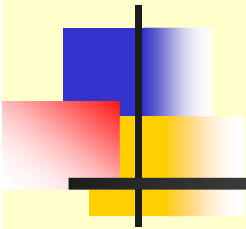
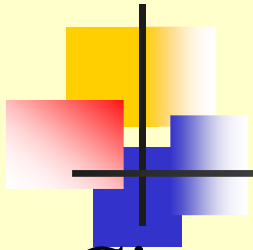


IV. CIRCUITE LOGICE COMBINATIONALE



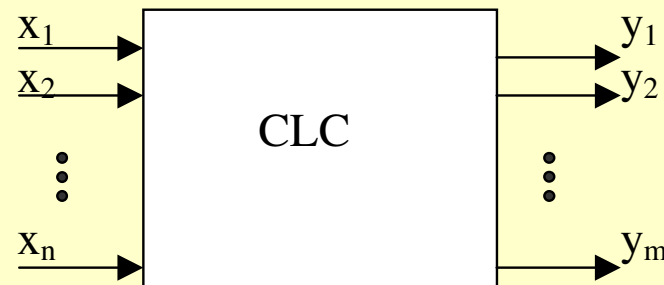


4.1. Definiții

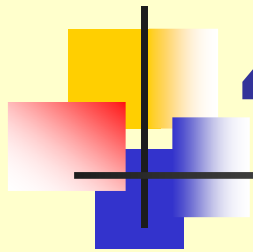
- Circuitele logice combinaționale, **CLC**, sunt un caz particular al automatelor finite (mașinilor de stare - FSM) - **automate de grad 0**
- CLC se caracterizează prin faptul că **variabilele de ieșire sunt independente de timp și de starea internă și sunt determinate doar de variabilele de intrare**
- Practic, datorită întârzierilor produse de circuitele logice și de conexiuni, modificarea variabilelor de ieșire simultan cu cele de intrare nu este realizabilă

4.1. Definiții

■ Schema bloc:



- Funcția de transfer - exprimă legătura între starea ieșirii și starea intrării
- Orice funcție de ieșire y (y_1, y_2, \dots, y_m) este funcție de toate variabilele de intrare (x_1, x_2, \dots, x_n)
 - $y_1 = f_1(x_1, x_2, \dots, x_n)$
 - $y_2 = f_2(x_1, x_2, \dots, x_n)$
 - $y_m = f_m(x_1, x_2, \dots, x_n)$

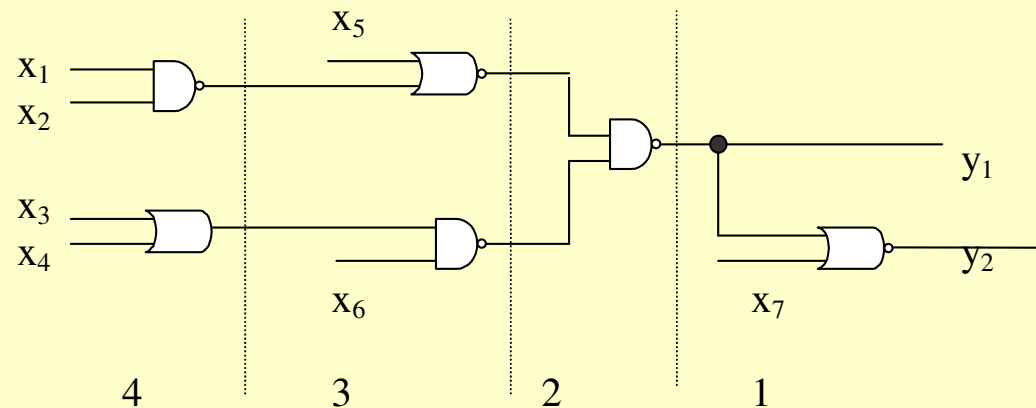


4.2. Analiza CLC

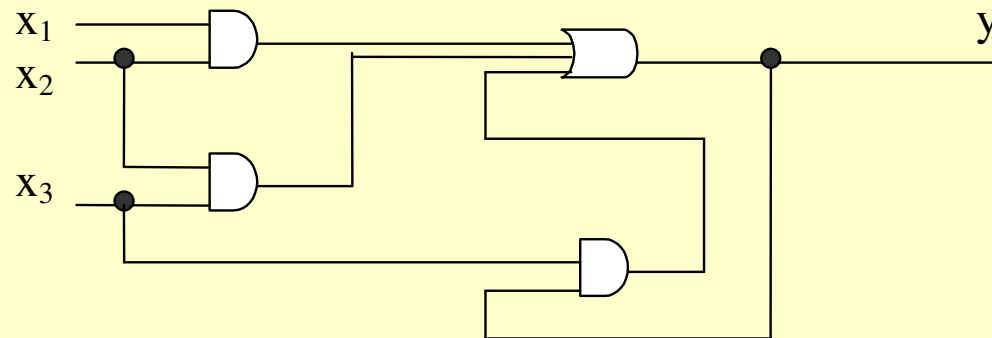
- Se cunoaște schema logică a circuitului și se urmărește stabilirea funcționării acestuia
- Expresiile ieșirilor se determină pornind de la intrări și urmărind transformările acestora
- Definiție - **număr de nivele** al unui CLC = numărul maxim de porți dintre intrări și ieșiri
- Numerotarea nivelelor - de la ieșire spre intrare

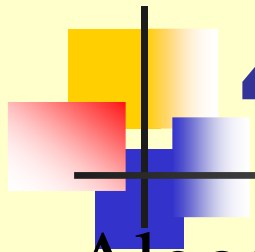
4.2. Analiza CLC

■ Exemplu 1 - circuit cu 4 nivele



■ Exemplu 2 - circuit cu legături inverse





4.2. Analiza CLC

- Algoritm de determinare a legăturilor inverse în CLC
 - se numerotează toate porțile logice care au ca intrări un subset din mulțimea variabilelor de intrare ale circuitului logic (de la 1 la k)
 - se numerotează de la $k+1$ porțile care au ca intrări fie intrări ale circuitului, fie ieșiri ale porților numerotate la punctul anterior
 - CLC - dacă am numerotat toate porțile - exemplu 1
 - CLS (circuit logic secvențial) dacă nu am numerotat toate porțile - exemplu 2



4.3. Sinteza CLC

- Se cunoaște funcția pe care trebuie să o îndeplinească circuitului și se urmărește determinarea structurii (schemei) acestuia
- Etapele sintezei CLC:
 - 1. Enunțul problemei
 - 2. Alcătuirea tabelului de adevăr, definirea funcției sau funcțiilor
 - 3. Minimizarea funcției sau funcțiilor
 - 4. Desenarea schemei circuitului
- Metode de implementare - diferențiate după nivelul de complexitate al integratelor utilizate



4.3.1. Sinteza CLC cu SSI

- Circuitele integrate de tip SSI (small scale integration) au până la 50 de tranzistoare integrate pe capsulă
- Dintre ele - porțile logice fundamentale: ȘI (AND), SAU (OR), NU (NOT), ȘI-NU (NAND), SAU-NU (NOR), SAU-EXCVLUSIV (XOR)
- În general se utilizează pentru adaptarea la aplicație a circuitelor MSI și LSI standardizate, care nu satisfac cu exactitate cerințele de proiectare



4.3.2. Sinteza CLC cu MSI

- Circuitele integrate de tip MSI (medium scale integration) au până la 500 de tranzistoare integrate pe capsulă
- Oferă structuri mai complexe, disponibile ca și **structuri standard (specializate)**
- Forma funcțiilor care trebuie implementate trebuie corelată cu funcțiile circuitelor MSI
- **Observație:** În general nu se mai fac minimizări



4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 1. Convertoare de cod
- **Definiție** - sunt CLC care permit trecerea dintr-un cod binar în alt cod binar
- La intrare se aplică cuvintele unui cod, la ieșire se obțin cuvintele celui alt cod
- Fac compatibilă funcționarea a 2 sisteme în care informația este codificată în mod diferit



4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 1. Convertoare de cod
- **Exemplu:** Conversiile din Gray în binar zecimal și invers
- 1) Cod Gray \rightarrow Cod BCD (8421)
- Cuvintele de cod se notează:
 - Gray: g_n, g_{n-1}, \dots, g_0
 - BCD: b_n, b_{n-1}, \dots, b_0
- Reguli: $b_n = g_n$



4.3.2. Sinteza CLC cu MSI

■ Exemplu: 1) Cod Gray \rightarrow Cod BCD (8421)

g_3	g_2	g_1	g_0	b_3	b_2	b_1	b_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

4.3.2. Sinteza CLC cu MSI

- **Exemplu:** 1) Cod Gray \rightarrow Cod BCD (8421)
- Am completat combinația din tabel pentru valorile intrărilor pentru care funcțiile nu sunt definite
- Construim DK pentru a minimiza funcțiile b_3, b_2, b_1 și b_0

$g_3g_2 \backslash g_1g_0$		00	01	11	10
00					
01					
11	1	1	1	1	
10	1	1	1	1	

- Obținem: $b_3 = g_3$

4.3.2. Sinteza CLC cu MSI

■ Exemplu: 1) Cod Gray \rightarrow Cod BCD (8421)

$g_3g_2 \backslash g_1g_0$	00	01	11	10
00				
01	1	1	1	1
11				
10	1	1	1	1

$g_3g_2 \backslash g_1g_0$	00	01	11	10
00			1	1
01	1	1		
11			1	1
10	1	1		

$g_3g_2 \backslash g_1g_0$	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

■ Obținem: $b_2 = g_2 \oplus g_3$ $b_1 = g_1 \oplus g_2 \oplus g_3$

$$b_0 = g_1 \oplus g_2 \oplus g_3 \oplus g_4$$



4.3.2. Sinteza CLC cu MSI

- **Exemplu:** 1) Cod Gray \rightarrow Cod BCD (8421)

- Deci, în general la trecerea din Gray în binar:

- $b_n = g_n$

- $b_i = \begin{cases} 1 & \text{dacă } \bigoplus_{j=n-1}^i g_j = 1 \\ 0 & \text{dacă nu} \end{cases}$

- 2) Cod BCD \rightarrow Cod Gray

- Obținem asemănător: $g_n = b_n$ și $g_i = b_i \oplus b_{i+1}$



4.3.2. Sinteza CLC cu MSI

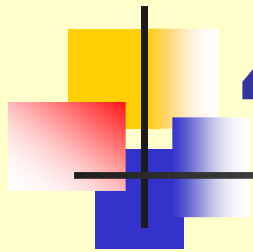
- **CLC MSI specializate (uzuale)**
- **2. Codificatoare**
- **Definiție** - sunt CLC la care activarea unei intrări conduce la apariția unui cuvânt de cod la ieșiri
- Intrările codificatoarelor sunt active pe 0 logic!
- **Codificatoare prioritare**
 - mai multe intrări sunt active simultan
 - la ieșire se obține cuvântul de cod pentru intrarea cu prioritate maximă
 - prioritatea crește de la 0 înspre ultima cifră



4.3.2. Sinteza CLC cu MSI

■ Exemplu: Codificator din zecimal în BCD

Zecimal										BCD			
0	1	2	3	4	5	6	7	8	9	2^3	2^2	2^1	2^0
0	1	1	1	1	1	1	1	1	1	0	0	0	0
1	0	1	1	1	1	1	1	1	1	0	0	0	1
1	1	0	1	1	1	1	1	1	1	0	0	1	0
1	1	1	0	1	1	1	1	1	1	0	0	1	1
1	1	1	1	0	1	1	1	1	1	0	1	0	0
1	1	1	1	1	0	1	1	1	1	0	1	0	1
1	1	1	1	1	1	0	1	1	1	0	1	1	0
1	1	1	1	1	1	1	0	1	1	0	1	1	1
1	1	1	1	1	1	1	1	0	1	1	0	0	0
1	1	1	1	1	1	1	1	1	0	1	0	0	1

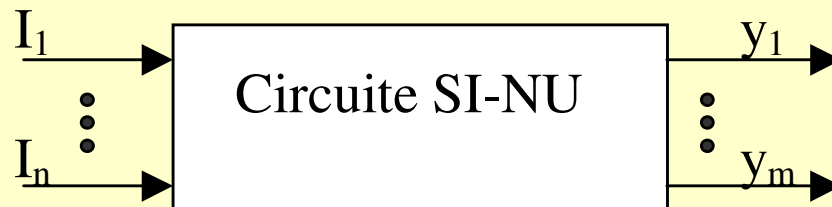


4.3.2. Sinteza CLC cu MSI

- **Exemplu:** Codificator din zecimal în BCD
- Funcțiile pentru cele 4 ieșiri sunt:
 - $2^3 = \overline{8} + \overline{9} = \overline{8 \cdot 9}$
 - $2^2 = \overline{4 \cdot 5 \cdot 6 \cdot 7}$
 - $2^1 = \overline{2 \cdot 3 \cdot 6 \cdot 7}$
 - $2^0 = \overline{1 \cdot 3 \cdot 5 \cdot 7 \cdot 9}$

4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 3. Decodificatoare
- **Definiție** - sunt CLC la care se activează doar una din ieșiri, pentru codul corespunzător de pe intrări
- Ieșirile decodificatoarelor sunt active pe 0 logic



- Numărul ieșirilor distincte ale decodificatoarelor este: $m \leq 2^n$

4.3.2. Sinteza CLC cu MSI

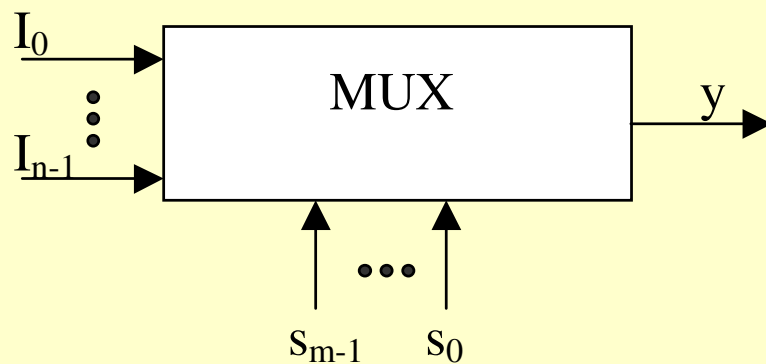
- **Exemplu:** Decodificator pentru 3 cifre binare

I ₂	I ₁	I ₀	O ₇	O ₆	O ₅	O ₄	O ₃	O ₂	O ₁	O ₀
0	0	0	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	0	1
0	1	0	1	1	1	1	1	0	1	1
0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	1	1	0	1	1	1	1
1	0	1	1	1	0	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1

- Funcțiile ieșirilor sunt: $O_7 = I_2 \cdot I_1 \cdot I_0$; $O_6 = I_2 \cdot I_1 \cdot \overline{I_0}$;
 $O_5 = I_2 \cdot \overline{I_1} \cdot I_0$; $O_4 = I_2 \cdot \overline{I_1} \cdot \overline{I_0}$; $O_3 = \overline{I_2} \cdot I_1 \cdot I_0$; $O_2 =$
 $\overline{I_2} \cdot I_1 \cdot \overline{I_0}$; $O_1 = \overline{I_2} \cdot \overline{I_1} \cdot I_0$; $O_0 = \overline{I_2} \cdot \overline{I_1} \cdot \overline{I_0}$

4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 4. Multiplexoare - MUX
- **Definiție** - sunt CLC care permit trecerea datelor de pe una din intrări la o ieșire unică. Selecția (comanda) intrării se face printr-un cuvânt de cod de selecție numit și adresă



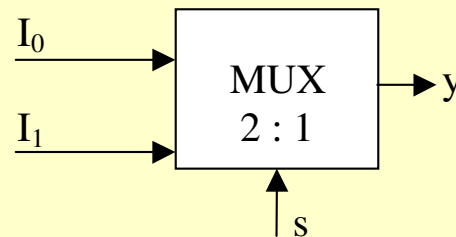


4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 4. Multiplexoare
- Cu “m” linii de selecție se pot selecta 2^m intrări
- Funcția de ieșire este:
$$y = I_i \quad \text{unde } i \text{ este numărul de combinații}$$
$$i = s_{m-1} \cdot 2^{m-1} + \dots + s_1 \cdot 2^1 + s_0 \cdot 2^0$$
- Aplicații: selecția secvențială a datelor, conversia paralel-serie a datelor, sisteme de transmisie a datelor pe un singur canal, implementarea CLC cu o ieșire

4.3.2. Sinteza CLC cu MSI

■ Exemplu: Multiplexor 2:1



- Are 2 intrări de date, I_0 și I_1 , o intrare de selecție s și o ieșire y
- Avem ecuația ieșirii: $y = I_0 \cdot \overline{s} + I_1 \cdot s$
 - Dacă $s = 0 \Rightarrow y = I_0$
 - Dacă $s = 1 \Rightarrow y = I_1$
- Multiplexorul trimite la ieșire data de intrare care corespunde selecției din acel moment

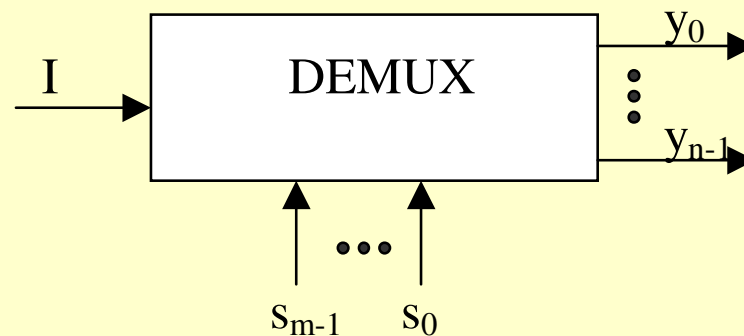


4.3.2. Sinteza CLC cu MSI

- Există multiplexoare de tip: 2:1, 4:1, 8:1, 16:1, 32:1, 64: 1 etc.
- Multiplexoarele integrate:
 - au disponibile atât ieșirea adevărată cât și ieșirea negată
 - au o intrare de **validare** (Enable) care permite o funcție ȘI suplimentară - dacă această intrare nu este activă, se oprește funcționarea circuitului
- Multiplexoarele cu număr mare de intrări de date se pot obține prin **cascadarea** multiplexoarelor cu număr mai mic de intrări de date

4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 5. Demultiplexoare - DEMUX
- **Definiție** - sunt CLC care permit transmiterea datelor de pe o intrare de date pe una din ieșirile selectate. Selecția (comanda) ieșirii se face printr-un cuvânt de cod de selecție numit și adresă





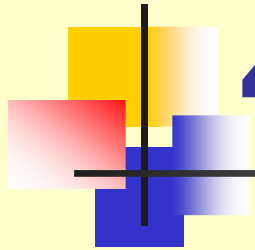
4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 5. Demultiplexoare
- Cu “m” linii de selecție se pot selecta 2^m ieșiri
- Funcțiile de ieșire sunt:

$$y_0 = \overline{s_{m-1}} \cdot \dots \cdot \overline{s_1} \cdot \overline{s_0} \cdot I$$

$$y_1 = \overline{s_{m-1}} \cdot \dots \cdot \overline{s_1} \cdot s_0 \cdot I$$

$$y_2^{m-1} = s_{m-1} \cdot \dots \cdot s_1 \cdot s_0 \cdot I$$



4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 5. Demultiplexoare
- Există demultiplexoare de tip: 1:2, 1:4, 1:8, 1:16 etc.
- Demultiplexoarele cu număr mai mare de ieșiri se obțin prin **cascadare**
- Se pot folosi la conversia din serie în paralel a informației

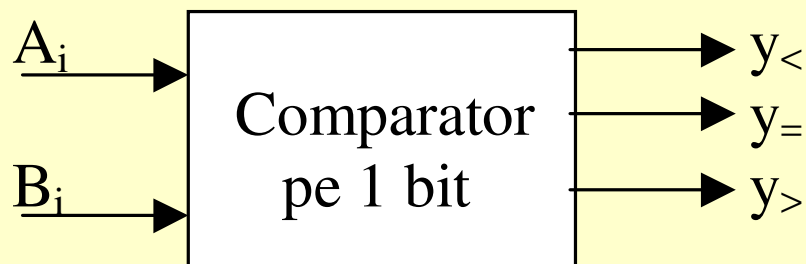


4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- **6. Comparatoare numerice**
- **Definiție** - sunt CLC care permit determinarea valorii relative a două numere binare
- Comparatoare:
 - pe 1 bit
 - pe mai mulți biți
- În general comparatorul pe 1 bit reprezintă celula de bază pentru compararea numerelor pe mai mulți biți

4.3.2. Sinteza CLC cu MSI

■ Exemplu: Comparator pe 1 bit



■ Avem funcțiile de ieșire:

$$y_{<} = \overline{A_i} \bullet B_i \quad \text{pentru } A_i < B_i$$

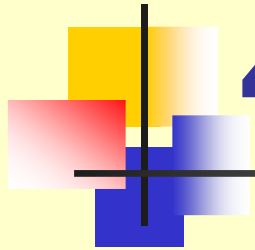
$$y_{=} = A_i \oplus B_i = A_i \odot B_i \quad \text{pentru } A_i = B_i$$

$$y_{>} = A_i \bullet \overline{B_i} \quad \text{pentru } A_i > B_i$$



4.3.2. Sinteza CLC cu MSI

- **CLC MSI specializate (uzuale)**
- **7. Generatoare / detectoare de paritate**
- **Definiție** - sunt CLC cu rol de a determina și genera paritatea sau imparitatea numărului de variabile de intrare egale cu 1
- Bitul de paritate se utilizează ca metodă de verificare a transferului de date
 - număr biți de 1 + bit de paritate = număr par (paritate pară)
 - număr biți de 1 + bit de paritate = număr impar (paritate impară)

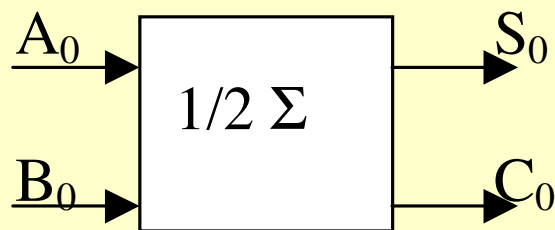


4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 7. Generatoare / detectoare de paritate
- Realizarea detectoarelor de paritate se bazează pe funcția SAU-EXCLUSIV:
 - valoare 0 - pentru număr par
 - valoare 1 - pentru număr impar

4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 8. Sumatoare / scăzătoare
- Definiție - sunt CLC care realizează adunarea, respectiv scăderea cifrelor binare
- Semisumator - CLC care face suma a 2 numere binare de 1 bit, fără să țină cont de transportul de la bitul de semnificație imediat inferioară





4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 8. Sumatoare / scăzătoare
- Tabelul de adevăr pentru semisumator:

A_0	B_0	C_0	S_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

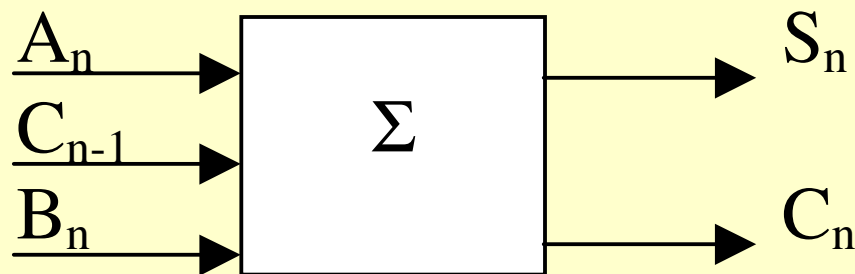
- Ecuațiile pentru suma S_0 și transportul spre rangul superior C_0

$$S_0 = \overline{A_0} \cdot B_0 + A_0 \cdot \overline{B_0} = A_0 \oplus B_0$$

$$C_0 = A_0 \cdot B_0$$

4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 8. Sumatoare / scăzătoare
- Sumatorul pentru bitul de rang “n” ține cont și de transportul de la rangul imediat inferior - este un sumator complet pe 1 bit





4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 8. Sumatoare / scăzătoare
- Tabelul de adevăr pentru sumatorul complet pe 1 bit este:

C_{n-1}	A_n	B_n	C_n	S_n
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



4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 8. Sumatoare / scăzătoare
- Ecuațiile pentru sumă și transport pentru rangul superior
$$S_n = \overline{A_n} \cdot \overline{B_n} \cdot \overline{C_{n-1}} + \overline{A_n} \cdot B_n \cdot \overline{C_{n-1}} + \overline{A_n} \cdot \overline{B_n} \cdot C_{n-1} + A_n \cdot B_n \cdot C_{n-1}$$
$$A_n \cdot B_n \cdot C_{n-1} = A_n \oplus B_n \oplus C_{n-1}$$
$$C_n = A_n \cdot C_{n-1} + B_n \cdot C_{n-1} + A_n \cdot B_n$$
- Sumatoarele pentru cuvinte binare cu mai mulți biți se pot realiza prin interconectarea celor pe 1 bit - adunarea se efectuează în paralel, transportul în serie



4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 8. Sumatoare / scăzătoare
- Semiscăzător - CLC care realizează scăderea a 2 numere binare de 1 bit, fără să țină cont de împrumutul de la rangul inferior
- Ecuațiile pentru semiscăzător:

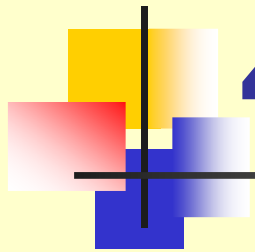
$$D_0 = \overline{A_0} \bullet B_0 + A_0 \bullet \overline{B_0} = A_0 \oplus B_0$$

$$I_0 = A_0 \bullet B_0$$



4.3.2. Sinteza CLC cu MSI

- CLC MSI specializate (uzuale)
- 8. Sumatoare / scăzătoare
- Scăzătorul complet pe 1 bit ține cont și de împrumutul de la rangul inferior
- Ecuațiile pentru diferență și împrumut către rangul superior
$$D_n = \overline{A_n} \cdot \overline{B_n} \cdot I_{n-1} + \overline{A_n} \cdot B_n \cdot \overline{I_{n-1}} + A_n \cdot \overline{B_n} \cdot \overline{I_{n-1}} + A_n \cdot B_n \cdot I_{n-1} = A_n \oplus B_n \oplus I_{n-1}$$
$$I_n = \overline{A_n} \cdot I_{n-1} + B_n \cdot I_{n-1} + \overline{A_n} \cdot B_n$$
- Scăzătoarele pentru cuvinte binare cu mai mulți biți se pot realiza prin interconectarea celor pe 1 bit



4.3.2. Sinteza CLC cu MSI

- **CLC MSI specializate (uzuale)**
- **9. Unități aritmetico-logice (ALU)**
- **Definiție** - sunt CLC care realizează operații de tip aritmetic și de tip logic
- Operații aritmetice: adunare, scădere, incrementare, decrementare, comparare de egalitate
- Operații logice: ȘI, SAU, NU, ȘI-NU, SAU-NU, SAU-EXCLUSIV, COINCIDENȚĂ

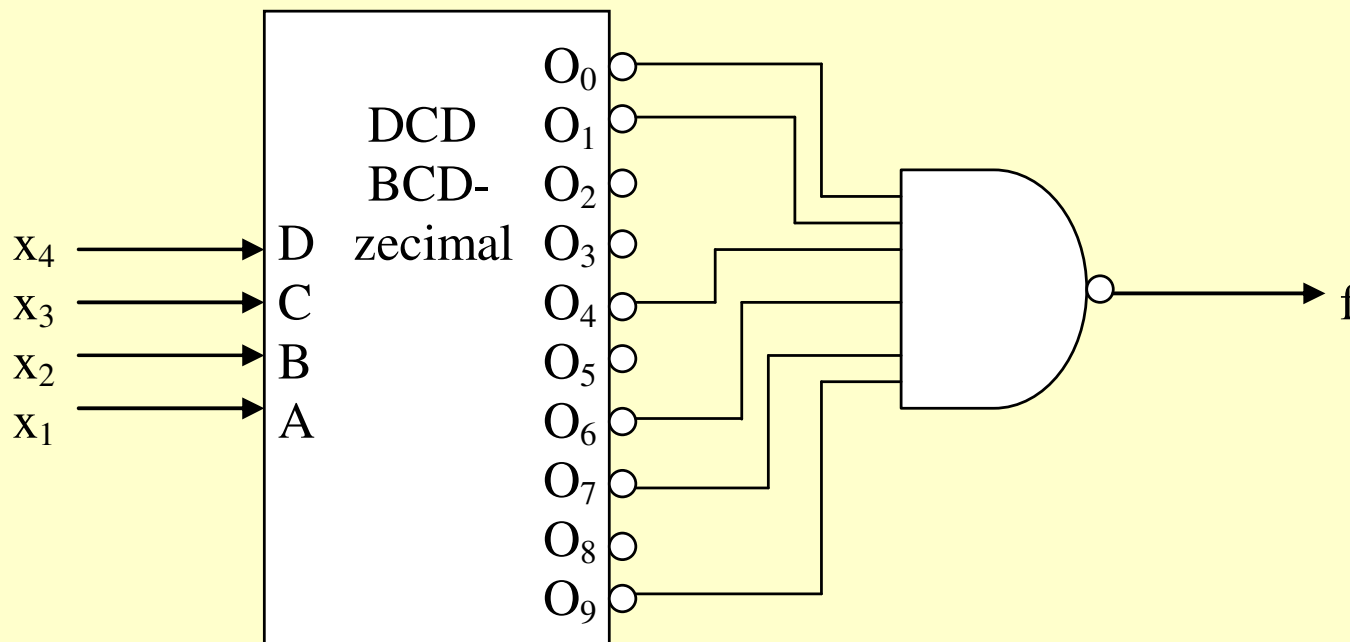
4.3.2.1. Implementarea funcțiilor booleene cu MSI

- Decodificatorul, Demultiplexorul și Multiplexorul generează în interiorul lor toți termenii canonici → sunt **circuite universale**
- **Implementarea cu Decodificator** a unei funcții booleene:
 - nu necesită minimizare
 - la ieșirea decodicatorului avem toți termenii canonici **negați** ai formeii canonice disjunctive (FCD) a funcției
 - realizarea funcției se face prin adăugarea unei porți logice de tip ȘI-NU care are numărul de intrări egal cu numărul de termeni ai funcției

4.3.2.1. Implementarea funcțiilor booleene cu MSI

- **Exemplu:** Să se implementeze cu un decodificator funcția booleană de 4 variabile:

$$f(x_4, x_3, x_2, x_1) = \Sigma(0, 1, 4, 6, 7, 9)$$



4.3.2.1. Implementarea funcțiilor booleene cu MSI

- **Implementarea cu Multiplexor** a unei funcții booleene se bazează pe ecuația care definește funcționarea multiplexorului
- La o funcție booleană cu “ n ” variabile de intrare, dacă dăm factor comun 2 variabile de intrare, se obțin 4 funcții de “ $n-2$ ” variabile de intrare
- La o funcție booleană cu “ n ” variabile de intrare, dacă dăm factor comun 3 variabile de intrare, se obțin 8 funcții de “ $n-3$ ” variabile de intrare
- Ș.a.m.d.

4.3.2.1. Implementarea funcțiilor booleene cu MSI

- **Implementarea cu Multiplexor**
 - Dacă variabilele scoase în factor comun se pun pe selecțiile multiplexorului, funcțiile rămase se pot conecta la intrările de date ale multiplexorului
 - **Exemplu:** pentru o funcție de 4 variabile, dacă folosim un MUX 8:1, punem 3 variabile pe selecțiile MUX și rămâne ca pe intrările de date să punem cele 8 funcții de o variabilă rămase
- Singurele funcții posibile de o variabilă sunt: x , \overline{x} , 0 și 1

4.3.2.1. Implementarea funcțiilor booleene cu MSI

■ Implementarea cu Multiplexor

■ **Exemplu:** Implementarea cu MUX 8:1 a funcției

$$\begin{aligned} f(x_3, x_2, x_1, x_0) &= \Sigma (0, 1, 3, 5, 9, 10, 13, 15) = \\ &= \overline{x_3}\overline{x_2}\overline{x_1}\overline{x_0} + \overline{x_3}\overline{x_2}\overline{x_1}x_0 + \overline{x_3}\overline{x_2}x_1\overline{x_0} + \overline{x_3}\overline{x_2}x_1x_0 + \\ &+ x_3\overline{x_2}\overline{x_1}\overline{x_0} + x_3\overline{x_2}\overline{x_1}x_0 + x_3\overline{x_2}x_1\overline{x_0} + x_3\overline{x_2}x_1x_0 \end{aligned}$$

- Pe intrările de selecție se aplică variabilele x_2, x_1, x_0
- Pentru determinarea celor 8 funcții care urmează să fie puse pe intrările de date ale MUX, $I_0 - I_7$, se construiește un tabel

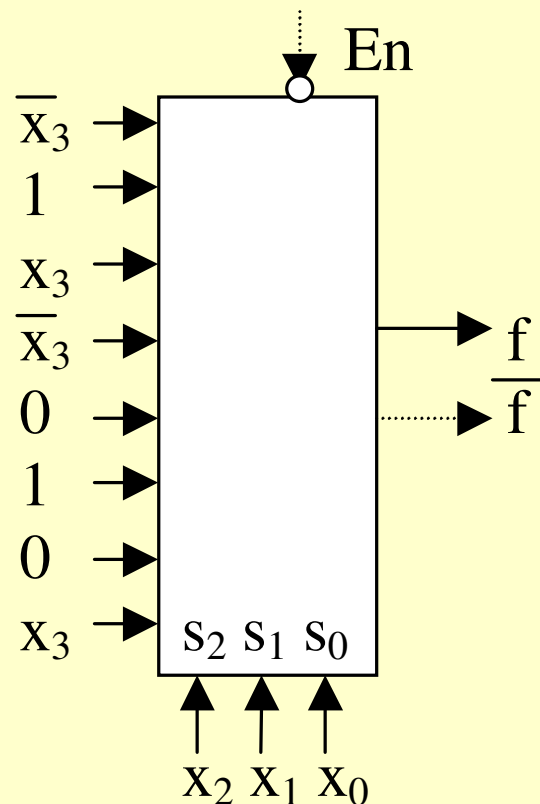
4.3.2.1. Implementarea funcțiilor booleene cu MSI

■ Implementarea cu Multiplexor - exemplu

Intrare MUX	Valoare funcție de o variabilă	Termeni funcție - Variabile pe selecție
I_0	$\overline{x_3}$	$\overline{x_3} \quad \overline{x_2} \overline{x_1} \overline{x_0}$
I_1	1	$(\overline{x_3} + x_3) \quad \overline{x_2} \overline{x_1} x_0$
I_2	x_3	$x_3 \quad \overline{x_2} x_1 \overline{x_0}$
I_3	$\overline{x_3}$	$\overline{x_3} \quad \overline{x_2} x_1 x_0$
I_4	0	$0 \quad x_2 \overline{x_1} \overline{x_0}$
I_5	1	$(\overline{x_3} + x_3) \quad x_2 \overline{x_1} x_0$
I_6	0	$0 \quad x_2 x_1 \overline{x_0}$
I_7	x_3	$x_3 \quad x_2 x_1 x_0$

4.3.2.1. Implementarea funcțiilor booleene cu MSI

■ Implementarea cu Multiplexor - exemplu



■ **Observație:** în funcție de variabilele alese pe selecții, implementarea funcției diferă

4.3.2.1. Implementarea funcțiilor booleene cu MSI

- Implementarea cu Multiplexor - exemplu
- La funcția reprezentată prin DK implementarea cu MUX se bazează pe o altă DK, care definește domeniul intrărilor de date I
- Domeniul intrărilor pentru funcția de 4 variabile, dacă selecțiile sunt x_2, x_1, x_0

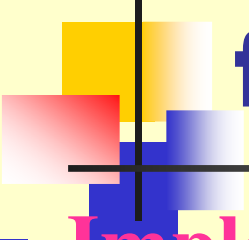
$x_3x_2 \backslash x_1x_0$		00	01	11	10
x_3x_2	00	I_0	I_1	I_3	I_2
	01	I_4	I_5	I_7	I_6
	11	I_4	I_5	I_7	I_6
	10	I_0	I_1	I_3	I_2

4.3.2.1. Implementarea funcțiilor booleene cu MSI

- Implementarea cu Multiplexor - exemplu
- DK pentru funcția f este:

$x_3x_2 \backslash x_1x_0$		00	01	11	10
x_3x_2	00	1	1	1	
	01		1		
	11		1	1	
	10		1		1

- Prin comparare cu DK pentru intrări obținem pentru $I_0 - I_7$ valorile: $I_0 = \bar{x}_3$; $I_1 = 1$; $I_2 = x_3$; $I_3 = \bar{x}_3$; $I_4 = 0$; $I_5 = 1$; $I_6 = 0$; $I_7 = x_3$, identice cu cele obținute anterior



4.3.2.1. Implementarea funcțiilor booleene cu MSI

- **Implementarea cu Multiplexor**
- Avantajele implementării cu MUX
 - se poate utiliza un singur circuit
 - doar o singură variabilă trebuie să fie disponibilă și adevărată și negată
 - MUX are intrarea de Enable care poate fi folosită ca un ȘI final cu întreaga funcție
- Dezavantajele implementării cu MUX
 - nu se pot implementa sisteme de funcții
 - nu se pot implementa funcții cu numărul de termeni mai mare decât numărul intrărilor