II.4. Minimizarea funcțiilor booleene prin metoda diagramelor Karnaugh

Metoda Veitch-Karnaugh

- oferă posibilitatea de a grupa vizual termenii conjuncție din FND pentru care se poate aplica unificarea
- pentru unificarea a doi termeni, aceștia trebuie să difere pe o singură variabilă
  - la un termen apare negată, la celălalt nenegată
- asemenea termeni devin vecini într-o diagramă Karnaugh

Structura diagramei Karnaugh

- tabel bidimensional
- numele variabilelor
  - pe linii, respectiv coloane
- zona etichetelor
  - etichetă - şir de n biţi
  - fiecare bit corespunde unei variabile (intrări)
  - apar toate combinațiile posibile de valori
- zona valorilor funcției (ieșiri)

Exemple de diagrame

2 variabile

A \ B	0	1
0	1	1
1		

3 variabile

A \ BC	00	01	11	10
0		1		1
1		1	1	

4 variabile

AB \ CD	00	01	11	10
00	1	1		
01				
11		1		1
10	1			1

Codul Grey

- etichetele nu se scriu în ordinea naturală, ci în ordinea Grey
- oricare două etichete consecutive, inclusiv prima şi ultima, diferă printr-un singur bit
  - 2 biți: 00, 01, 11, 10
  - 3 biți: 000, 001, 011, 010, 110, 111, 101, 100
  - 4 biți: 0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000

Adiacenţe în diagrame (1)

- două poziţii sunt adiacente dacă etichetele corespunzătoare diferă pe un singur bit
  - codul Grey translează adiacența în vecinătate
- pentru o funcţie de *n* variabile, o locaţie are *n* locaţii adiacente
  - $n < 5$: locațiile adiacente locației date se determină vizual (sus, jos, stânga, dreapta)
  - $n \geq 5$: şi alte adiacenţe decât cele vizibile direct

Adiacenţe în diagrame (2)

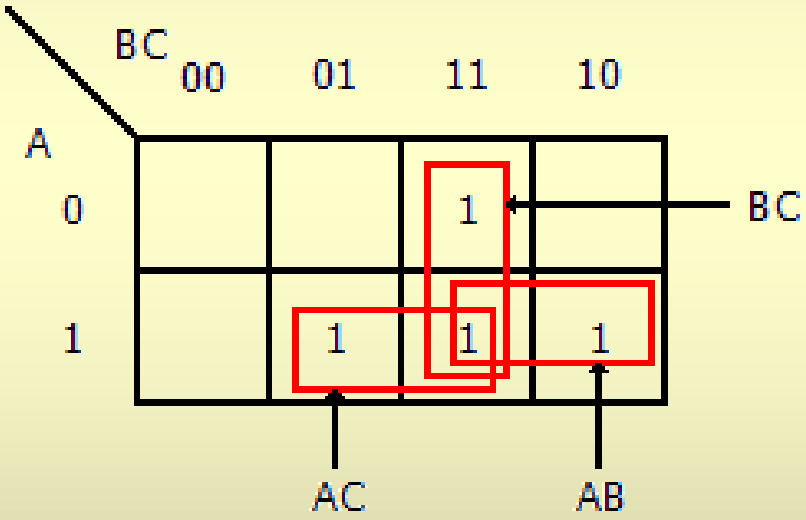
- pot fi mai mult de 2 locații adiacente
  - extinderea unificării la mai mult de 2 variabile
- în diagramele Karnaugh, acesta corespund unor blocuri de 2^k locații
  - putere a lui 2 atât pe linii, cât și pe coloane
    - inclusiv puterea 0
    - formă dreptunghiulară
  - pentru fiecare locaţie, blocul trebuie să conţină exact *k* locaţii adiacente cu ea

Minimizare Karnaugh

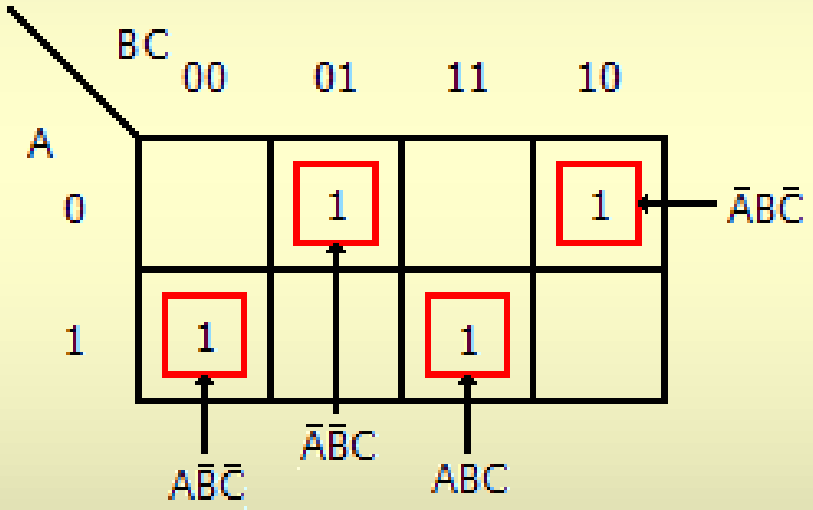
- se caută blocuri conţinând numai valori 1
  - corespunzătoare unor adiacențe (v. anterior)
  - blocuri cât mai mari şi mai puţine
- pentru fiecare bloc cu 2^k locaţii 1
  - avem un termen conjuncţie cu n-k variabile
  - conține variabilele cu valori constante pentru toate locaţiile din bloc
    - constant 0: variabilă negată; constant 1: nenegată
  - toți acești termeni sunt legați prin disjuncție

Exemple

majoritate din 3

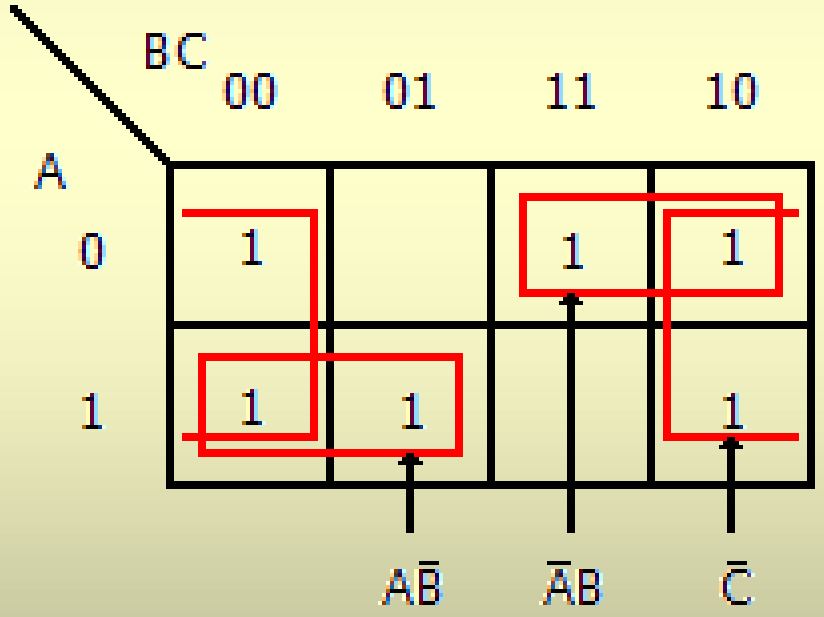


imparitate

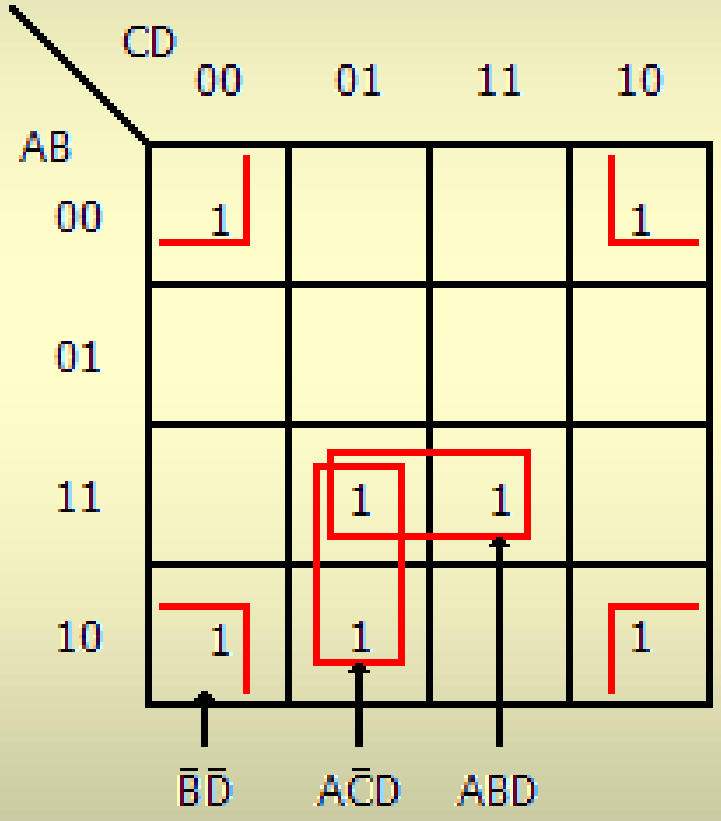


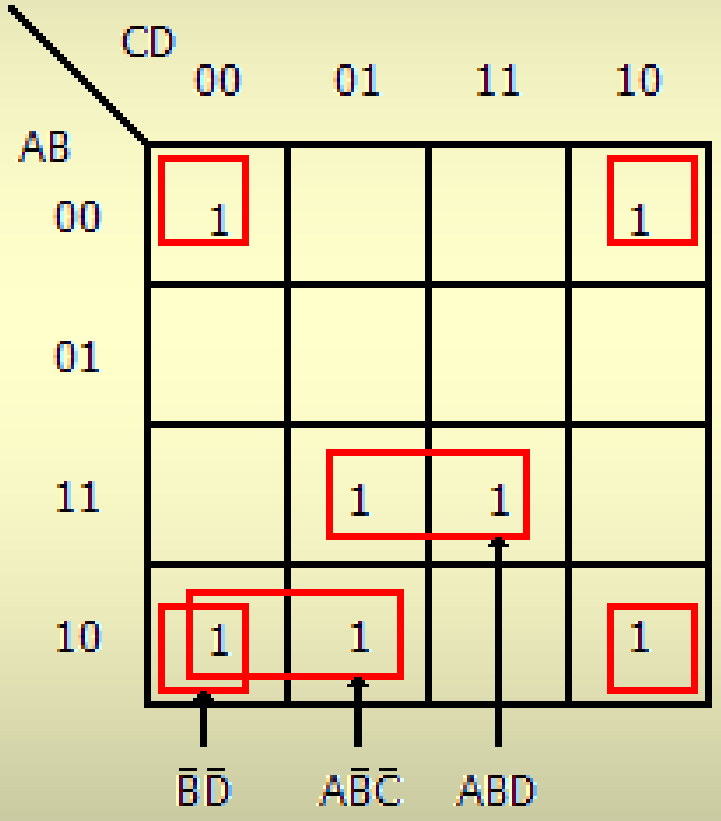
Adiacenţa liniilor/coloanelor extreme

$$f = \Sigma(0,2,3,4,5,6)$$



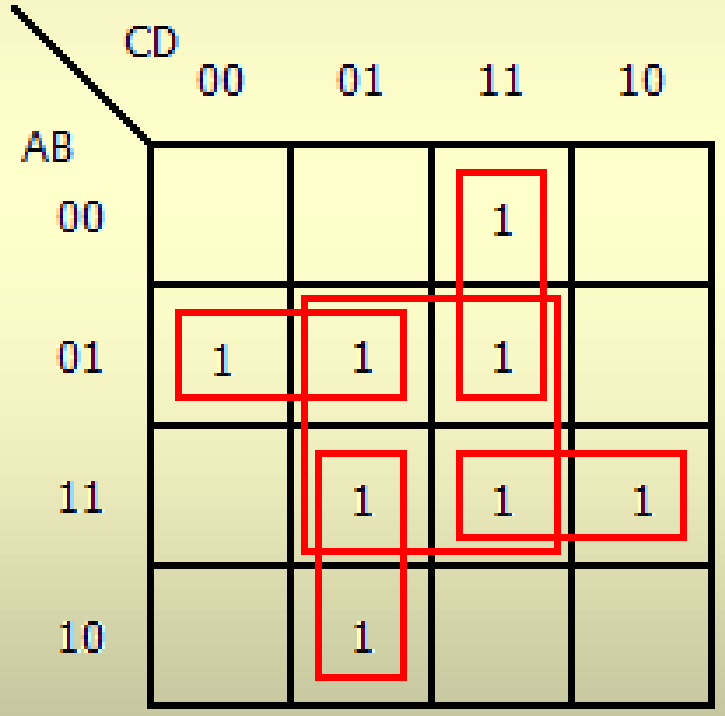
Expresia depinde de grupare



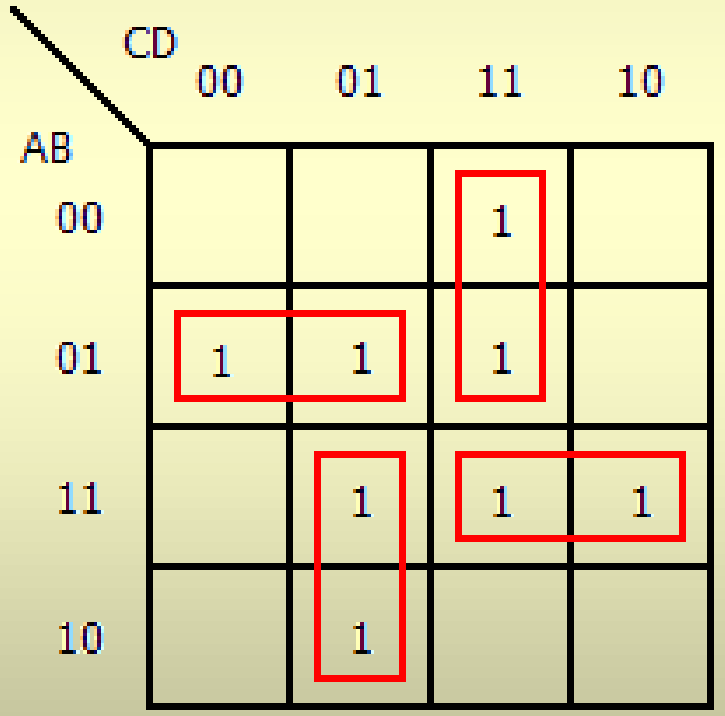


Evitarea redundanţelor

simplificare neminimală



simplificare minimală

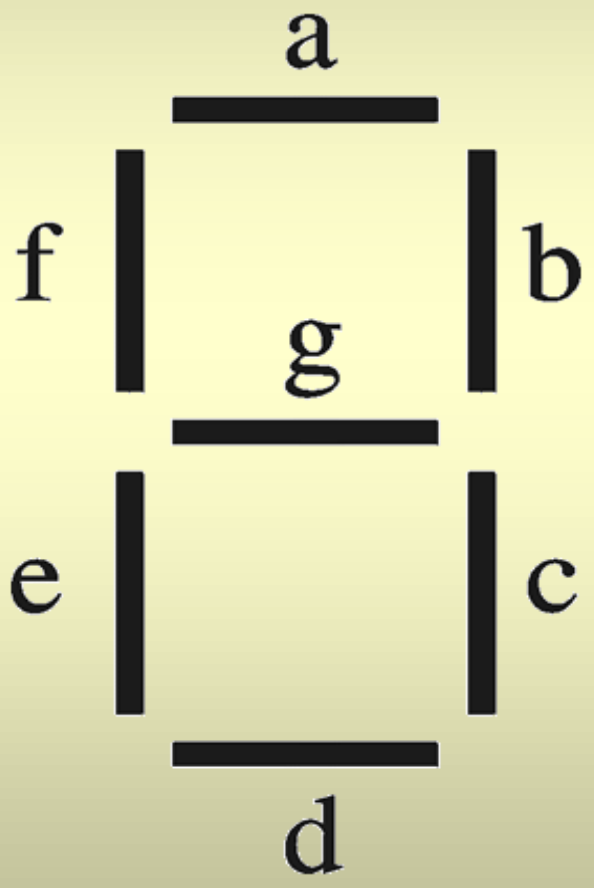


Combinaţii imposibile de valori

- anumite combinaţii de valori nu vor apărea niciodată la intrări
  - din definiția comportamentului dorit
  - dar diagrama trebuie completată pentru toate combinațiile de valori ale variabilelor
- în locațiile corespunzătoare acestor combinații se poate trece 0 sau 1
  - astfel încât să obținem o expresie cît mai simplă

Exemplu - afişaj zecimal

- afişaj cu 7 segmente
- selectarea segmentelor pentru fiecare cifră
  - 0 - stins
  - 1 - aprins
- comanda pe intrare - 4 variabile
  - o cifră zecimală se poate scrie pe 4 biți



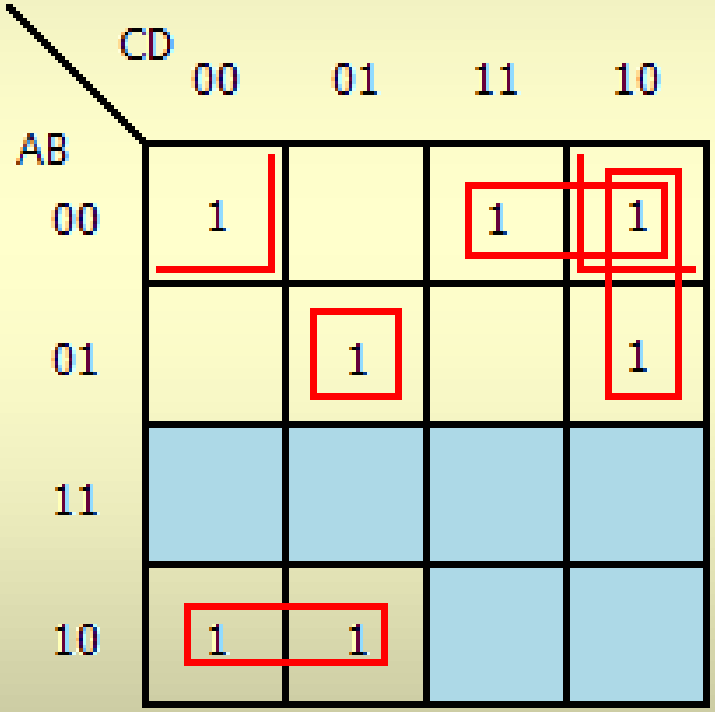
Segmentul *d* - tabel de adevăr

Nr	A	B	C	D	*d*
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	0

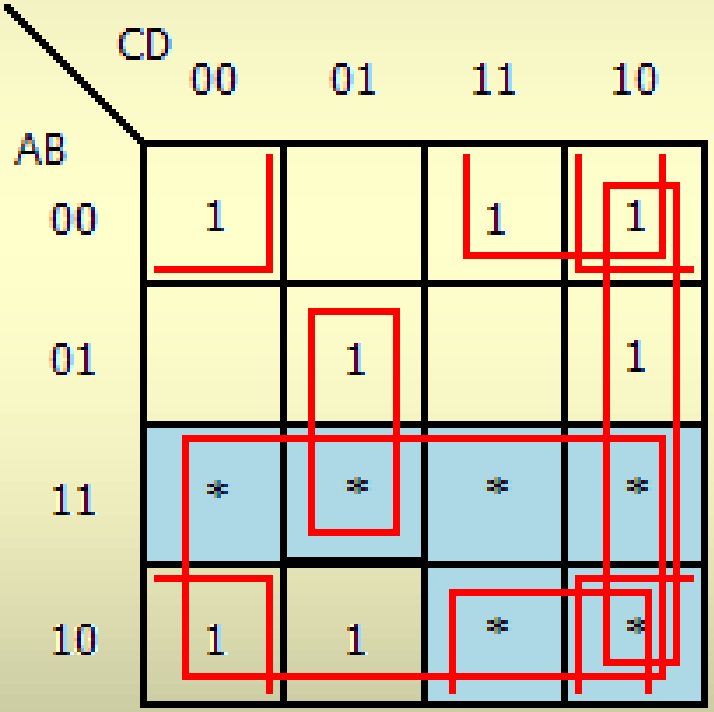
Nr	A	B	C	D	*d*
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	*
11	1	0	1	1	*
12	1	1	0	0	*
13	1	1	0	1	*
14	1	1	1	0	*
15	1	1	1	1	*

Expresii mai simple

"funcţionare de siguranţă"



combinaţii imposibile



Temă: comparator pe 2 biţi

- 4 variabile: A, B, C, D
- formează 2 numere
  - $N_1 = AB$
  - $N_2 = CD$
- 3 ieşiri - corespund valorilor de adevăr
  - $LT = (N_1 < N_2)$
  - $EQ = (N_1 = N_2)$
  - $GT = (N_1 > N_2)$

A	B	C	D	LT	EQ	GT
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

Temă: multiplicator pe 2 biţi

- 4 variabile: A, B, C, D
- formează 2 numere
  - $N_1 = AB$
  - $N_2 = CD$
- 4 ieşiri - formează produsul $N_1 \cdot N_2$

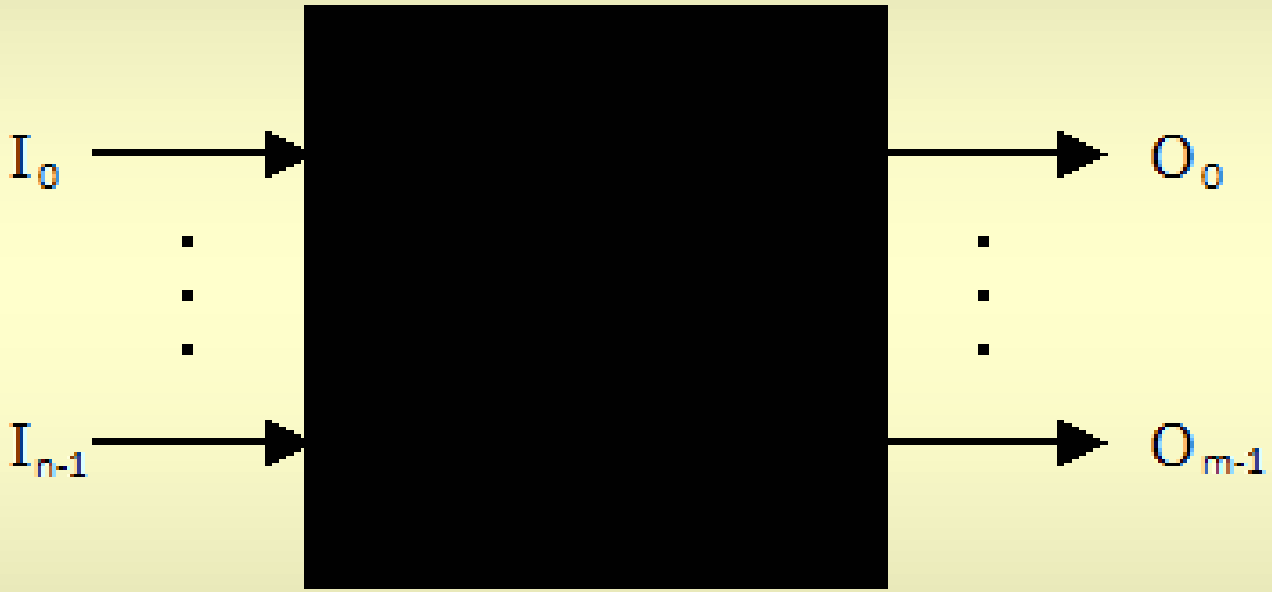
A	B	C	D	P8	P4	P2	P1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Temă: incrementare cu 1 BCD

- 4 variabile
  - formează un număr BCD
  - între 0 şi 9
- 4 ieşiri - numărul de la intrare incrementat
  - rezultatul este tot un număr BCD

I8	I4	I2	I1	O8	O4	O2	O1
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	0	0	0	0
1	0	1	0	*	*	*	*
1	0	1	1	*	*	*	*
1	1	0	0	*	*	*	*
1	1	0	1	*	*	*	*
1	1	1	0	*	*	*	*
1	1	1	1	*	*	*	*

II.5. Circuite combinaţionale

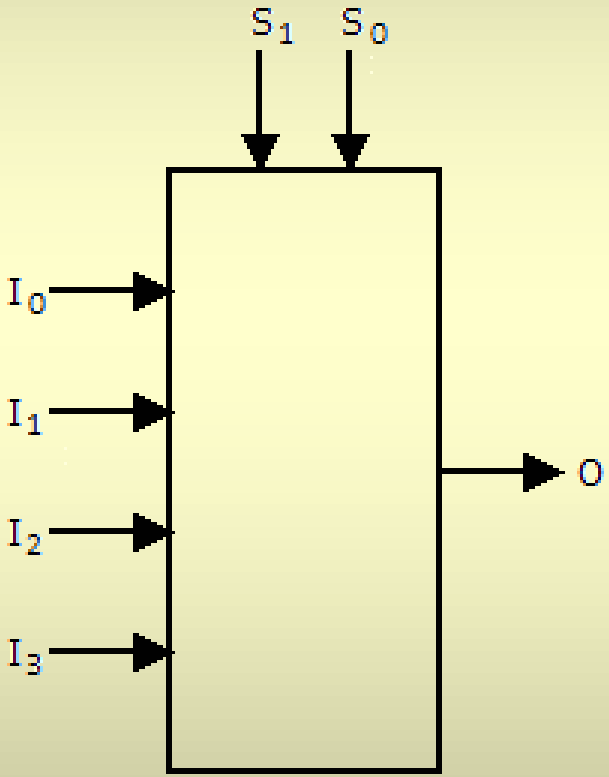


- valorile de la ieşire depind doar de valorile de la intrare din momentul respectiv

Multiplexorul

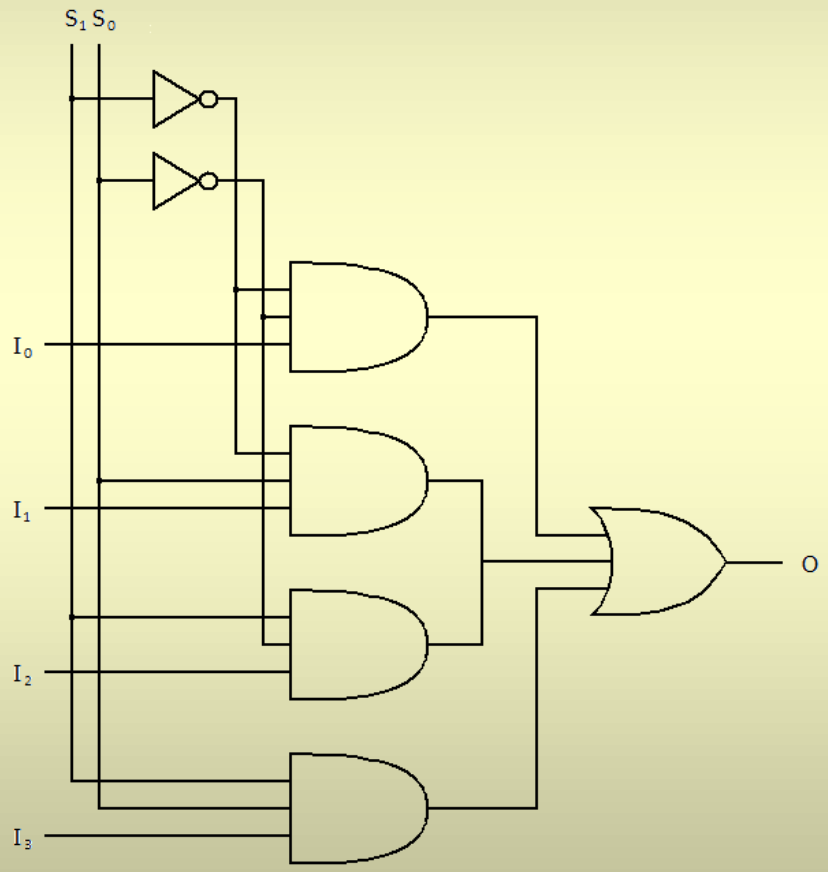
- 2^n intrări (de date)
- *n* intrări de selecţie (variabile de control)
  - biţi de control (de adresă)
- o singură ieşire
- fiecare intrare corespunde unui termen FND cu variabile de control
- una dintre intrări (bit) este selectată - devine valoare de ieşire

Multiplexor 4→1 (*n*=2)



S_1	S_0	O
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Diagrama logică (4→1)

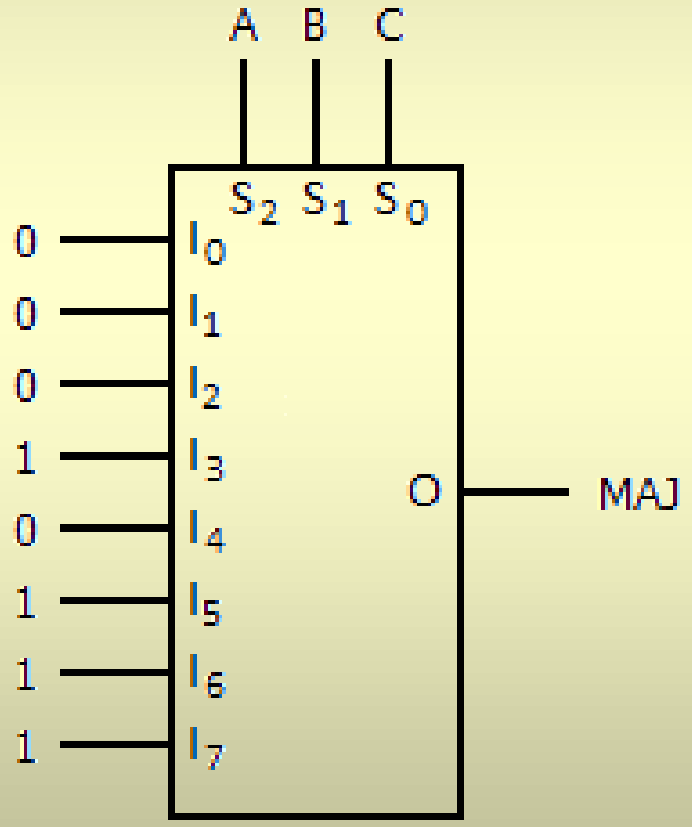


Implementare funcții booleene

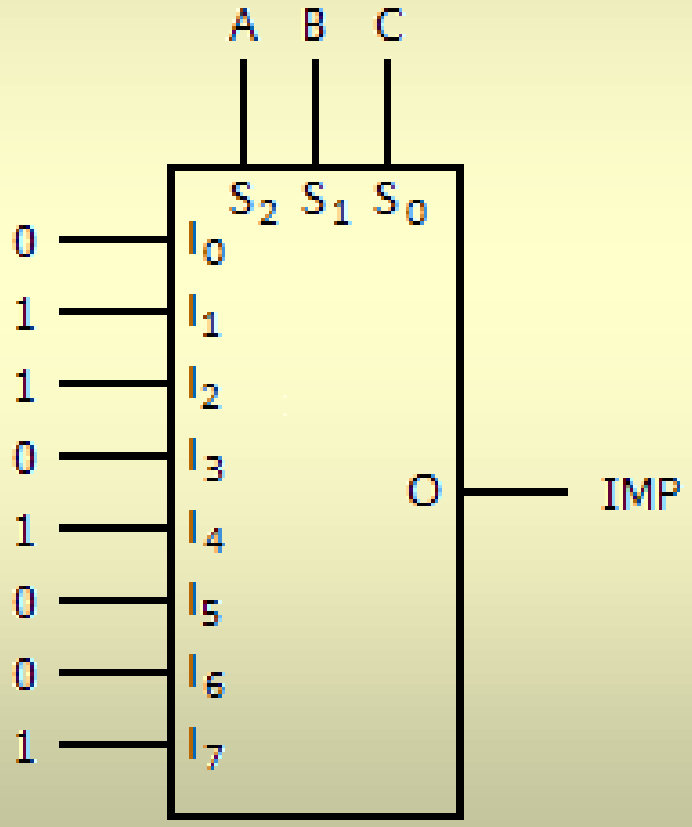
- intrările de selecție formează un număr
- care reprezintă indicele intrării de date care este selectată ca valoare de ieșire
- putem astfel implementa funcții booleene cu ajutorul multiplexorului
  - intrări de date - ieșirile corespunzătoare liniilor din tabelul de adevăr
  - intrări de selecție - intrările funcției booleene

Exemple

majoritate din 3



imparitate

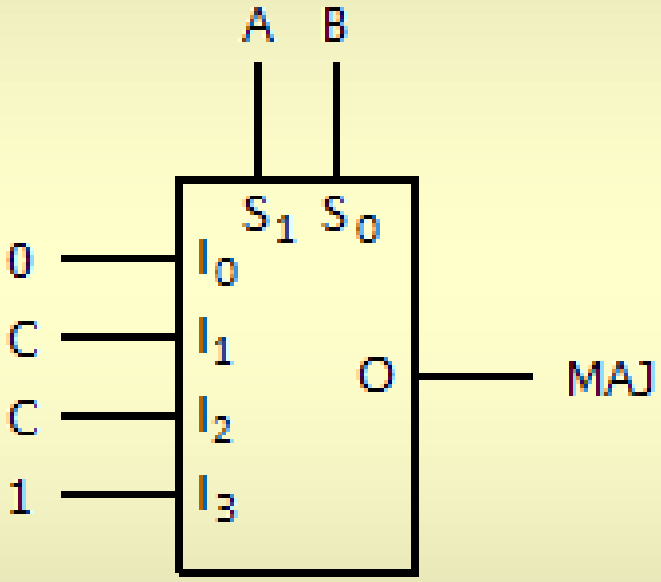


Implementare eficientă - *folding*

majoritate din 3

A	B	C	
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

A	B	
0	0	0
0	1	C
1	0	C
1	1	1

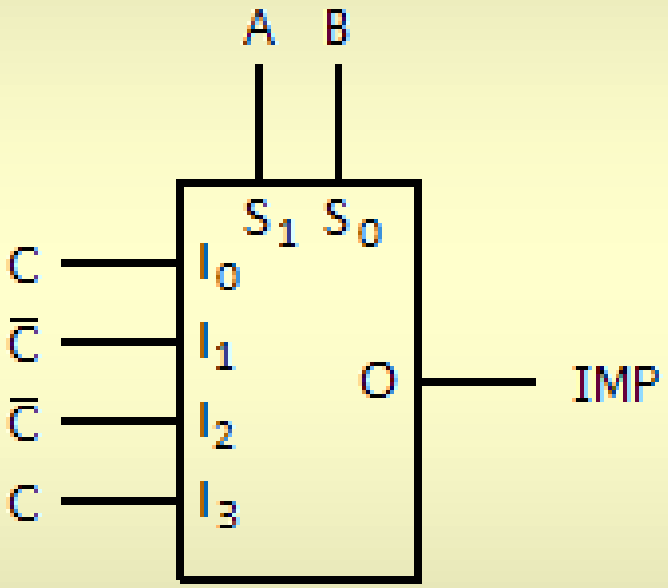


Implementare eficientă - *folding*

imparitate

A	B	C	
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

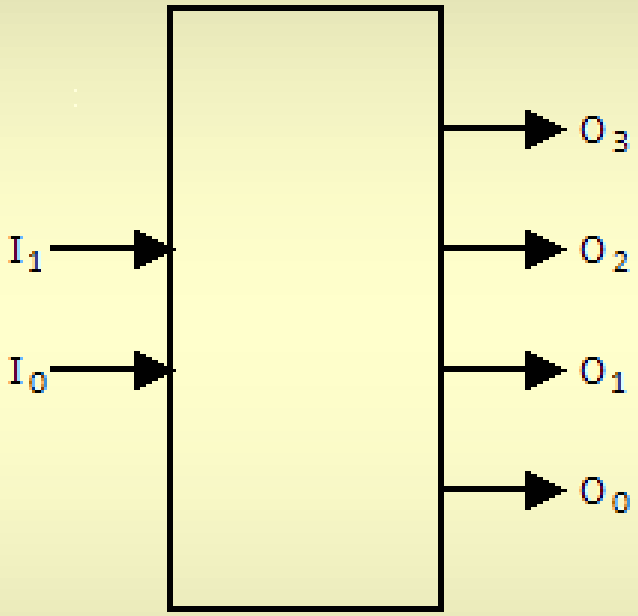
A	B	
0	0	C
0	1	$\bar{C}$
1	0	$\bar{C}$
1	1	C



Decodorul

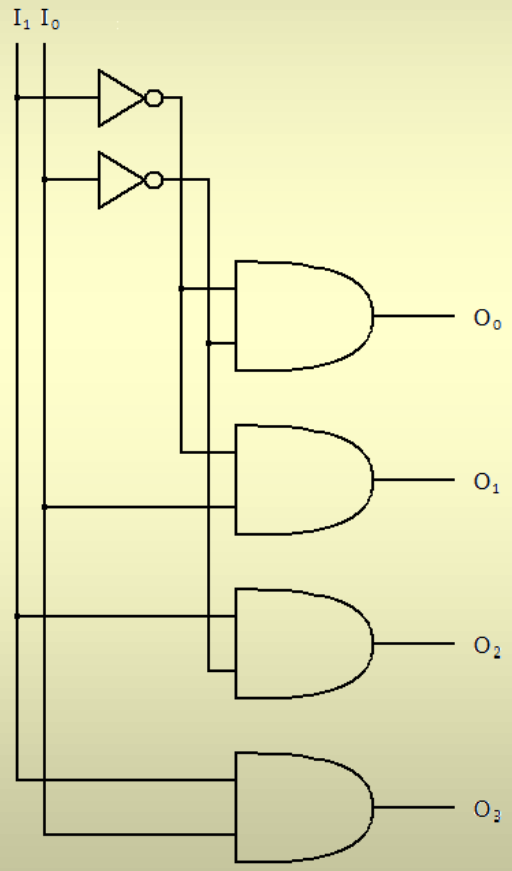
- n intrări
- 2^n ieşiri
- în fiecare moment, exact una din ieșiri este activată
  - cea al cărei indice este egal cu numărul format de intrări
  - fiecare ieşire corespunde unui termen FND scris cu variabilele de intrare

Decodor *n*=2



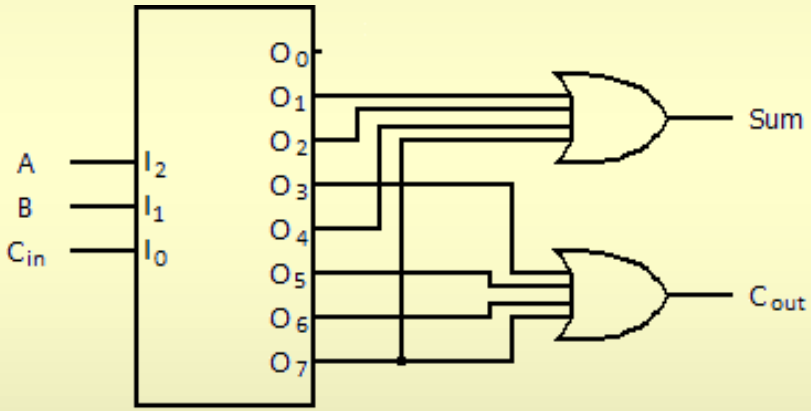
I_1	I_0	O_3	O_2	O_1	O_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Diagrama logică (*n*=2)



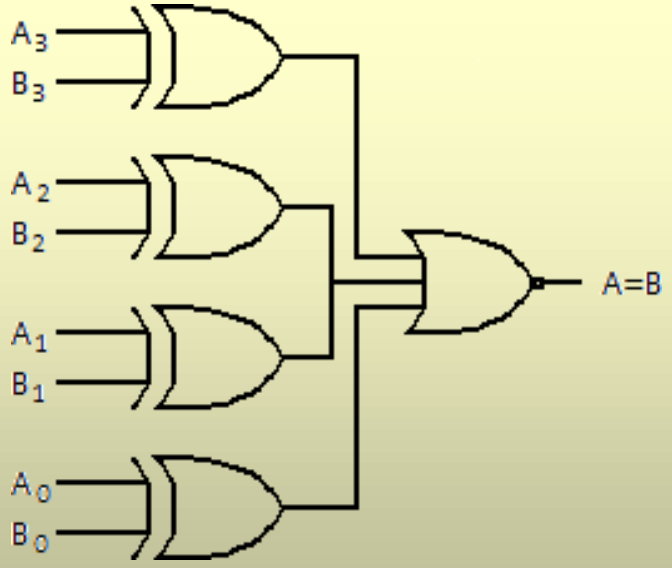
Adunare - implementare cu decodor

A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Comparatorul

- operatorii de comparare: = , > , < , ≥, ≤
  - exemplu de implementare: egalitate pe 4 biţi
  - temă: comparator complet (<, =, >)

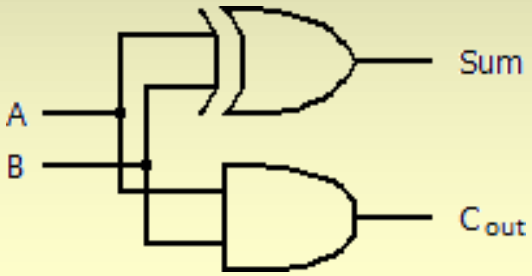


Sumatorul

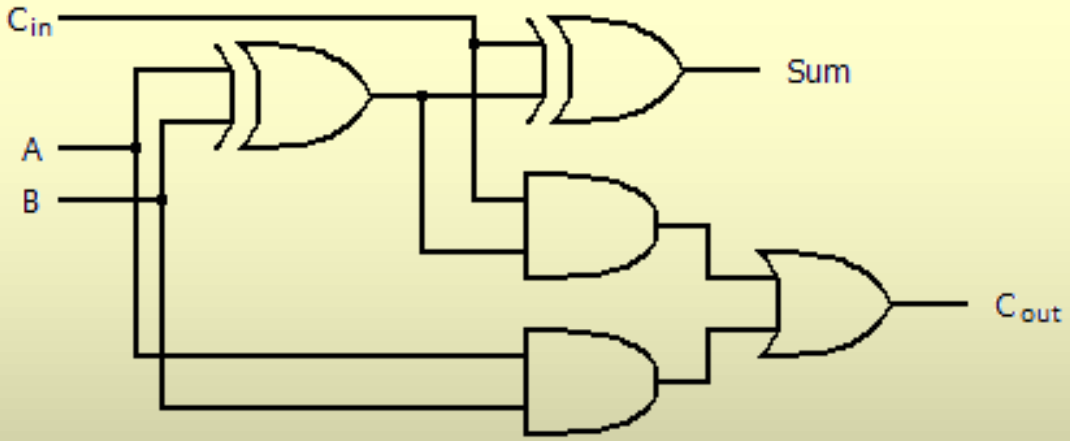
- semi-sumatorul (*half-adder*)
  - adună cei doi biţi de intrare
  - ieşire: un bit sumă şi un bit transport
  - nu poate fi extins pentru mai multe cifre
- sumatorul complet (*full adder*)
  - adună cei trei biţi de intrare (inclusiv transport)
  - aceeași ieşire: un bit sumă şi un bit transport
  - poate fi extins pentru mai multe cifre

Diagrame logice

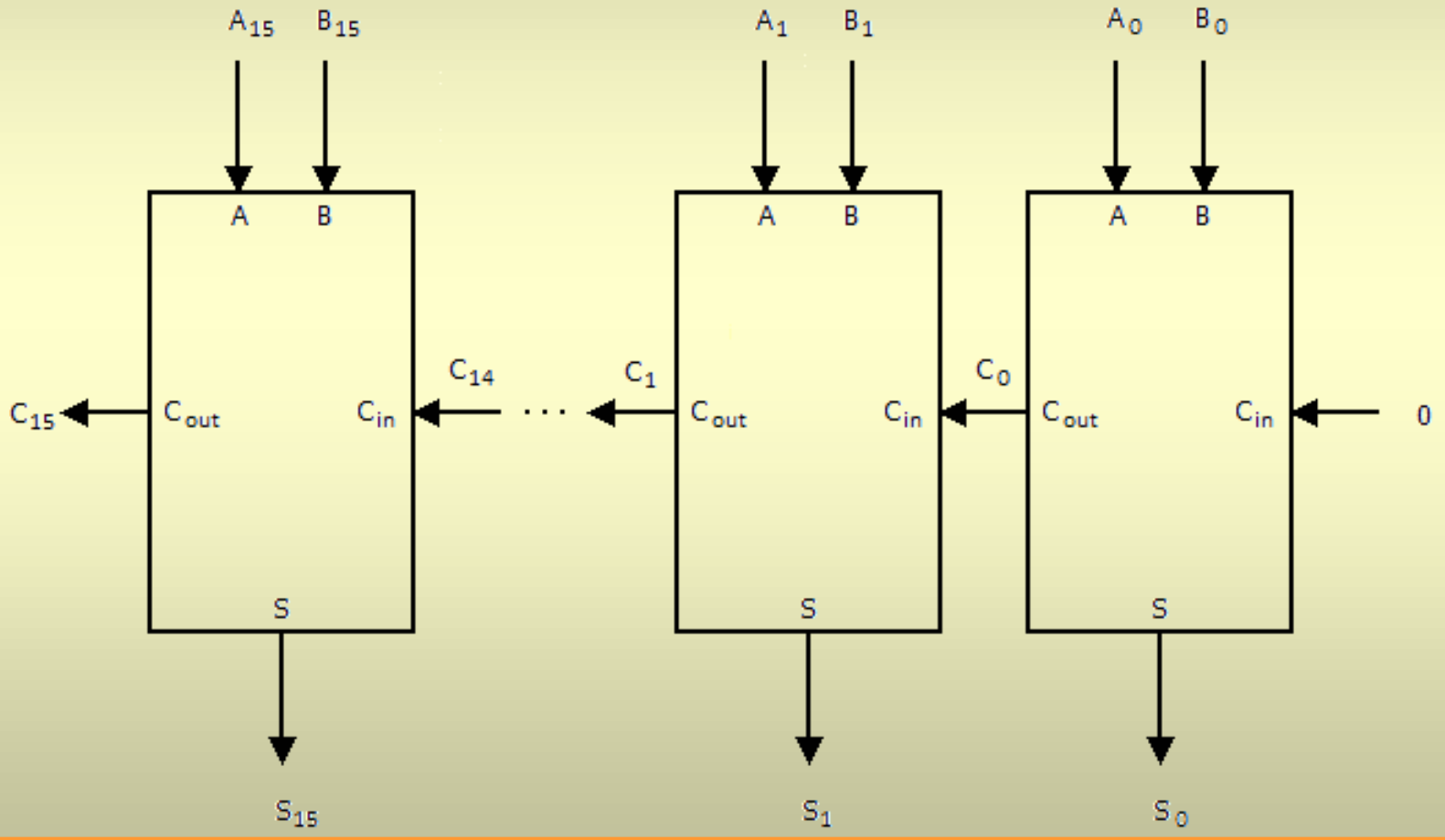
A	B	Sum	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Sumator serial (16 biți)



Sumatoare seriale

- această variantă este sumatorul cu propagarea transportului
- avantaj: același circuit (simplu), repetat
- dezavantaj: viteza
  - la fiecare rang, trebuie așteptat rezultatul de pe rangul anterior
  - deci întârzierea este proporţională cu numărul de biţi

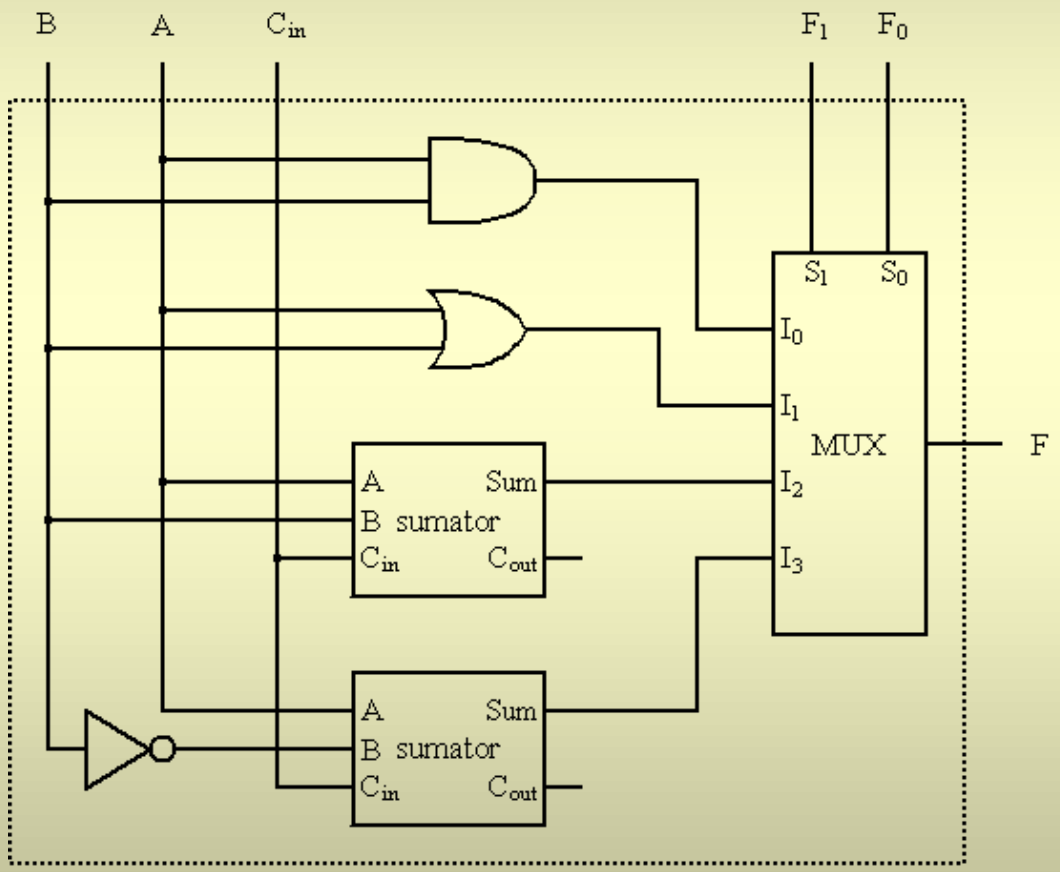
Accelerarea adunării (1)

- sumator cu anticiparea transportului
  - *carry lookahead adder*
  - transportul de intrare - generat independent pentru fiecare rang
    - $C_0 = A_0 B_0$
    - $C_1 = A_0 B_0 A_1 + A_0 B_0 B_1 + A_1 B_1$
    - ...
    - $C_i = G_i + P_i C_{i-1} = A_i B_i + (A_i + B_i) \cdot C_{i-1}$
    - ...

Accelerarea adunării (2)

- sumatorul cu anticiparea transportului
  - avantaj - viteza
    - elimină întârzierea datorată propagării transportului
  - dezavantaj - circuite suplimentare, complexe
  - de obicei - combinație anticipare-propagare
- sumator cu selecţia transportului
  - la fiecare rang se calculează suma pentru C_{in}=0 și C_{in}=1, apoi se selectează cea corectă

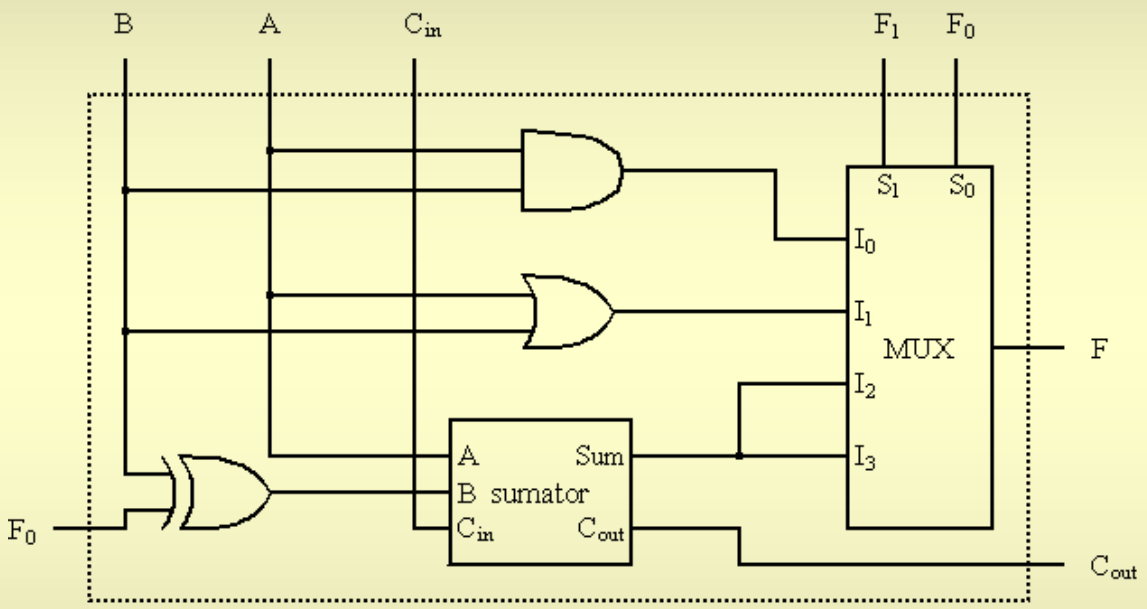
Unitate aritmetică și logică elementară (1 bit)



F_1	F_0	F
0	0	A and B
0	1	A or B
1	0	A+B
1	1	A-B

F_1,F_0 - semnale de control

Variantă îmbunătăţită



F_1	F_0	F
0	0	A and B
0	1	A or B
1	0	A+B
1	1	A-B

F_1,F_0 - semnale de control

Unitate aritmetică și logică pe 16 biți