

Proiectare logică

Curs 5

Analiza circuitelor logice combinaționale.
Circuite SSI și MSI. Proiectare cu SSI și MSI

Cristian Vancea

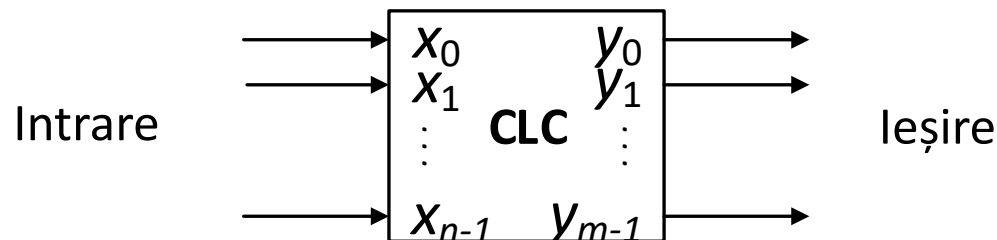
<https://users.utcluj.ro/~vcristian/PL.html>

Cuprins

- Circuite logice combinaționale (CLC)
- Analiza CLC
- Circuite SSI (Small Scale Integration) uzuale
- Proiectare CLC cu circuite SSI
- Circuite MSI (Medium Scale Integration) uzuale
- Proiectare CLC cu circuite MSI

Circuite logice combinaționale

Definiție – circuitele logice combinaționale (CLC) sunt **automate finite de ordin 0**: ieșirile depind numai de variabilele de intrare.



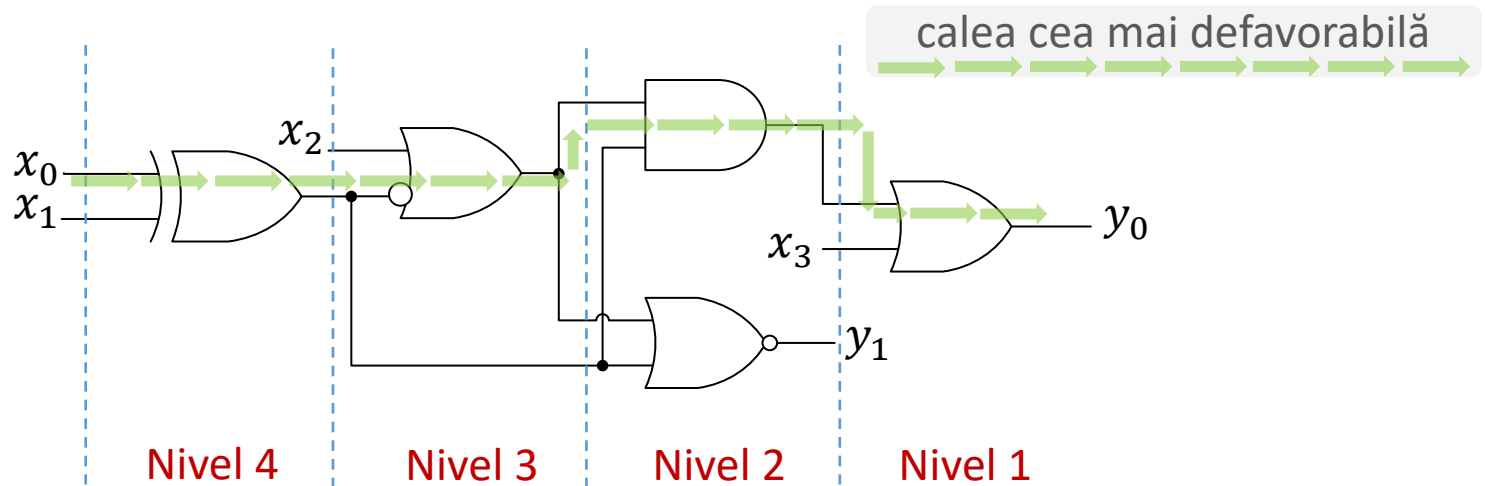
$$\begin{cases} y_0 = f_0(x_0, x_1, \dots, x_{n-1}) \\ y_1 = f_1(x_0, x_1, \dots, x_{n-1}) \\ \dots \\ y_{m-1} = f_{m-1}(x_0, x_1, \dots, x_{n-1}) \end{cases}$$

f_0, f_1, \dots, f_{m-1} – funcții de transfer

Circuite logice combinaționale

Analiza CLC

- Se determină funcționalitatea unui circuit dat urmărind transformările aplicate variabilelor de intrare.
- Definiție** – **numărul de niveluri logice** este numărul maxim de porți logice prin care se propagă semnalul de la o variabilă de intrare până la ieșire; se numerotează de la ieșire spre intrare.



Circuite logice combinaționale

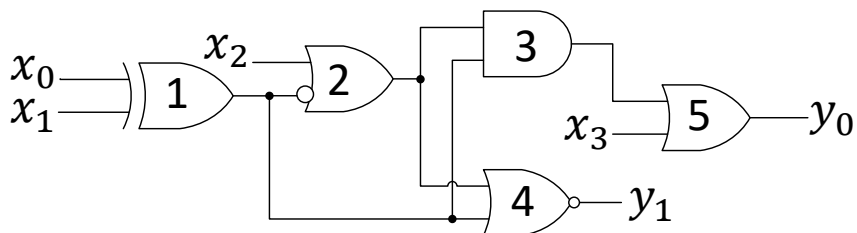
Analiza CLC

Se determină dacă **un circuit este CLC** prin **absența legăturilor inverse**.

Algoritmul de determinare a legăturilor inverse

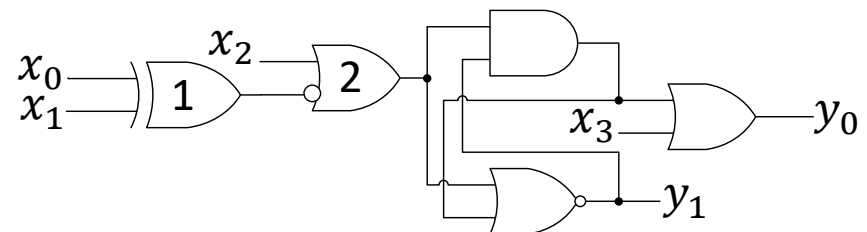
- Se numerează crescător porțile care au ca intrări numai variabile de intrare.
- Se continuă numerotarea porților care au ca intrări variabile de intrare și/sau ieșiri ale porților deja numerotate.
- Dacă la final există cel puțin o poartă nenumerotată înseamnă că există legături inverse și circuitul nu este CLC.

Ex₁:



Este CLC

Ex₂:



Nu este CLC

Circuite logice combinaționale

Sinteza CLC

- Etape:
 1. Enunțul problemei
 2. Definirea funcțiilor
 3. Minimizarea funcțiilor
 4. Desenarea schemei circuitului
- Există mai multe metode de implementare în funcție de complexitatea circuitelor folosite.

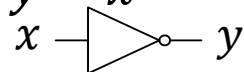
Circuite logice combinaționale

Circuite **SSI** (Small Scale Integration) uzuale

- SSI au până la 50 tranzistoare integrate – Exemple:

Inversor (NOT)

$$y = \bar{x}$$

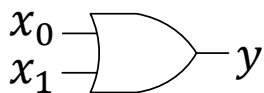


x	y
0	1
1	0

SAU (OR)

$$y = x_1 + x_0$$

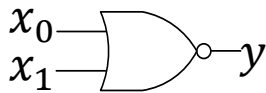
x_1	x_0	y
0	0	0
0	1	1
1	0	1
1	1	1



SAU-NU (NOR)

$$y = \overline{x_1 + x_0}$$

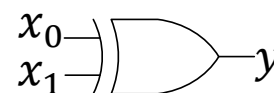
x_1	x_0	y
0	0	1
0	1	0
1	0	0
1	1	0



SAU-EXCLUSIV (XOR)

$$y = x_1 \oplus x_0$$

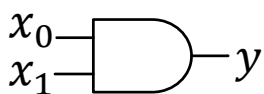
x_1	x_0	y
0	0	0
0	1	1
1	0	1
1	1	0



ȘI (AND)

$$y = x_1 \cdot x_0$$

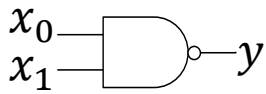
x_1	x_0	y
0	0	0
0	1	0
1	0	0
1	1	1



ȘI-NU (NAND)

$$y = \overline{x_1 \cdot x_0}$$

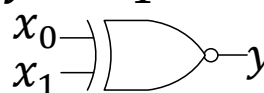
x_1	x_0	y
0	0	1
0	1	1
1	0	1
1	1	0



COINCIDENȚĂ (XNOR)

$$y = x_1 \odot x_0$$

x_1	x_0	y
0	0	1
0	1	0
1	0	0
1	1	1



Circuite logice combinaționale

Sinteza funcțiilor booleene cu circuite **SSI**

- Se creează schema logică folosind porțile fundamentale în conformitate cu:
 - Expresia funcției
 - Forma Canonică Conjunctivă
 - Forma Canonică Disjunctivă
 - Forma Disjunctivă Minimă
 - Forma Conjunctivă Minimă
 - Expresii echivalente obținute aplicând axiomele și proprietățile algebrei booleene
- Variațiuni:
 - Implementare numai cu porți ȘI-NU
 - Implementare numai cu porți SAU-NU

Circuite logice combinaționale

Sinteza funcțiilor booleene cu ȘI-NU

- Se aduce funcția la FDM, se aplică dubla negație și De Morgan

Ex: $n = 4$ $f = \sum(1,3,5,7,8,9,12,13)$

$x_3x_2 \backslash x_1x_0$	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	1	0	0
10	1	1	0	0

Dubla negație:

$$f = \overline{\overline{f}}$$

De Morgan

$$f \xrightarrow{\text{FDM}} \overline{x_3} \cdot x_0 + x_3 \cdot \overline{x_1} = \overline{\overline{\overline{x_3} \cdot x_0 + x_3 \cdot \overline{x_1}}} = \overline{\overline{\overline{x_3} \cdot x_0} \cdot \overline{\overline{x_3 \cdot \overline{x_1}}}} =$$

$$= \overline{\overline{\overline{x_3} \cdot x_0} \cdot \overline{\overline{x_3 \cdot \overline{x_1}}}} = \overline{\overline{\overline{x_3} \cdot x_0} \cdot \overline{\overline{x_3} \cdot \overline{\overline{x_1}}}} =$$

$$\overline{\overline{\overline{x_3} \cdot x_0} \cdot \overline{\overline{x_3} \cdot \overline{\overline{x_1}}}} = \overline{\overline{\overline{x_3} \cdot x_0} \cdot \overline{\overline{x_3} \cdot \overline{\overline{x_1}}}} =$$

$$\overline{\overline{\overline{x_3} \cdot x_0} \cdot \overline{\overline{x_3} \cdot \overline{\overline{x_1}}}} = \overline{\overline{\overline{x_3} \cdot x_0} \cdot \overline{\overline{x_3} \cdot \overline{\overline{x_1}}}} =$$

Circuite logice combinaționale

Sinteza funcțiilor booleene cu SAU-NU

- Se aduce funcția la FCM, se aplică dubla negație și De Morgan

Ex: $n = 4$ $f = \sum(1,3,5,7,8,9,12,13)$

$x_3x_2 \backslash x_1x_0$	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	1	0	0
10	1	1	0	0

Dubla negație:

$$f = \overline{\overline{f}}$$

De Morgan

$$f \xrightarrow{\text{FCM}} (x_3 + x_0) \cdot (\overline{x_3} + \overline{x_1}) = \overline{\overline{(x_3 + x_0) \cdot (\overline{x_3} + \overline{x_1})}} = \overline{\overline{x_3 + x_0 + \overline{x_3} + \overline{x_1}}} =$$

$$\overline{\overline{x}} = \overline{x + x} = \overline{x + 0}$$

$$= \overline{\overline{x_3 + x_0} + \overline{\overline{x_3 + x_3} + \overline{x_1} + 0}}$$

Circuite logice combinaționale

Circuite **MSI** (Medium Scale Integration) uzuale

- MSI au până la 500 tranzistoare integrate.
- Implementează funcții standard mai complexe => expresia logică de implementat trebuie adaptată la aceste funcții.

Circuite logice combinaționale

Circuite MSI uzuale – **Convertoare de cod**

- Realizează conversia de la un cod binar la altul.
- Se aplică un cod pe intrare și se obține un alt cod la ieșire.
- Sunt utile la comunicarea între 2 sisteme care codifică informația diferit.

Circuite logice combinaționale

Circuite MSI uzuale – **Convertoare de cod**

Convertor Gray → BCD (8421)

x_3	x_2	x_1	x_0	y_3	y_2	y_1	y_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

y_3 :

$x_3x_2 \backslash x_1x_0$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$$y_3 = x_3$$

y_1 :

$x_3x_2 \backslash x_1x_0$	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

$$\begin{aligned}
 y_1 &= \bar{x}_3 \cdot \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot x_2 \cdot x_1 + x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 = \\
 &= \bar{x}_3 \cdot (\bar{x}_2 \cdot x_1 + x_2 \cdot \bar{x}_1) + x_3 \cdot (x_2 \cdot x_1 + \bar{x}_2 \cdot \bar{x}_1) = \\
 &= \bar{x}_3 \cdot (x_2 \oplus x_1) + x_3 \cdot (x_2 \oplus x_1) = x_3 \oplus x_2 \oplus x_1
 \end{aligned}$$

$$y_0 = \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 \cdot x_0 + \dots = x_3 \oplus x_2 \oplus x_1 \oplus x_0$$

y_2 :

$x_3x_2 \backslash x_1x_0$	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$$y_2 = \bar{x}_3 \cdot x_2 + x_3 \cdot \bar{x}_2 = x_3 \oplus x_2$$

y_0 :

$x_3x_2 \backslash x_1x_0$	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

Circuite logice combinaționale

Circuite MSI uzuale – **Convertoare de cod**

Convertor Gray \rightarrow BCD (8421)

$$\begin{array}{l} y_3 = x_3 \\ y_2 = x_3 \oplus x_2 \\ y_1 = x_3 \oplus x_2 \oplus x_1 \\ y_0 = x_3 \oplus x_2 \oplus x_1 \oplus x_0 \end{array} \quad \begin{array}{c} \text{Generalizare } n \text{ biți} \\ \longrightarrow \end{array} \quad \begin{array}{l} y_{n-1} = x_{n-1} \\ y_i = x_{n-1} \oplus x_{n-2} \oplus \cdots \oplus x_i, i = \overline{0, n-2} \end{array}$$

Convertor BCD (8421) \rightarrow Gray

Formula generală pentru n biți după minimizare:

$$\begin{array}{l} y_{n-1} = x_{n-1} \\ y_i = x_i \oplus x_{i+1}, i = \overline{0, n-2} \end{array}$$

Ex: $n = 4$

$$\begin{array}{l} y_3 = x_3 \\ y_2 = x_2 \oplus x_3 \\ y_1 = x_1 \oplus x_2 \\ y_0 = x_0 \oplus x_1 \end{array}$$

Circuite logice combinaționale

Circuite MSI uzuale – Codificatoare

- La activarea unei intrări se generează codul binar corespunzător la ieșire.
- O intrare este considerată activă când are valoarea 0 (logică negativă).
- **Codificator prioritar**: intrările au priorități prestabilite; dacă mai multe intrări sunt active ieșirea va avea codul intrării cu prioritatea cea mai mare.

Codificator Zecimal → BCD

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	y_3	y_2	y_1	y_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



De Morgan

$$y_3 = \overline{x_8} + \overline{x_9} = \overline{x_8 \cdot x_9}$$

$$y_2 = \overline{x_4} + \overline{x_5} + \overline{x_6} + \overline{x_7} = \overline{x_4 \cdot x_5 \cdot x_6 \cdot x_7}$$

$$y_1 = \overline{x_2} + \overline{x_3} + \overline{x_6} + \overline{x_7} = \overline{x_2 \cdot x_3 \cdot x_6 \cdot x_7}$$

$$y_0 = \overline{x_1} + \overline{x_3} + \overline{x_5} + \overline{x_7} + \overline{x_9} = \overline{x_1 \cdot x_3 \cdot x_5 \cdot x_7 \cdot x_9}$$



pr. mică → pr. mare

Dacă e codificator prioritar atunci prioritatea crește începând de la intrarea 0 la 9.

Circuite logice combinaționale

Circuite MSI uzuale – **Decodificatoare (DCD)**

- În prezența unui cod binar pe intrare se activează ieșirea corespunzătoare.
- Ieșirile sunt active pe 0 (logică negativă).
- Pentru n biți de intrare numărul de ieșiri este $m \leq 2^n$.



Ex: $n = 3$ $m = 8$

x_2	x_1	x_0	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0



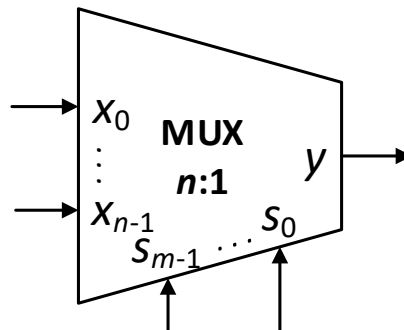
$$\begin{aligned}
 y_0 &= S_0 = \overline{P_0} = \overline{\overline{x_2} \cdot \overline{x_1} \cdot \overline{x_0}} \\
 y_1 &= S_1 = \overline{P_1} = \overline{\overline{x_2} \cdot \overline{x_1} \cdot x_0} \\
 y_2 &= S_2 = \overline{P_2} = \overline{\overline{x_2} \cdot x_1 \cdot \overline{x_0}} \\
 y_3 &= S_3 = \overline{P_3} = \overline{\overline{x_2} \cdot x_1 \cdot x_0} \\
 y_4 &= S_4 = \overline{P_4} = \overline{x_2 \cdot \overline{x_1} \cdot \overline{x_0}} \\
 y_5 &= S_5 = \overline{P_5} = \overline{x_2 \cdot \overline{x_1} \cdot x_0} \\
 y_6 &= S_6 = \overline{P_6} = \overline{x_2 \cdot x_1 \cdot \overline{x_0}} \\
 y_7 &= S_7 = \overline{P_7} = \overline{x_2 \cdot x_1 \cdot x_0}
 \end{aligned}$$

Observație:
Implementează la ieșire toți mintermii negați (maxtermi).

Circuite logice combinaționale

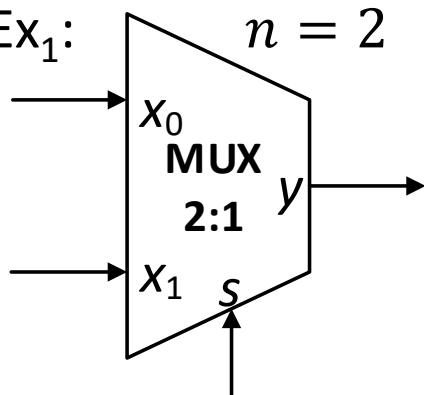
Circuite MSI uzuale – Multiplexoare (MUX)

- Datele de la una din intrări sunt trecute la ieșirea care este unică.
- Selecția se face pe baza unui cod dat de **semnalele de selecție**.
- Pentru m biți de selecție numărul de intrări este $n = 2^m$.



$$y = x_k, k = s_{m-1} \times 2^{m-1} + s_{m-2} \times 2^{m-2} + \dots + s_0 \times 2^0 \text{ (nr. de combinație)}$$

Ex₁: $n = 2$



s	y
0	x_0
1	x_1



s	x_0	x_1	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$s \backslash x_0 x_1$	00	01	11	10
0	0	0	1	1
1	0	1	1	0

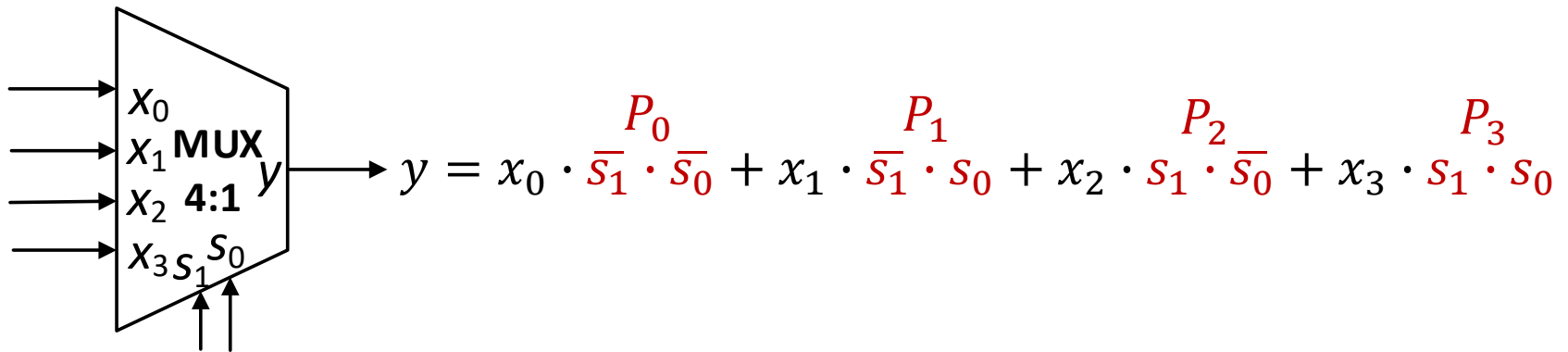
$$y = x_0 \cdot \bar{s} + x_1 \cdot s$$

P_0, P_1 = mintermi de variabila s

Circuite logice combinaționale

Circuite MSI uzuale – **Multiplexoare (MUX)**

Ex₂: $n = 4$



Obs₁: Implementează SAU peste mintermii de s_1, s_0 pentru care $x_i = 1$.

└─> FCD pentru $f(s_1, s_0)$

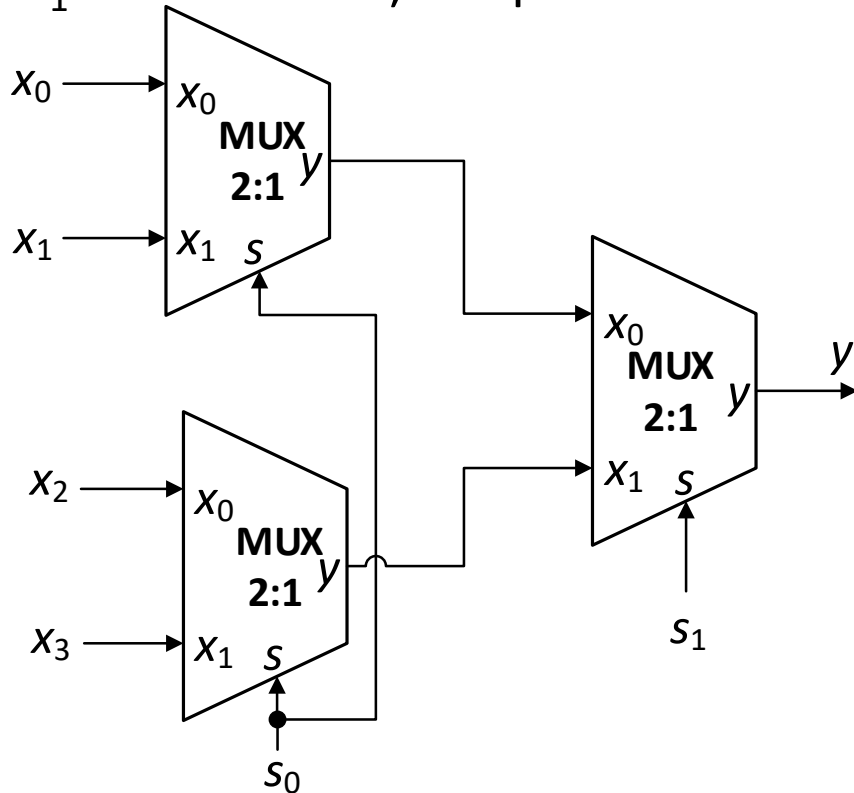
Obs₂: În funcție de valorile 0 sau 1 pe intrările x_i se activează/dezactivează mintermi de $s_1, s_0 \Rightarrow y =$ diverse funcții de variabilele s_1 și s_0 .

Circuite logice combinaționale

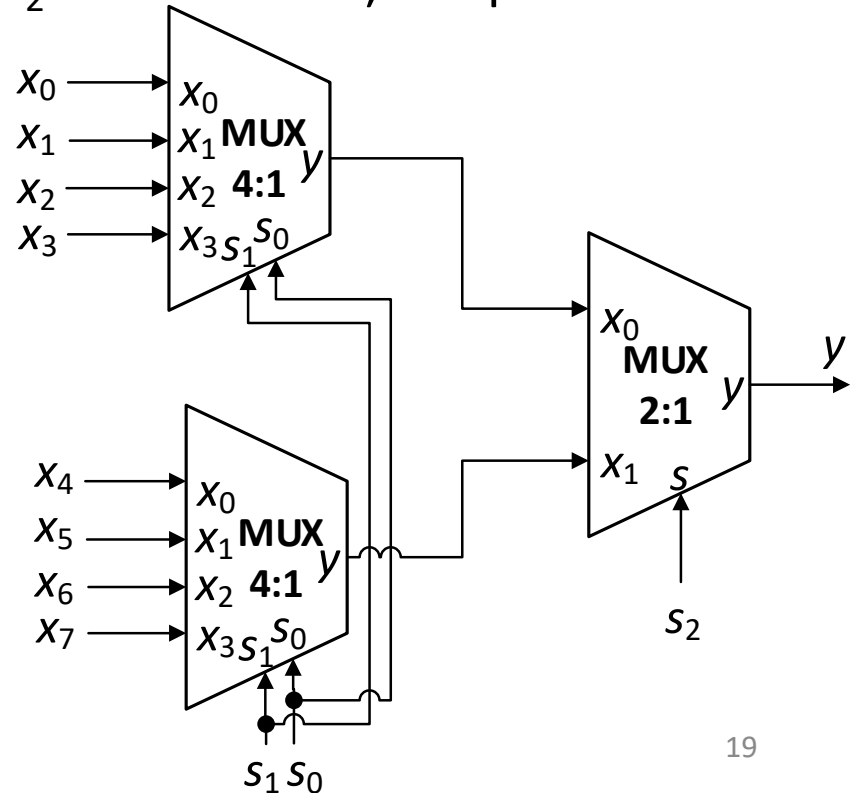
Circuite MSI uzuale – **Multiplexoare (MUX)**

- Există MUX 2:1, 4:1, 8:1, 16:1, 32:1, 64:1, etc.
- Pot prezenta suplimentar ieșirea negată și un semnal de activare (Enable).
- Cele cu număr mare de intrări se pot realiza prin cascadarea celor cu număr mai mic de intrări.

Ex₁: **MUX 4:1** obținut prin cascada



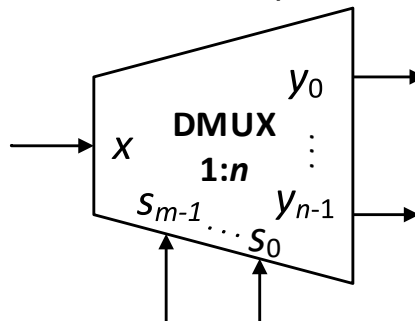
Ex₂: **MUX 8:1** obținut prin cascada



Circuite logice combinaționale

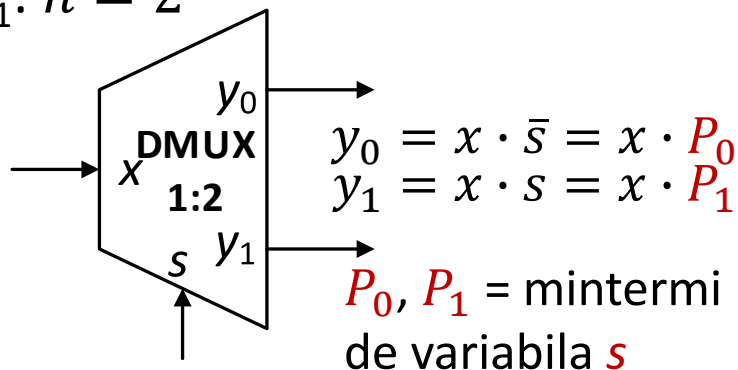
Circuite MSI uzuale – **Demultiplexoare (DMUX)**

- Datele de la intrarea (unică) sunt trecute la una din ieșirile selectate.
- Selecția se face pe baza unui cod dat de semnalele de selecție.
- Pentru m biți de selecție numărul de ieșiri este $n = 2^m$.

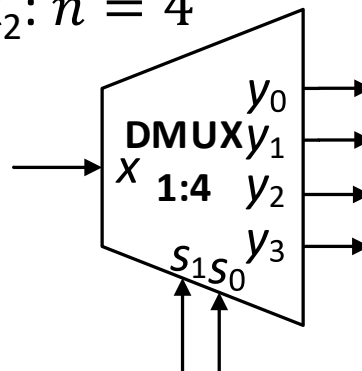


$$y_k = \begin{cases} x, & k = s_{m-1} \times 2^{m-1} + s_{m-2} \times 2^{m-2} + \dots + s_0 \times 2^0 \text{ (nr. de combinație)} \\ 0, & \text{altfel} \end{cases}$$

Ex₁: $n = 2$



Ex₂: $n = 4$



$$\begin{aligned} y_0 &= x \cdot \bar{s}_1 \cdot \bar{s}_0 = x \cdot P_0 \\ y_1 &= x \cdot \bar{s}_1 \cdot s_0 = x \cdot P_1 \\ y_2 &= x \cdot s_1 \cdot \bar{s}_0 = x \cdot P_2 \\ y_3 &= x \cdot s_1 \cdot s_0 = x \cdot P_3 \end{aligned}$$

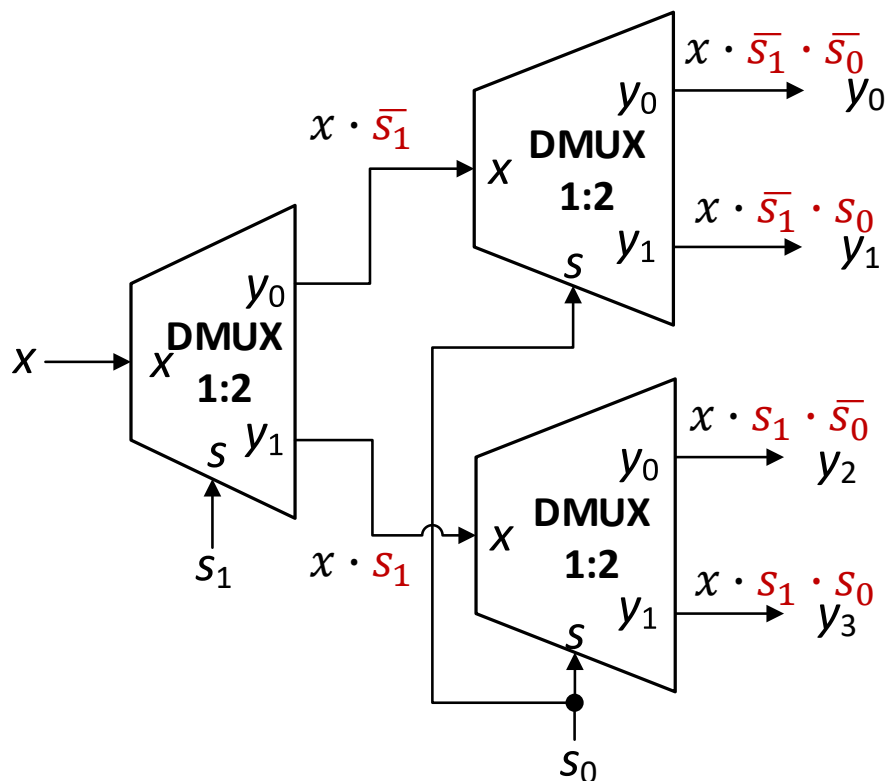
Observație: Dacă $x = 1$ implementează la ieșire toți **mintermi** de ²⁰ s_1 și s_0 .

Circuite logice combinaționale

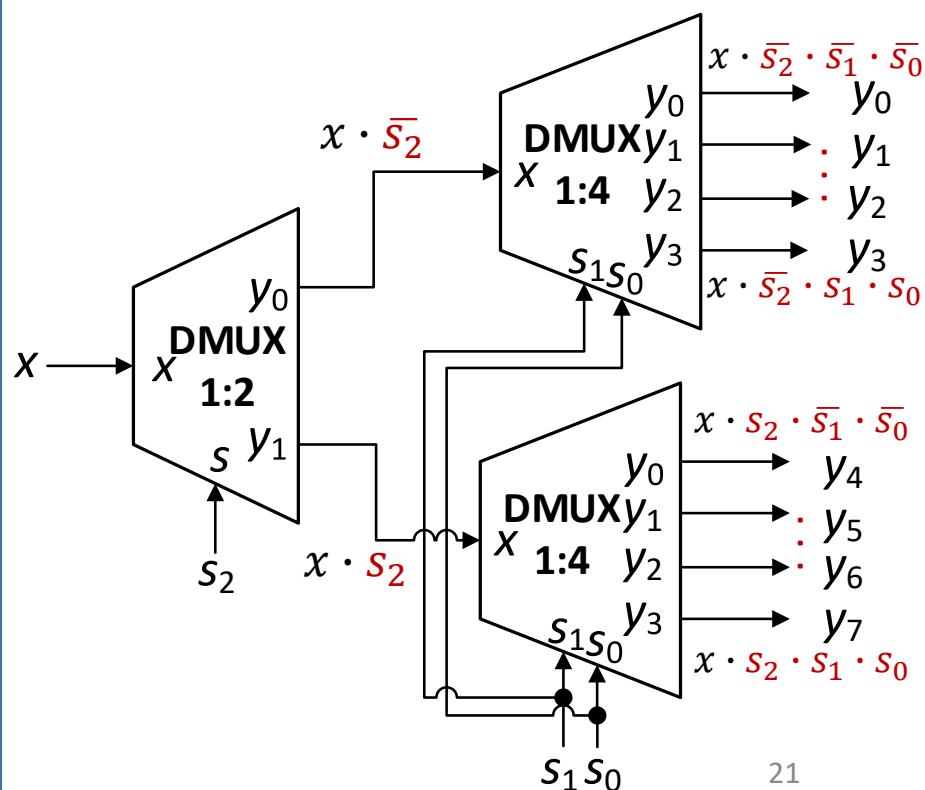
Circuite MSI uzuale – **Demultiplexoare (DMUX)**

- Există DMUX 1:2, 1:4, 1:8, 1:16, 1:32, 1:64, etc.
- Cele cu număr mare de ieșiri se pot realiza prin cascada celor cu număr mai mic de ieșiri.

Ex₁: **DMUX 1:4** obținut prin cascada



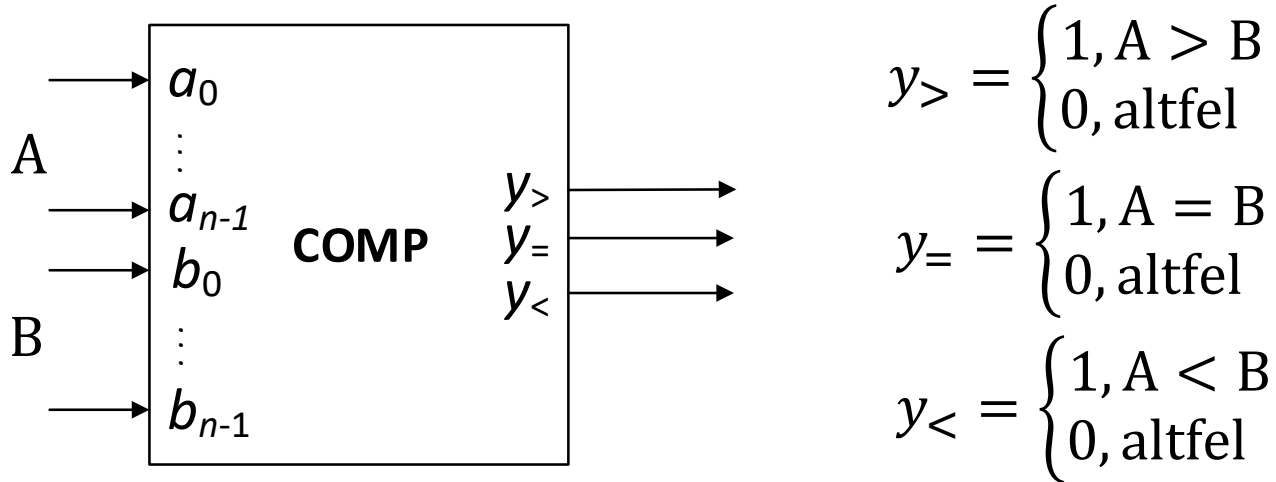
Ex₂: **DMUX 1:8** obținut prin cascada



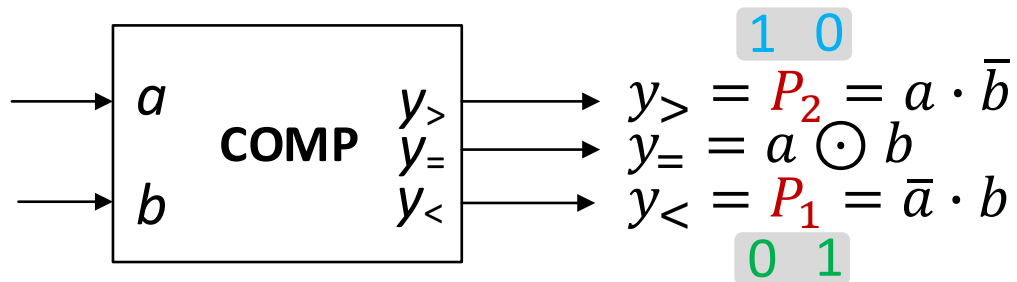
Circuite logice combinaționale

Circuite MSI uzuale – Comparatoare numerice

- Determină la ieșire relația de mărime dintre 2 numere aplicate pe intrare.



Ex: $n = 1$ – Comparator pe 1 bit

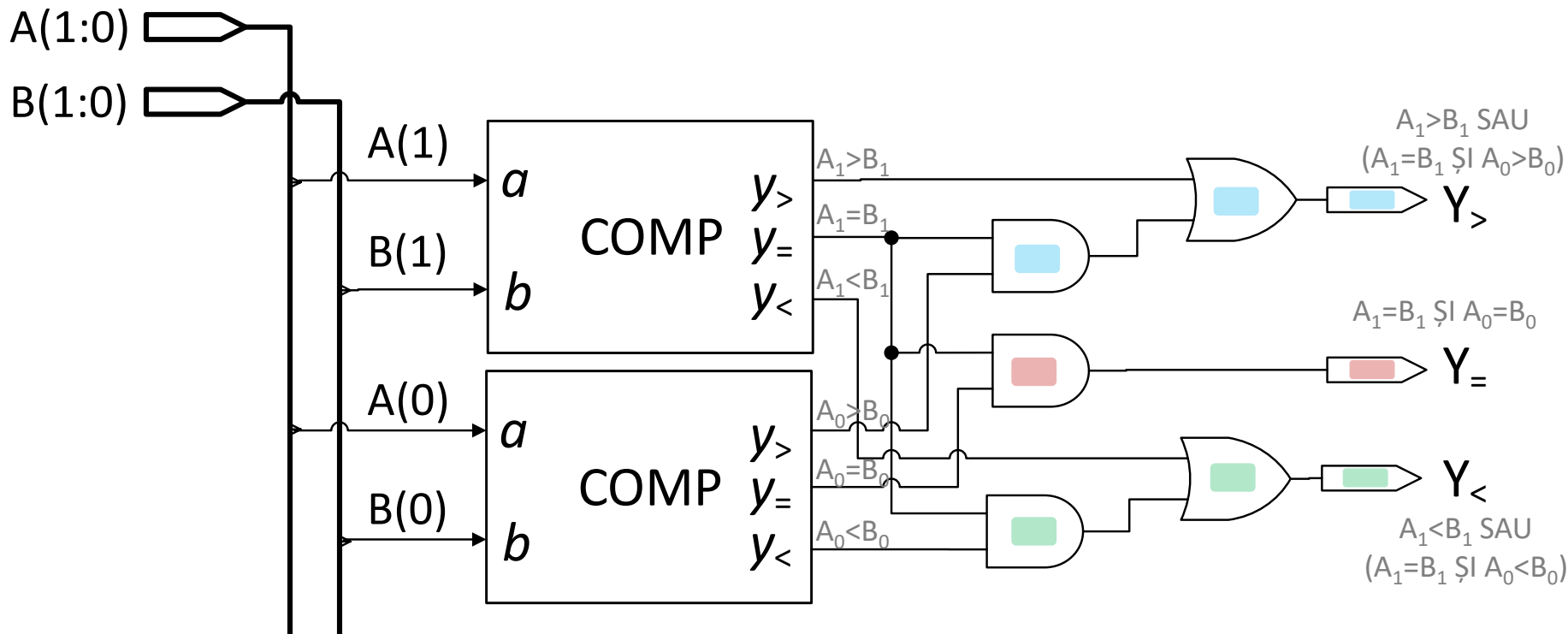


Circuite logice combinaționale

Circuite MSI uzuale – Comparatoare numerice

- Comparatoarele pe mai mulți biți se pot realiza din comparatoare pe mai puțini biți și porți logice.

Ex: $n = 2$ – Comparator pe 2 biți realizat prin cascada

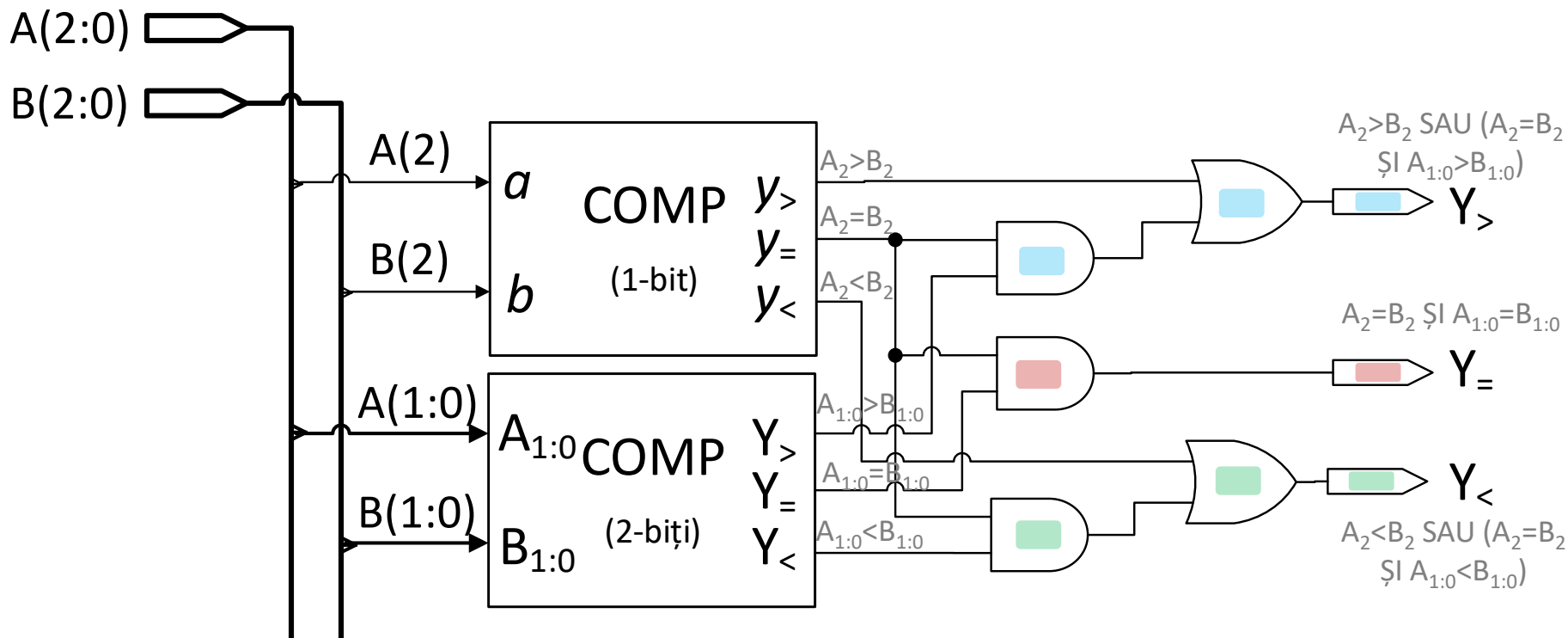


Circuite logice combinaționale

Circuite MSI uzuale – Comparatoare numerice

- Comparatoarele pe mai mulți biți se pot realiza din comparatoare pe mai puțini biți și porți logice.

Ex: $n = 3$ – Comparator pe 3 biți realizat prin cascaderare



Obs: Prin cascaderare se poate genera treptat pentru orice număr de biți. ²⁴

Circuite logice combinaționale

Circuite MSI uzuale – **Detectoare/generatoare de paritate**

- Sunt utilizate la detecția erorilor de transmisie.
- Paritate **pară/impară** = numărul de biți de 1 ai unui cuvânt transmis trebuie să fie par/impar.
- Generator – generează 1 bit astfel încât concatenat la cuvântul de transmis să se respecte paritatea utilizată. Rezultatul concatenării se transmite pe canalul de comunicație.
- Detector – semnalează dacă valorile primite la destinație codifică un cuvânt corect sau eronat din punct de vedere al parității.
- Sunt construite din porți **XOR**. Observație:

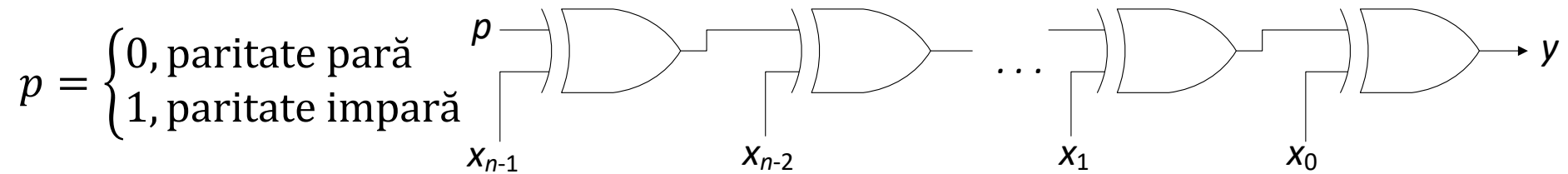
$$x_0 \oplus x_1 \oplus \dots \oplus x_{n-1} = \begin{cases} 0, & (x_0, x_1, \dots, x_{n-1}) \text{ conține un număr } \textbf{par} \text{ de } 1 \\ 1, & (x_0, x_1, \dots, x_{n-1}) \text{ conține un număr } \textbf{impar} \text{ de } 1 \end{cases}$$

Circuite logice combinaționale

Circuite MSI uzuale – **Detectoare/generatoare de paritate**

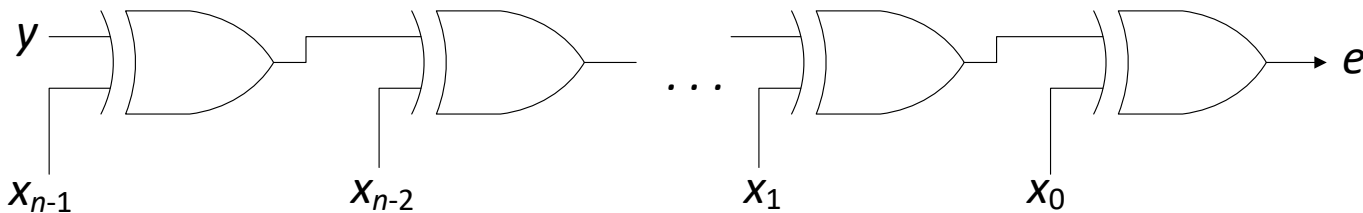
- Pentru un număr mai mare de biți aceste unități se pot realiza prin cascaderare de unități XOR.

Generator pe n biți realizat prin cascaderare



- Se va transmite cuvântul: **$yx_{n-1}x_{n-2} \dots x_1x_0$** .

Detector pe $n + 1$ biți realizat prin cascaderare



Paritate pară

$$e = \begin{cases} 1, \text{eroare} \\ 0, \text{corect} \end{cases}$$

Paritate impară

$$e = \begin{cases} 0, \text{eroare} \\ 1, \text{corect} \end{cases}$$

Circuite logice combinaționale

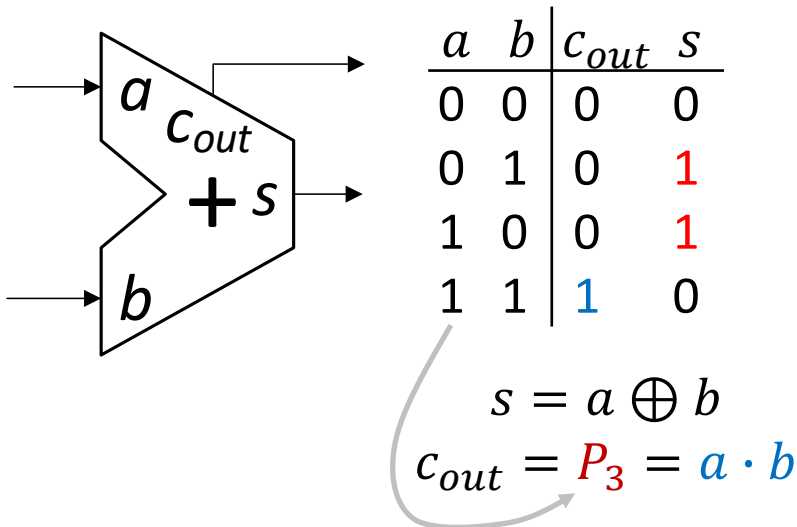
Circuite MSI uzuale – Sumatoare/Scăzătoare

- Realizează adunarea/scăderea în baza 2.

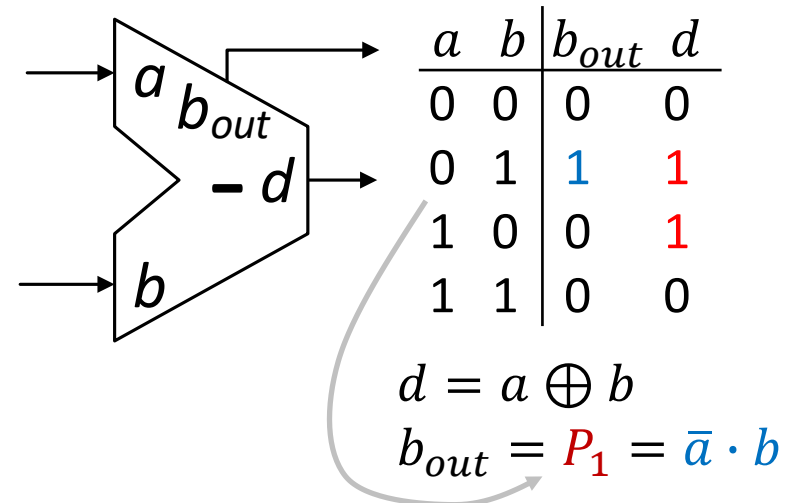
Semisumator/semiscăzător pe 1 bit

- Realizează suma/diferența pe 1 bit fără a ține cont de bitul de transport (carry)/împrumut (borrow).

Semisumator: $a + b$



Semiscăzător: $a - b$



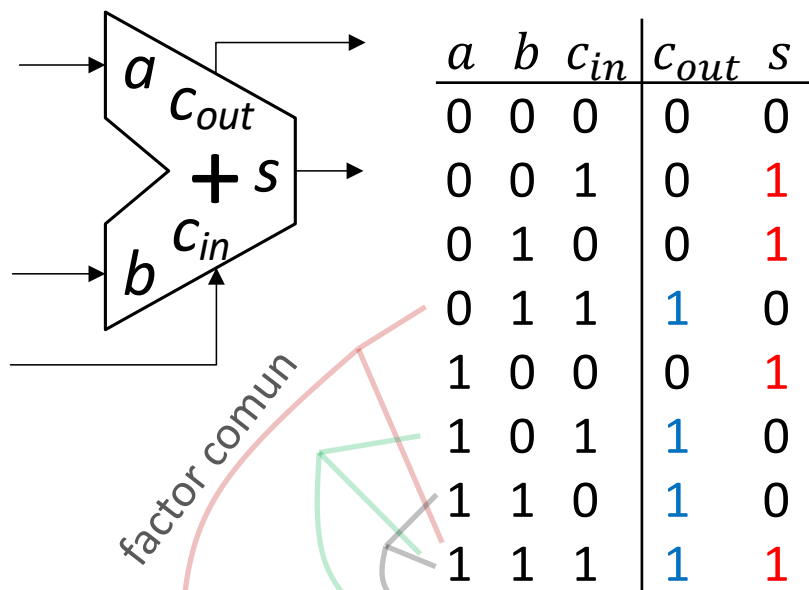
Circuite logice combinaționale

Circuite MSI uzuale – Sumatoare/Scăzătoare

Sumator/Scăzător (complet) pe 1 bit

- Realizează suma/diferența pe 1 bit ținând cont de bitul de transport (carry)/împrumut (borrow).

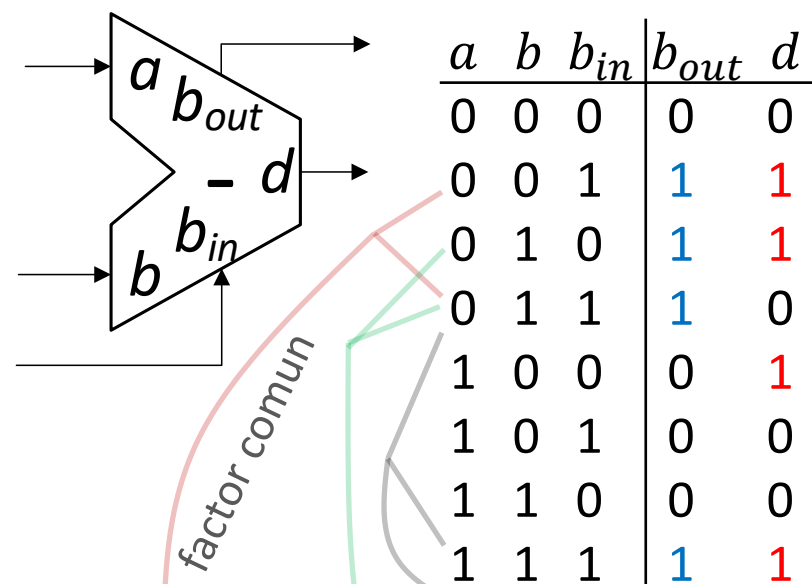
Sumator: $a + b + c_{in}$



$$c_{out} = (b \cdot c_{in}) + (a \cdot c_{in}) + (a \cdot b)$$

$$s = a \oplus b \oplus c_{in} \text{ (nr. impar de 1 pe } a, b, c_{in})$$

Scăzător: $a - b - b_{in}$



$$b_{out} = (\bar{a} \cdot b_{in}) + (\bar{a} \cdot b) + (b \cdot b_{in})$$

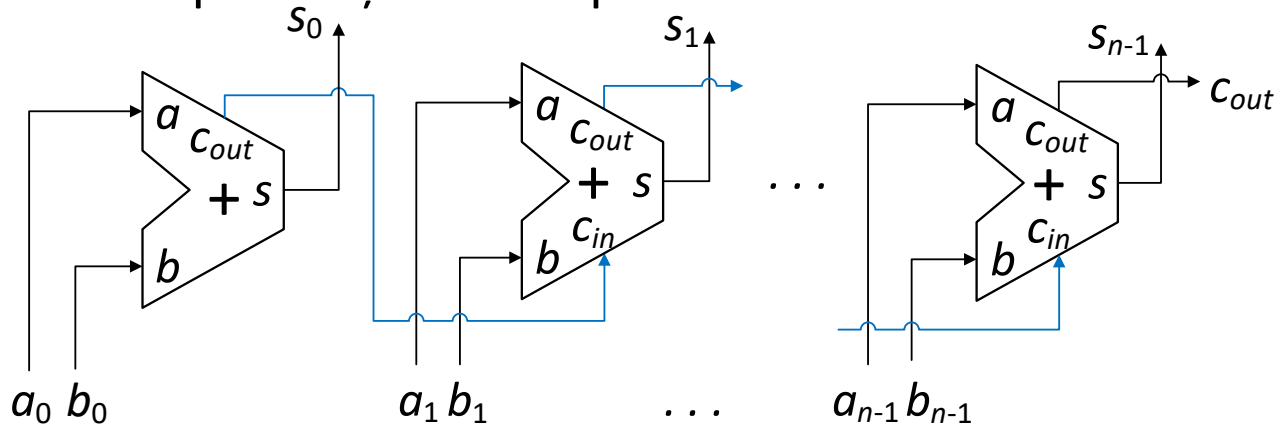
$$d = a \oplus b \oplus b_{in} \text{ (nr. impar de 1 pe } a, b, b_{in})$$

Circuite logice combinaționale

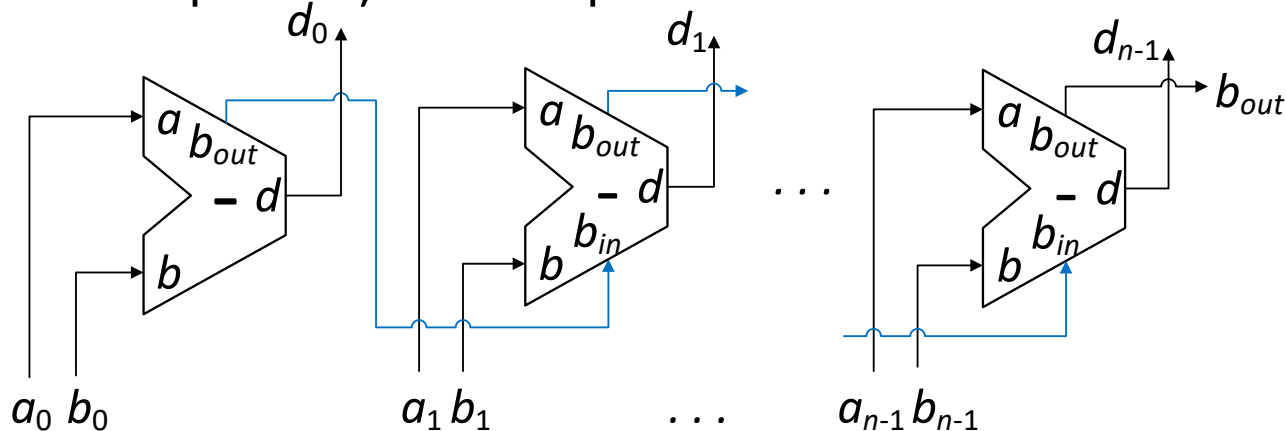
Circuite MSI uzuale – **Sumatoare/Scăzătoare**

- Pentru un număr mai mare de biți aceste unități se pot realiza prin cascaderare de unități pe 1 bit.

Sumator pe n biți realizat prin cascaderare



Scăzător pe n biți realizat prin cascaderare

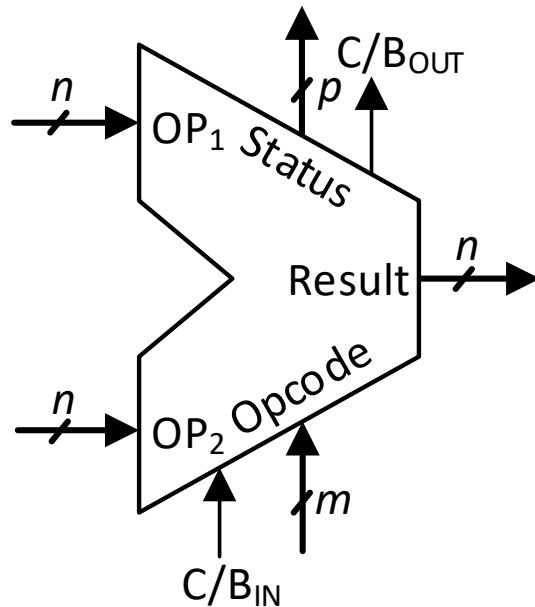


Circuite logice combinaționale

Arithmetic-Logic Unit

Circuite MSI uzuale – Unități Aritmetico-Logice (UAL/ALU)

- Realizează următoarele tipuri de operații pe operanzi cu 1 sau mai mulți biți:
 - Operații aritmetice: adunare, scădere, incrementare, decrementare, comparare.
 - Operații logice: ȘI, SAU, NOT, ȘI-NU, SAU-NU, XOR, XNOR.



- Opcode – cod binar care specifică operația de efectuat \Rightarrow până la 2^m operații
- Status – set de biți care oferă informații despre operanzi sau rezultat: is-zero, =, >, <, depășire, paritate

- Operațiile pe mai mulți biți se pot realiza prin cascada de unități UAL de dimensiuni mai mici.

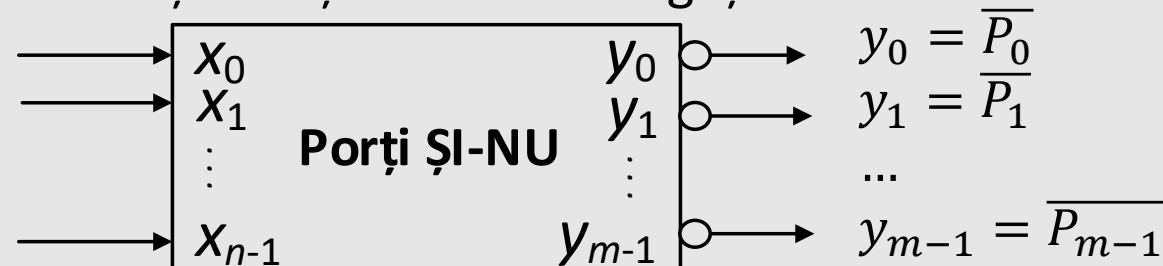
Circuite logice combinaționale

Sinteza funcțiilor booleene cu circuite **MSI**

- Nu este necesar pasul de minimizare. **Funcțiile se aduc la forma canonică.**
- **Circuit universal** = implementează toți termenii canonici de variabilele sale.
- Obs₁: **Decodificatorul** (DCD) implementează toți termenii canonici ai variabilelor de intrare => circuit universal.
- Obs₂: **Multiplexorul** (MUX) și **Demultiplexorul** (DMUX) implementează toți termenii canonici ai variabilelor de selecție => circuit universal.

Sinteza funcțiilor booleene cu **DCD**

- DCD implementează la ieșire toți mintermii negați.



- Obs: Dacă se aplică ȘI-NU peste un subset de ieșiri se obține (aplicând **De Morgan**) o formă canonică disjunctivă (FCD) ce include **mintermii** asociați.

$$\overline{y_{i1} \cdot y_{i2} \cdot \dots \cdot y_{ik}} = \overline{\overline{P_{i1}} \cdot \overline{P_{i2}} \cdot \dots \cdot \overline{P_{ik}}} = P_{i1} + P_{i2} + \dots + P_{ik}$$

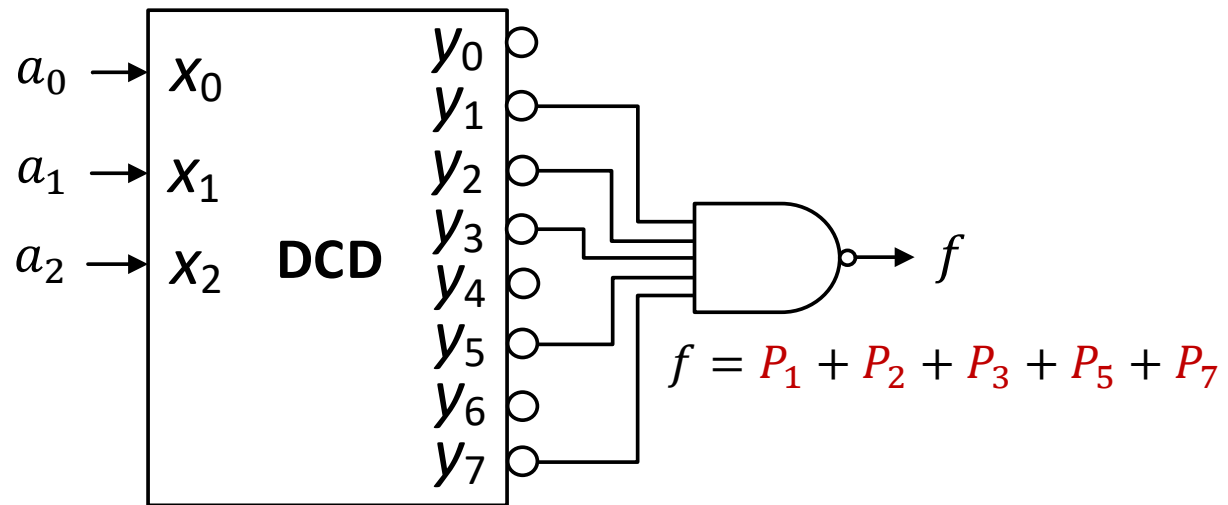
Circuite logice combinaționale

Sinteza funcțiilor booleene cu **DCD**

- Obs: Pentru a implementa o funcție scrisă în forma canonică disjunctivă (FCD) este suficient să se aplice ȘI-NU peste ieșirile DCD corespunzătoare mintermilor funcției.

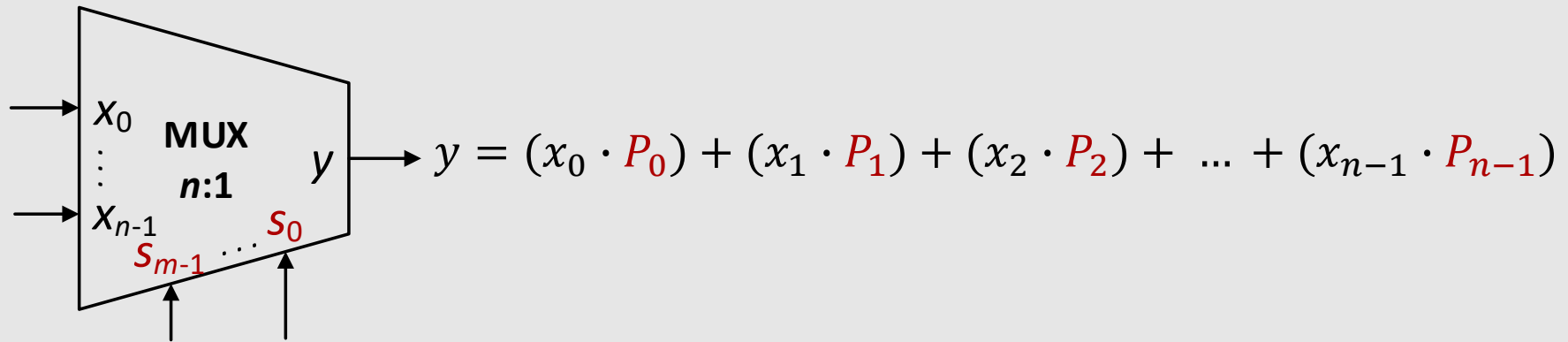
Ex: $n = 3$ $f = (a_0 \cdot a_1) + (a_0 \cdot \overline{a_1}) + (a_1 \cdot \overline{a_2}) \xRightarrow{\text{FCD}} \sum(1,2,3,5,7)$

	a_2	a_1	a_0	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1



Circuite logice combinaționale

Sinteza funcțiilor booleene cu **MUX**



- Obs: MUX implementează forma canonică disjunctivă (FCD) care conține **mintermii** semnalelor de selecție în conjuncție ȘI cu intrările x_i .
- **Metoda:** O funcție de n variabile se poate implementa cu un MUX având **$n - 1$ variabile de selecție**. Funcția se aduce în forma canonică disjunctivă (FCD). Se scoate factor comun fiecare minterm generat de cele $n - 1$ variabile și **termenul care se obține în conjuncție ȘI cu fiecare minterm va fi valoarea semnalului x_i asociat mintermului respectiv.**

Circuite logice combinaționale

Sinteza funcțiilor booleene cu **MUX**


Ex: $n = 3$ $f = (a_0 \cdot a_1) + (a_0 \cdot \overline{a_1}) + (a_1 \cdot \overline{a_2}) \xRightarrow{\text{FCD}} \sum(1,2,3,5,7)$

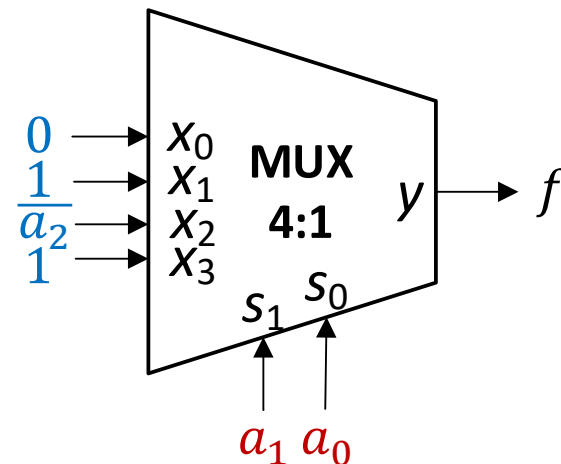
- Se scrie funcția în forma canonică disjunctivă (FCD).
- Se aleg 2 variabile a_1, a_0 ca intrări de selecție ale MUX și se scot factor comun **mintermii** asociați acestora.

	a_2	a_1	a_0	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

$$f \xRightarrow{\text{FCD}} (\overline{a_2} \cdot \overline{a_1} \cdot a_0) + (\overline{a_2} \cdot a_1 \cdot \overline{a_0}) + (\overline{a_2} \cdot a_1 \cdot a_0) + (a_2 \cdot \overline{a_1} \cdot a_0) + (a_2 \cdot a_1 \cdot a_0)$$

$$\begin{aligned}
 &= \overline{a_1} \cdot \overline{a_0} \cdot 0 + \\
 &+ \overline{a_1} \cdot a_0 \cdot (\overline{a_2} + a_2) + \\
 &+ a_1 \cdot \overline{a_0} \cdot \overline{a_2} + \\
 &+ a_1 \cdot a_0 \cdot (\overline{a_2} + a_2) = \\
 &= P_0 \cdot 0 + \\
 &+ P_1 \cdot 1 + \\
 &+ P_2 \cdot \overline{a_2} + \\
 &+ P_3 \cdot 1
 \end{aligned}$$

	a_1	a_0	Mintermii
0	0	0	$P_0 = \overline{a_1} \cdot \overline{a_0}$
1	0	1	$P_1 = \overline{a_1} \cdot a_0$
2	1	0	$P_2 = a_1 \cdot \overline{a_0}$
3	1	1	$P_3 = a_1 \cdot a_0$



Circuite logice combinaționale

Sinteza funcțiilor booleene cu **MUX**

Metoda de rezolvare grafică cu Diagramă Karnaugh

$$\text{Ex}_1: n = 3 \quad f = (a_0 \cdot a_1) + (a_0 \cdot \overline{a_1}) + (a_1 \cdot \overline{a_2}) \xRightarrow{\text{FCD}} \Sigma(1,2,3,5,7)$$

- Se realizează Diagrama Karnaugh în care se grupează celulele pentru care cele 2 intrări de selecție a_1, a_0 sunt identice.

$a_2 \backslash a_1 a_0$	00	01	11	10
0	0	1	1	1
1	0	1	1	0

- Rezultatul fiecărei grupări reprezintă intrarea în MUX corespunzătoare numărului de combinație dat de selecțiile a_1, a_0 .

- Rezultatul fiecărei grupări se calculează astfel:
 - Dacă gruparea conține valori identice atunci rezultatul este valoarea respectivă.
 - Dacă gruparea conține valori distincte atunci se calculează în funcție de celula care conține valoarea 1 astfel:
 - Dacă celula corespunde lui $a_2 = 0$ atunci rezultatul este $\overline{a_2}$.
 - Dacă celula corespunde $a_2 = 1$ atunci rezultatul este a_2 .

Circuite logice combinaționale

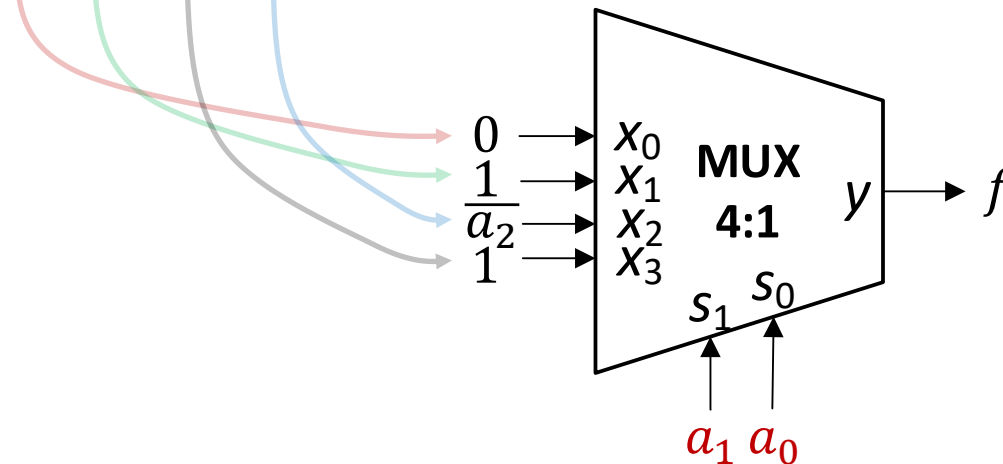
Sinteza funcțiilor booleene cu **MUX**

Metoda de rezolvare grafică cu Diagramă Karnaugh

$$\text{Ex}_1: n = 3 \quad f = (a_0 \cdot a_1) + (a_0 \cdot \overline{a_1}) + (a_1 \cdot \overline{a_2}) \xRightarrow{\text{FCD}} \Sigma(1,2,3,5,7)$$

$a_2 \backslash a_1 a_0$	00	01	11	10
0	0	1	1	1
1	0	1	1	0

- Rezultatul fiecărei grupări reprezintă intrarea în MUX corespunzătoare numărului de combinație dat de selecțiile a_1, a_0 .



Obs: Intrările de date x_i ale MUX vor avea valorile 0, 1 sau $\overline{a_2}$, după caz.

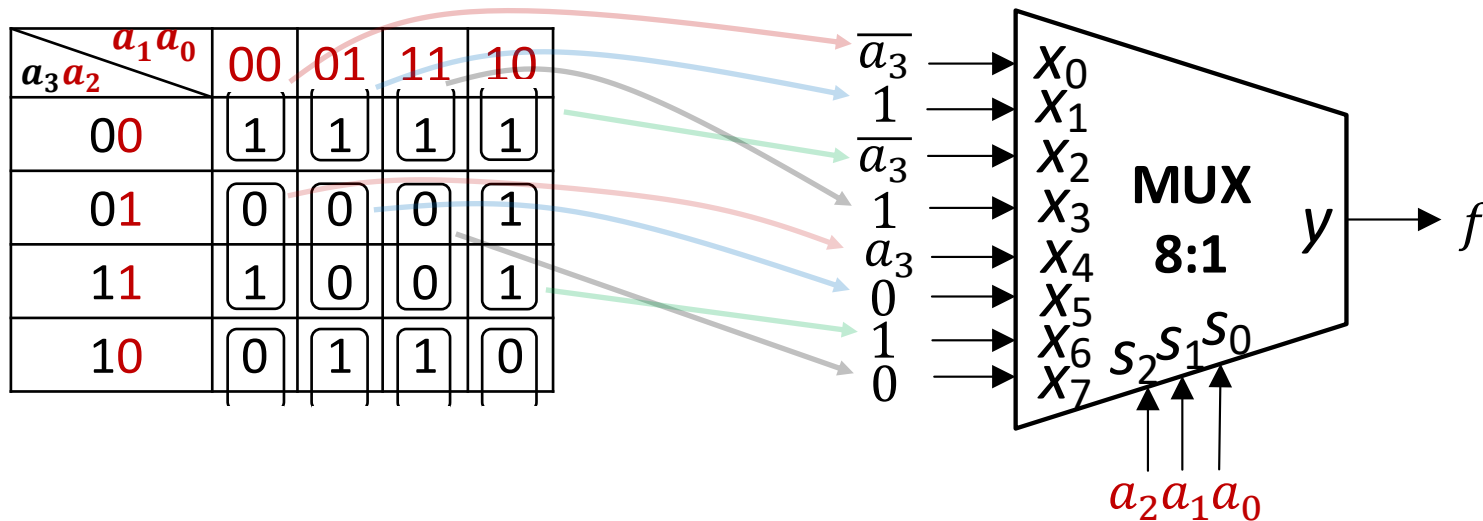
Circuite logice combinaționale

Sinteza funcțiilor booleene cu **MUX**

Metoda de rezolvare grafică cu Diagramă Karnaugh

Ex₂: $n = 4$ $f = \sum(0,1,2,3,6,9,11,12,14)$

- Celulele se grupează după a_2, a_1, a_0 care vor fi intrări de selecție în MUX.



Obs: Intrările de date x_i ale MUX vor avea valorile 0, 1, a_3 sau $\overline{a_3}$, după caz.

Circuite logice combinaționale

Sinteza funcțiilor booleene cu **MUX**

Avantaje:

- Se utilizează un circuit pentru întreaga funcție.
- O singură variabilă trebuie să fie disponibilă atât negată cât și nenegată.

Dezavantaje:

- Numărul variabilelor funcției este limitat de numărul intrărilor de selecție.
- Nu se pretează la implementarea unor funcții foarte simple.