$S$
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



**Figura 10.4** Celula sumator de 1 bit: tabelul de adevăr și structura.

Funcțiile ieșirilor sunt:

$$S = A \oplus B \oplus C_I$$

$$C_O = A \cdot B + A \cdot C_I + B \cdot C_I$$

3. Implementați un sumator complet de un bit cu un circuit decodificator de 3 biți.

*Soluție*

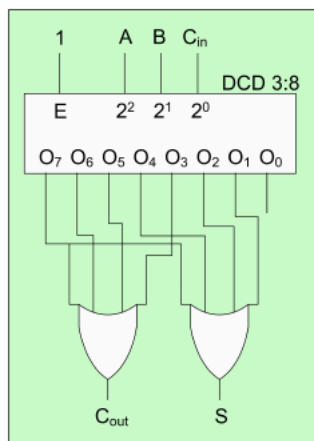
Funcțiile de transfer ale sumatorului complet de 1 bit sunt:

$$S = A \oplus B \oplus C_{in} = \sum(1, 2, 4, 7)$$

$$C_{out} = A \cdot B + A \cdot C_{in} + B \cdot C_{in} = \sum(3, 5, 6, 7)$$

Implementarea, prezentată în figura 10.5, utilizează un decodificator pentru generarea tuturor mintermilor și câte o poartă OR pentru fiecare funcție de ieșire.





**Figura 10.5** Sumator complet de 1 bit implementat cu decodificator.

4. Proiectați un sumator complet de 1 bit utilizând multiplexoare 4:1.

*Soluție*

Tabelul de adevăr al sumatorului complet de 1 bit este:

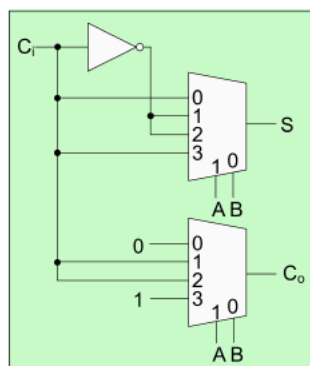
$A$	$B$	$C_i$	$C_o$	$S$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Pe baza tabelului de adevăr rezultă expresiile funcțiilor:

$$S(A, B, C_i) = m_0 \cdot C_i + m_1 \cdot \overline{C_i} + m_2 \cdot \overline{C_i} + m_3 \cdot C_i$$

$$C_o(A, B, C_i) = m_0 \cdot 0 + m_1 \cdot C_i + m_2 \cdot C_i + m_3 \cdot 1, \text{ unde } m_i(A, B).$$

Rezultă implementarea cu MUX 4:1 prezentată în figura 10.6.



**Figura 10.6** Sumator complet de 1 bit implementat cu multiplexor 4:1.

### 10.3 Pentru cei ce vor să învețe

1. Proiectați cu porți logice un circuit care convertește un număr de 4 biți în numărul negat (reprezentat în complement față de 2).

*Soluție*

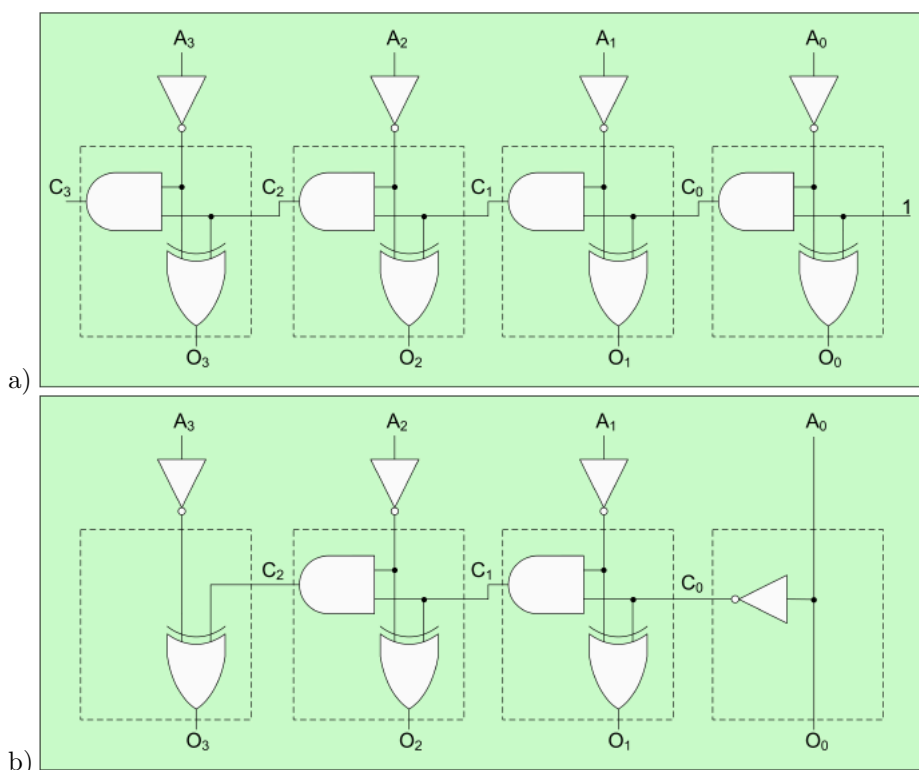
Complementul față de 2 al unui număr binar se poate afla prin negarea bit cu bit a numărului respectiv și adunarea cu 1. Operația se poate implementa cu ajutorul a 4 inversoare și a 4 semi-sumatoare, ca în figura 10.7-a.

Circuitul se poate optimiza dacă se observă că:

- semi-sumatorul cel mai puțin semnificativ are transport inițial tot timpul 1. Ca efect, circuitul se poate simplifica:

$$O_0 = \overline{A_0} \oplus 1 = A_0 \text{ și } C_0 = \overline{A_0} \cdot 1 = \overline{A_0}$$

- semi-sumatorul cel mai semnificativ nu trebuie să genereze transport de ieșire.



**Figura 10.7** Circuit de conversie a unui număr în numărul negat (reprezentat în complement față de 2).

În final, circuitul prezentat în figura 10.7-b constă din 4 inversoare, 3 porți XOR cu 2 intrări și 2 porți AND cu 2 intrări. Se observă că  $O_0 = A_0$ , adică circuitul de complementare față de 2 păstrează paritatea.

2. Proiectați un circuit de "vot majoritar" cu 3 intrări și implementați-l cu sumator complet de 1 bit. Ieșirea circuitului are valoarea logică a majorității intrărilor.

*Soluție*

Tabelul de adevăr al circuitului "vot majoritar" este prezentat în figura 10.8.

Funcția circuitului este:

$$VOT = V_2 \cdot V_1 + V_2 \cdot V_0 + V_1 \cdot V_0$$

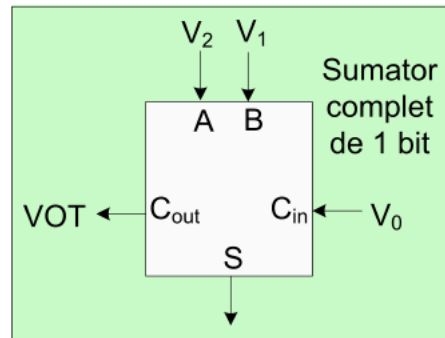
Comparând funcția "vot majoritar" cu funcțiile implementate de un sumator complet de 1 bit, se observă identitatea ecuațiilor funcției "vot majoritar" și a ieșirii de transport a sumatorului complet de 1 bit:

$$C_o = A \cdot B + A \cdot C_i + B \cdot C_i$$

Funcția "vot majoritar" poate fi implementată cu un sumator complet de 1 bit dacă acesta este conectat ca în figura 10.8.



$V_2$	$V_1$	$V_0$	VOT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



**Figura 10.8** Funcția ”vot majoritar”: tabelul de adevăr și implementarea cu sumator complet de 1 bit.

3. Să se proiecteze un circuit scăzător de 2 biți, utilizând două circuite scăzătoare complete de 1 bit.

*Soluție*

Scăzătorul de ordin inferior scade biții mai puțin semnificativi ai operandilor și are bitul de împrumut de intrare egal cu 0. Scăzătorul de ordin superior scade biții mai semnificativi ai operandilor și are bitul de împrumut de intrare provenit de la bitul de împrumut al scăzătorului de ordin inferior.

4. Să se proiecteze un circuit sumator de 4 biți, utilizând patru circuite sumatoare complete de 1 bit.
5. Să se proiecteze un circuit sumator/scăzător de 4 biți. Selecția funcției se va face cu o intrare suplimentară.

*Soluție*

Implementarea pornește de la observația că diferența numerelor reprezentate în complement față de 2 se poate scrie ca:

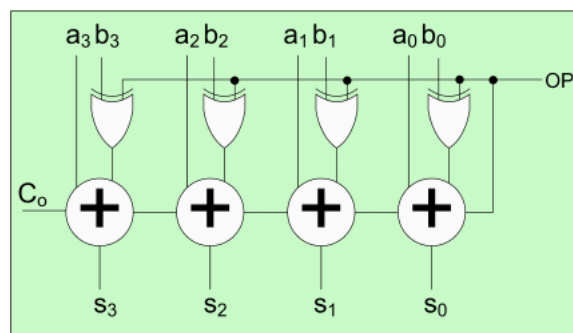
$A - B = A + (-B) = A + \overline{B} + 1$ , unde s-a notat cu  $+$  operatorul de adunare binară.

Rezultă că cele două operații de adunare și de scădere pot fi implementate cu același circuit conform expresiilor:

$A + B = A + B + 0 = A + B \oplus OP + OP$ , (pentru  $OP = 0$ , sumă) și

$A - B = A + \overline{B} + 1 = A + B \oplus OP + OP$ , (pentru  $OP = 1$ , diferență).

Pentru implementare s-a folosit observația că poarta XOR realizează o ”negare comandată” ( $B \oplus OP = B$  dacă  $OP = 0$  și  $B \oplus OP = \overline{B}$  dacă  $OP = 1$ ). Circuitul propus este prezentat în figura 10.9.



**Figura 10.9** Circuit de adunare/scădere pe 4 biți implementat cu 4 sumatoare complete de 1 bit.

6. Proiectați o unitate logico-aritmetică, ALU (Engl. ”Arithmetic Logic Unit”) pe 4 biți care implementează următoarele operații:



Selecția			Ieșiri	Operația
$S_2$	$S_1$	$S_0$	$\{C_o, REZ_{[3:0]}\}$	
0	0	0	0000	reset
0	0	1	$B - A$	scădere
0	1	0	$A - B$	scădere
0	1	1	$A + B$	adunare
1	0	0	$A \oplus B$	XOR
1	0	1	$A + B$	OR
1	1	0	$A \cdot B$	AND
1	1	1	$A$	trece $A$

În plus față de cei 4 biți de date de ieșire, ALU mai are o ieșire  $C_o$ , cu semnificația ”depășire”, activată doar în cazul operațiilor aritmetice.

Care sunt valorile rezultatelor la ieșirea ALU, dacă la intrare se aplică operandii  $A_{[3:0]} = 1101$  și  $B_{[3:0]} = 0101$ ?

7. Proiectați o unitate logico-aritmetică pe 1 bit care are simbolul și tabela de funcționare prezentate în figura 10.10. ALU are două intrări de date ( $A$  și  $B$ ), o intrare de transport relevantă la operațiile aritmetice și 2 biți de selecție a operației realizate. În plus față de ieșirea de rezultat  $R$  mai există o ieșire de transport  $C_o$  a cărei valoare este setată în cazul operațiilor aritmetice (în rest  $C_o = 0$ ).

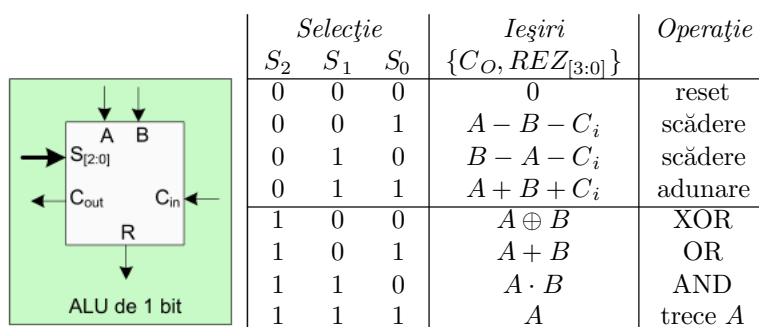


Figura 10.10 ALU pe 1 bit: simbol și tabel de funcționare.

### Soluție

ALU implementează 8 funcții. Cele 3 intrări de selecție ale funcției  $S[2:0]$  se conectează pe intrările de selecție ale unui MUX 8:1. Ieșirea multiplexorului va fi ieșirea ALU,  $R$ . Intrările multiplexorului sunt conectate conform tabelului de funcționare astfel:  $I_0 = 0$ ,  $I_4 = A \oplus B$ ,  $I_5 = A + B$ ,  $I_6 = A \cdot B$ ,  $I_7 = A$ .

Operațiile aritmetice se pot implementa cu circuite aritmetice sumatoare/scăzătoare de 1 bit. Ieșirile de date ale acestora se conectează la intrările multiplexorului  $I_1, I_2, I_3$ . Se poate face o optimizare pentru a se utiliza același circuit de scădere în cazurile  $S = 1$  și  $S = 2$ , dacă la intrările de date se plasează două multiplexoare 2:1 având datele  $A$  și  $B$  inversate și pe selecție  $S[0]$ .

Bitul de transport  $C_o$  poate fi generat fie cu un multiplexor 8:1 cu selecție  $S[2:0]$  și date provenind de la bitul de transport al circuitelor de adunare/scădere, fie cu un MUX 4:1 (selecție  $S[1:0]$ ) urmat de un MUX 2:1 (selecție  $S[2]$ ).

8. Utilizați module de tipul ”ALU pe 1 bit” descrise la problema 7 pentru a proiecta o unitate logico-aritmetică pe 8 biți.
9. Proiectați un circuit de incrementare (adunare cu 1) a numerelor reprezentate pe 4 biți, utilizând 4 semi-sumatoare de 1 bit.

### Soluție

Un semi-sumator adună doi biți de date și prezintă la ieșirea circuitului suma lor, reprezentată pe doi biți. Se poate considera că cel mai puțin semnificativ bit este rezultatul iar cel mai semnificativ bit este bitul de transport către un sumator de ordin superior.

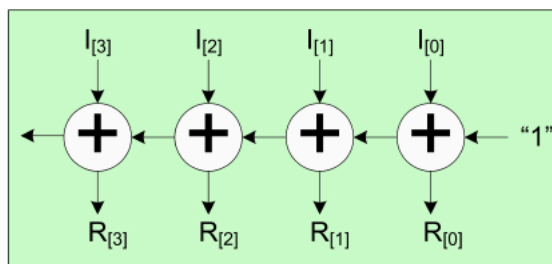
Circuitul de incrementare se poate realiza cu 4 semi-sumatoare de 1 bit dacă acestea se conectează astfel:

- Cel mai puțin semnificativ semi-sumator adună 1 la cel mai puțin semnificativ bit al operandului. Rezultatul reprezintă bitul cel mai puțin semnificativ iar transportul devine unul din operandii semi-sumatorului de ordin superior.



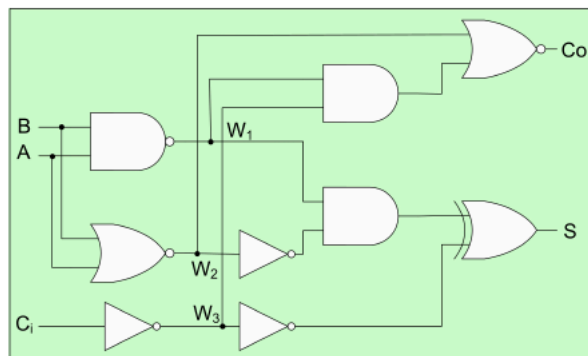
- Celelalte semi-sumatoare adună un bit al operandului cu transportul de la bitul inferior și produc câte un bit de rezultat și un transport către semi-sumatorul de ordin superior.

Circuitul este prezentat în figura 10.11.



**Figura 10.11** Circuit de incrementare pe 4 biți realizat cu 4 semi-sumatoare.

10. Verificați dacă structura de porți logice din figura 10.12 implementează un sumator complet de un bit.



**Figura 10.12** Circuit referit la problema 10.

#### Soluție

Ecuatiile logice ale ieșirilor sunt:

$$W_1 = \overline{A \cdot B}$$

$$W_2 = \overline{A + B}$$

$$W_3 = \overline{C}$$

$$C_o = \overline{W_2 + (W_1 \cdot W_3)}$$

$$S = (W_1 \cdot \overline{W_2}) \oplus \overline{W_3}$$

Din tabelul de adevăr prezentat în continuare rezultă că valorile ieșirilor  $C_o$  și  $S$  corespund ieșirilor unui sumator complet de 1 bit.

$A$	$B$	$C_{in}$	$W_1$	$W_2$	$W_3$	$W_1 \cdot W_3$	$W_1 \cdot \overline{W_2}$	$C_o$	$S$
0	0	0	1	1	1	1	0	0	0
0	0	1	1	1	0	0	0	0	1
0	1	0	1	0	1	1	1	0	1
0	1	1	1	0	0	0	1	1	0
1	0	0	1	0	1	1	1	0	1
1	0	1	1	0	0	0	1	1	0
1	1	0	0	0	1	0	0	1	0
1	1	1	0	0	0	0	0	1	1

## 10.4 Pentru cei ce vor să devină profesioniști

1. Implementați cu porți logice circuitul sumator complet cu considerarea *generării* și a *propagării* transportului.





*Soluție*

Circuitul sumator complet de 1 bit are trei intrări asociate celor 2 biți de date  $A$ ,  $B$  și transportului de intrare  $C_I$ . Celula generează două ieșiri: suma  $S$  și transportul de ieșire  $C_O$ .

Funcțiile ieșirilor sunt:

$$S = A \oplus B \oplus C_I$$

$$C_O = A \cdot B + A \cdot C_I + B \cdot C_I$$

Transportul  $C_O$  produs de o celulă sumatoare poate fi generat de celula curentă sau poate fi transportul primit la intrare  $C_I$  care se propagă prin celulă. În această nuanță, se pot defini două variabile intermediare: *generare*  $G$  și *propagare*  $P$ .

Generarea transportului de ieșire are loc când  $A = B = 1$ , adică  $G = A \cdot B$ .

Propagarea transportului de la intrare la ieșire are loc când doar una din intrări este 1, adică  $P = A \oplus B$ .

Tabelul de adevăr asociat sumatorului de 1 bit este:

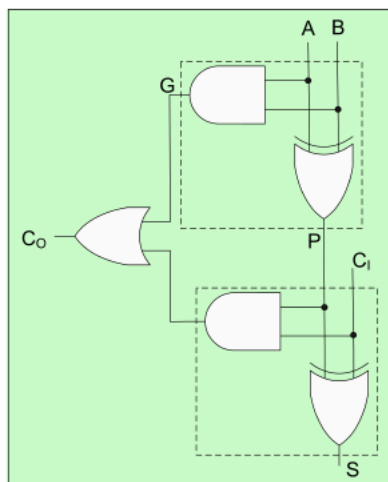
Intrări			Ieșiri		Variabile intermediare	
$A$	$B$	$C_I$	$C_O$	$S$	$P$	$G$
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	0	1	1	0
0	1	1	1	0	1	0
1	0	0	0	1	1	0
1	0	1	1	0	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	1

Ecuatiile sumatorului de 1 bit cu considerarea semnalelor intermediare  $G$  și  $P$  se pot scrie astfel:

$$S = A \oplus B \oplus C_I = P \oplus C_I$$

$$C_O = A \cdot B + A \cdot C_I + B \cdot C_I = A \cdot B + (A + B) \cdot C_I = A \cdot B + (A \oplus B) \cdot C_I = G + P \cdot C_I$$

Aceste relații determină implementarea din figura 10.13.



**Figura 10.13** Structura sumatorului complet de 1 bit, implementat cu semnale intermediare  $P$  și  $G$ .

- Proiectați cu porți logice un circuit combinațional care determină complementul față de 2 al numărului prezentat la intrare, pe orice număr de biți. Circuitul proiectat este unul iterativ, format dintr-un circuit de procesare a unui bit instanțiat pentru fiecare bit al operandului. Propuneți circuite pentru ambii algoritmi de complementare:
  - se complementează toți biții operandului și se adaugă 1;
  - se copiază biții de la dreapta la stânga până la primul 1, inclusiv, iar ceilalți biți se neagă.

*Soluție*

a) Pentru procesarea unui bit, se neagă acel bit și se adună cu o valoare provenită de la bitul anterior (mai puțin semnificativ), obținându-se bitul rezultat curent și un bit de transport. Pentru bitul 0, transportul de intrare

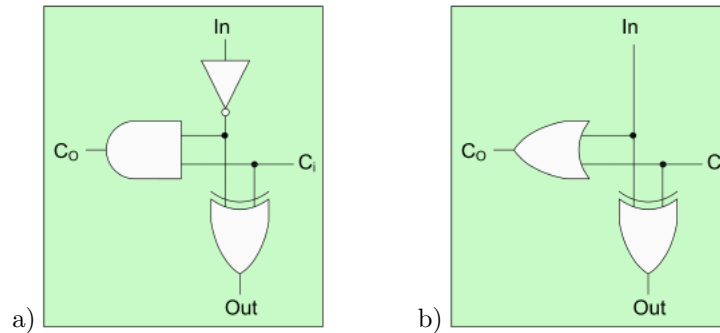


este  $C_i = 1$ . Biții se procesează în ordine, începând de la cel mai puțin semnificativ spre cel mai semnificativ. Circuitul pentru procesarea unui bit este prezentat în figura 10.14-a și se bazează pe ecuația aritmetică:  $\{C_o, Out\} = \overline{In} + C_i$ , unde cu  $+$  s-a notat operatorul de sumă aritmetică.

Rezultă ecuațiile logice:

$$C_o = \overline{In} \cdot C_i$$

$$Out = \overline{In} \oplus C_i$$



**Figura 10.14** Circuit iterativ de complementare (celulă pentru 1 bit): **a)** se complementează toți biții operandului și se adaugă 1 ( $C_{i[0]} = 1$ ), **b)** se copiază biții de la dreapta la stânga până la primul 1, inclusiv, iar ceilalți biți se neagă ( $C_{i[0]} = 0$ ).

**b)** Pentru procesarea unui bit se decide dacă acesta este negat sau nu pe baza unei informații provenite de la bitul de ordin inferior. Pentru bitul 0, transportul de intrare este  $C_i = 0$ . Biții se procesează în ordine, începând de la cel mai puțin semnificativ spre cel mai semnificativ. Circuitul pentru un bit are aceleași porturi și este caracterizat de următorul tabel de adevăr:

$In$	$C_i$	$Out$	$C_o$	Explicație
0	0	0	0	zero-uri inițiale, se copiază
0	1	1	1	bitul este 0, după ce deja s-a găsit un 1, deci se neagă
1	0	1	1	primul bit egal cu 1, se copiază și se activează transportul
1	1	0	1	bitul este 1, după ce deja s-a găsit un 1, deci se neagă

Rezultă funcțiile logice:

$$C_o = In + C_i, \text{ unde operatorul } + \text{ reprezintă OR logic și}$$

$$Out = In \oplus C_i$$

Circuitul pentru procesarea unui bit este prezentat în figura 10.14-b. Figura 10.15 prezintă circuitele de aflare a numărului negat, în complement față de 2, în cele două variante.

- Să se deducă funcțiile iterative ale unui circuit de incrementare a numerelor naturale reprezentate în binar. Să se implementeze circuitul pentru 4 biți.

*Soluție*

După minimizarea funcțiilor, utilizând diagramele V-K prezentate în figura 10.17, se obțin ecuațiile:

$$O_0 = \overline{I_0}$$

$$O_1 = \overline{I_1} \cdot I_0 + I_1 \cdot \overline{I_0} = I_1 \oplus I_0$$

$$O_2 = I_2 \cdot \overline{I_1} + I_2 \cdot \overline{I_0} + \overline{I_2} \cdot I_1 \cdot I_0 = I_2 \cdot (\overline{I_1} + \overline{I_0}) + \overline{I_2} \cdot I_1 \cdot I_0 = I_2 \cdot \overline{I_1 \cdot I_0} + \overline{I_2} \cdot I_1 \cdot I_0 = I_2 \oplus (I_1 \cdot I_0)$$

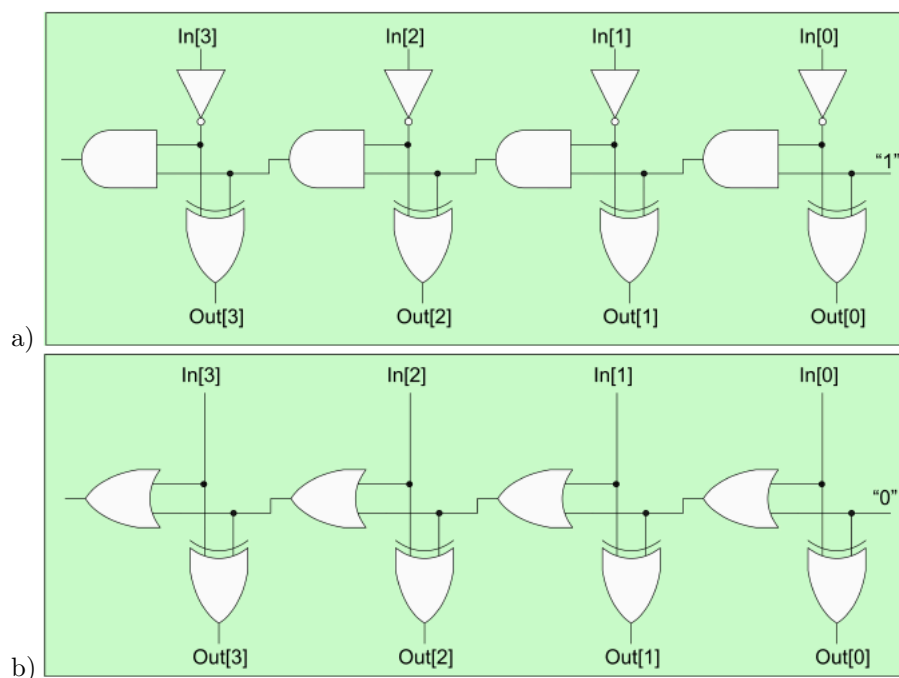
$$O_3 = I_3 \oplus I_2 \cdot I_1 \cdot I_0$$

Ecuațiile se pot deduce observând ca un bit al ieșirii este egal cu 1 fie dacă bitul de intrare este egal cu 0 și toți biții anteriori (mai puțin semnificativi) sunt 1, fie dacă bitul este egal cu 1, iar biții mai puțin semnificativi nu sunt toți egali cu 1. Altfel spus: funcția XOR între bitul curent și funcția AND între toți biții mai puțin semnificativi.

$$O_0 = \overline{I_0}$$

$$O_N = I_N \oplus (\prod_{i=0}^{N-1} I_i), \text{ pentru } \forall N > 0.$$





**Figura 10.15** Circuit de complementare pe 4 biți: **a)** se complementează toți biții operandului și se adaugă 1, **b)** se copiază biții de la dreapta la stânga până la primul 1, inclusiv, iar ceilalți biți se neagă.

4. Să se deducă funcțiile iterative ale unui circuit de decrementare a numerelor naturale reprezentate în binar.

*Soluție*

Se observă că în urma unei operații de decrementare (scădere cu 1) un bit de index  $N$  al rezultatului este egal cu 1 în două situații:

- toții biții mai puțin semnificativi ai operandului, cu indecși între  $N$  și 0 sunt egali cu 0, caz în care există un împrumut de la bitul de ordin  $N + 1$  sau
- bitul  $N$  este egal cu 1 și mai există cel puțin un bit egal cu 1 de index inferior, între  $N - 1$  și 0, caz în care bitul  $N$  nu este afectat de operația de decrementare.

Rezultă ecuațiile:

$$O_0 = \overline{I_0}$$

$$O_N = \overline{I_N} \cdot \overline{I_{N-1}} \cdot \dots \cdot \overline{I_0} + I_N \cdot (I_{N-1} + \dots + I_0) = \overline{I_N} \cdot \prod_{i=0}^{N-1} \overline{I_i} + I_N \cdot \sum_{i=0}^{N-1} I_i, \text{ pentru } \forall N > 0.$$

5. Proiectați următoarele circuite de înmulțire:

- circuit de înmulțire a două numere de 1 bit;
- circuit de înmulțire a două numere de câte 2 biți, utilizând porți AND cu 2 intrări și semisumatoare de 1 bit;
- circuit de înmulțire a două numere de câte 4 biți, utilizând porți AND cu 2 intrări și semisumatoare de 1 bit;
- circuit de înmulțire a două numere de câte 4 biți, utilizând circuite de înmulțire a câte 2 biți (similare celor descrise la punctul b).

6. Cum poate fi determinată depășirea în cazul operațiilor de adunare cu numere reprezentate în complement față de 2?

*Soluție*

În cazul operațiilor cu numere reprezentate în complement față de 2 asupra bitului de semn (cel mai semnificativ) se efectuează aceleași operații ca și asupra celorlalți biți. Valoarea bitului de semn (0 pentru rezultat pozitiv și 1 pentru rezultat negativ) va rezulta în urma prelucrărilor aritmetice. În cazul în care cei doi operanzi ai unei adunări au același semn, este posibil ca rezultatul să nu poată fi reprezentat pe același număr de biți. În acest caz se consideră că operația generează "depășire" a domeniului de reprezentare a numerelor. De exemplu, pe 8 biți, pot fi reprezentate numere întregi între  $-128$  și  $127$ . Prin însumarea numerelor  $100 + 120$  rezultatul nu poate fi reprezentat pe 8 biți.

```

100+  0_110_0100+
120   0_111_1000
===  =====
220   1_101_1100

```



Număr				Increment (+1)			
$I_3$	$I_2$	$I_1$	$I_0$	$O_3$	$O_2$	$O_1$	$O_0$
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

Figura 10.16 Tabelul de adevăr al circuitului de incrementare pe 4 biți.

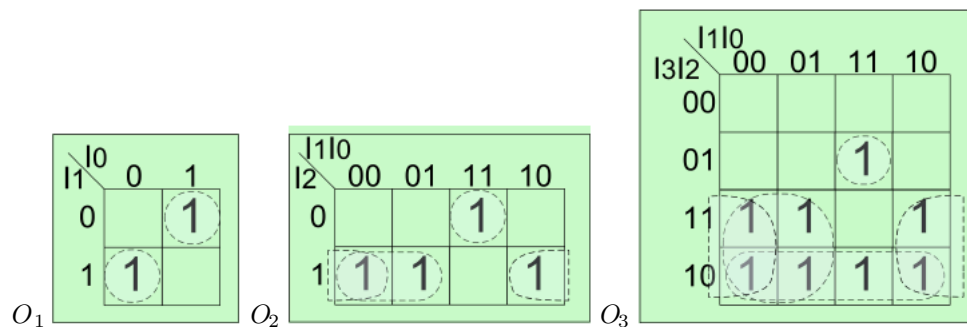


Figura 10.17 Diagramele V-K pentru circuitul de incrementare, problema 3.

Rezultatul binar 1101\_1100 este asociat în complement față de 2 numărului negativ  $-36$ . Greșeala provine din faptul că rezultatul corect 200 ar avea nevoie de 9 biți pentru a fi reprezentat în format complement față de 2. Determinarea depășirii se poate face comparând semnul rezultatului obținut cu semnul rezultatului așteptat. Dacă ambii operatori sunt pozitivi, atunci rezultatul trebuie să fie pozitiv, deci să aibă bitul de semn egal cu 0. Dacă ambii operatori sunt negativi, atunci rezultatul trebuie să fie negativ, deci să aibă bitul de semn egal cu 1. În cazul în care un operand este negativ și celălalt este pozitiv, nu se generează niciodată depășire iar rezultatul poate fi atât pozitiv, cât și negativ.

Circuitul de detecție a erorii de depășire are trei intrări: semnul celor 2 operanzi și semnul rezultatului. Tabelul de adevăr al funcției de eroare este prezentat în continuare. S-au notat  $S_A = A[7]$ ,  $S_B = B[7]$  semnele (bitul cel mai semnificativ) operanzilor și  $S_S = S[7]$  semnul sumei acestora.

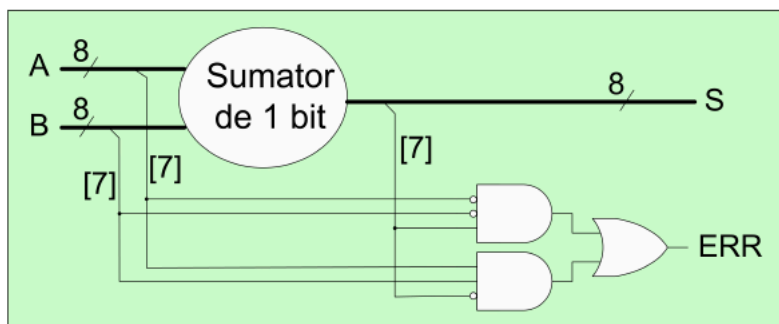
$S_A$	$S_B$	$S_S$	$ERR$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Ecuția logică a ieșirii este:



$$ERR = \overline{S_A} \cdot \overline{S_B} \cdot S_S + S_A \cdot S_B \cdot \overline{S_S}.$$

Implementarea circuitului este prezentată în figura 10.18.



**Figura 10.18** Circuit de depistare a erorii în cazul adunării numerelor în complement față de 2, referit la problema 6.

Este interesant de observat că bitul de eroare poate fi utilizat pentru a interpreta rezultatul în mod corect. În exemplul prezentat, rezultatul este  $1\_101\_1100|_{C_2} = -36$ . Însă, dacă se utilizează informația că de fapt rezultatul este eronat, acesta trebuie interpretat ca fiind o combinație binară având un bit suplimentar de aceeași valoare ca și operandii. În acest caz, rezultatul trebuie interpretat ca fiind  $0\_1101\_1100|_{C_2} = +220$ . Se observă că rezultatul corect se obține din rezultatul eronat prin adunarea numărului  $256|_{10} = 2^8$ , adică  $-36 + 256 = 220$ .

Aceeași corecție se face și în cazul sumei a doi operanzi negativi.

```
-107+  1_001_0101+
-105   1_001_0111
====  =====
-212   0_010_1100
```

Rezultatul este  $0\_010\_1100|_{C_2} = +44$ . Însă, dacă se utilizează informația că de fapt rezultatul este eronat, acesta trebuie interpretat ca fiind o combinație binară având un bit suplimentar de aceeași valoare ca și operandii. În acest caz, rezultatul trebuie interpretat ca fiind  $1\_0010\_1100|_{C_2} = -212$ . Se observă că rezultatul corect se obține din rezultatul eronat prin adunarea numărului  $-256|_{10}$ , adică  $44 + (-256) = -212$ .

