

# Logică digitală



-Curs 7-  
Circuite logice  
combinatoriale

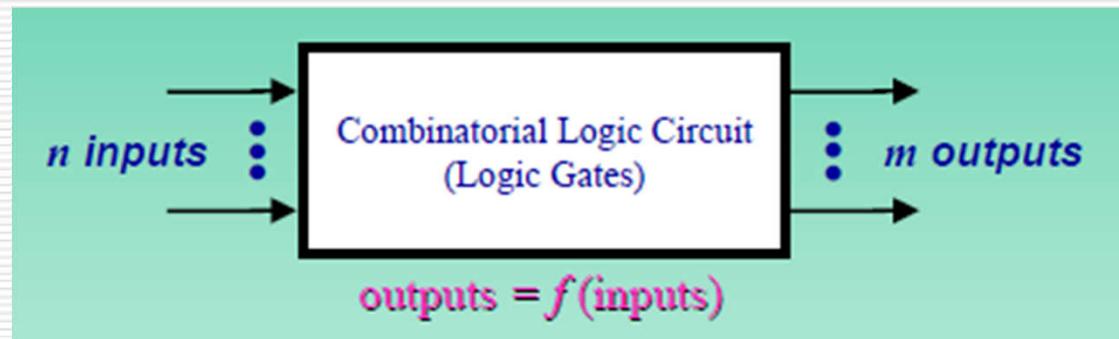
# **Circuite logice combinatională**

---

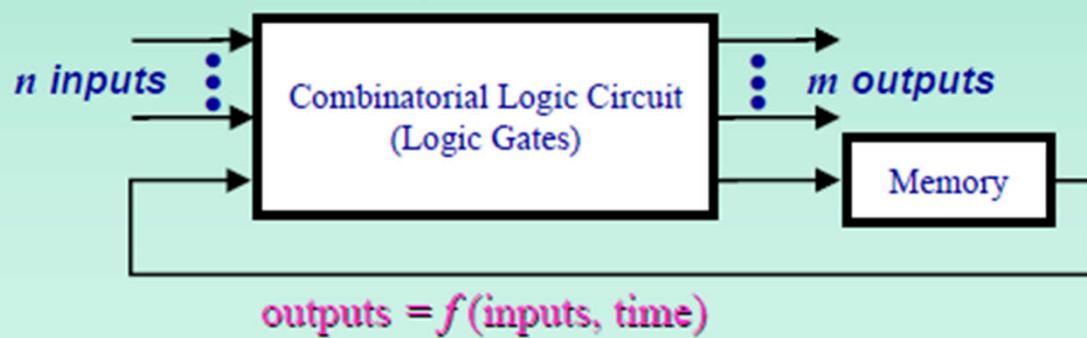
- Circuite de procesare
  - Circuite de conversie
  - Circuite de interconectare
  - Componente universale
-

# Clasificare componente digitale

- Componente combinaționale
  - Ușor de analizat, partaționat, verificat



- Componente secențiale



# Clasificare circuite combinaționale (I)

---

## Procesare

- Operații aritmetice (Adunare, Scădere, Înmulțire, Împărțire)
  - Operații logice (ȘI, SAU-Exclusiv, Negare, etc.)
  - Comparare
  - Operații de manipulare la nivel de bit (shift-are, rotație, ...).
-

# Clasificare circuite combinaționale (II)

---

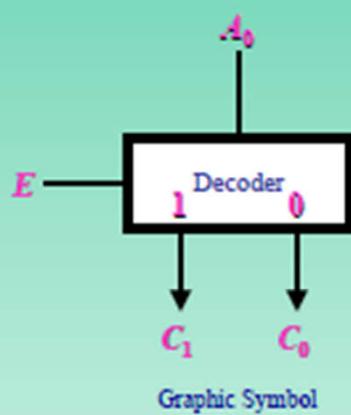
- Conversie date
    - Codificatoare
    - Decodificatoare
  - Interconnect-uri
    - Selectia sursei/destinației
    - Magistrale și interfete magistrală
  - Alte componente (blocuri din UC)
    - ROM
    - PLA
-

# Decodificator

---

- circuite logice combinaționale ce prezintă un număr de  $n$  intrări date și până la  $2^n$  ieșiri, care activează **ieșirea (UNA SINGURĂ)** corespunzătoare valorii combinației vectorului de intrare
- Poate avea intrări de activare, astfel încât ieșirea selectată nu pot fi activată decât dacă intrările de activare sunt active.
- Pt.  $n$  intrări și  $m$  ieșiri  $\rightarrow$  **DEC n-la-m**.
- Uzual sunt folosite pt. activarea (EN) componentelor

# Decodificatorul 1-la-2



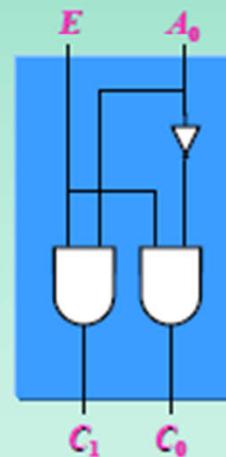
Graphic Symbol

$$\begin{aligned}C_0 &= EA'_0 \\C_1 &= EA_0\end{aligned}$$

Boolean Expression

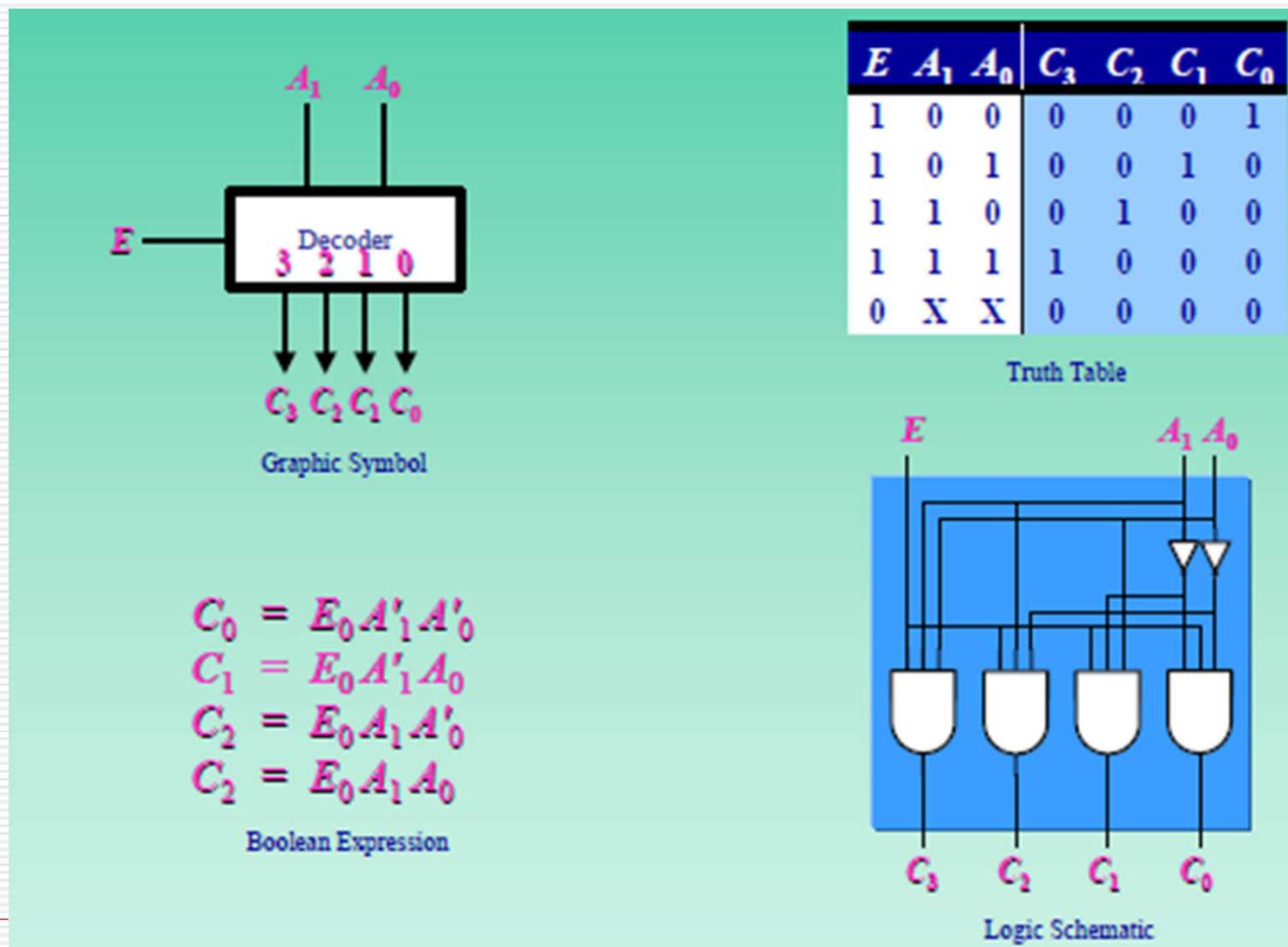
E	A <sub>0</sub>	C <sub>1</sub>	C <sub>0</sub>
1	0	0	1
1	1	1	0
0	X	0	0

Truth Table



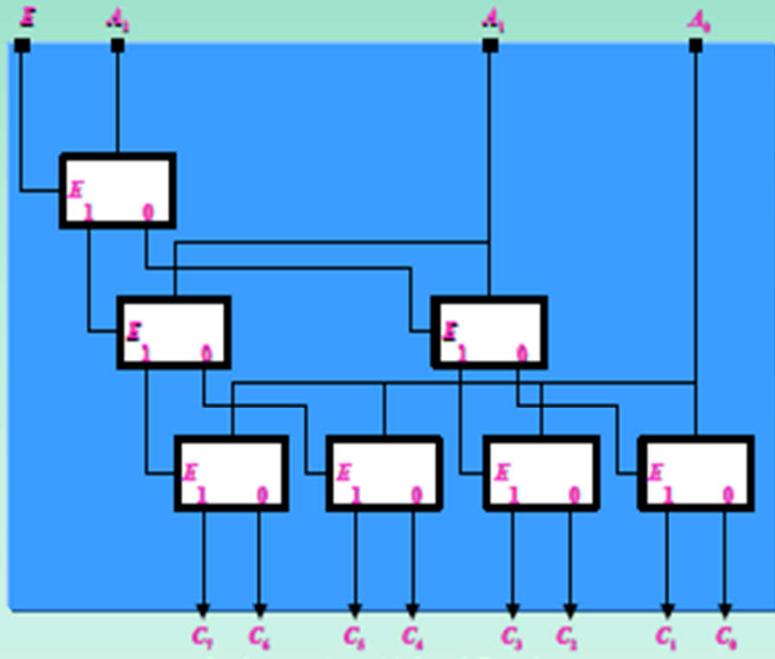
Logic Schematic

# Decodificatorul 2-la-4



# Decodificatorul 3-la-8

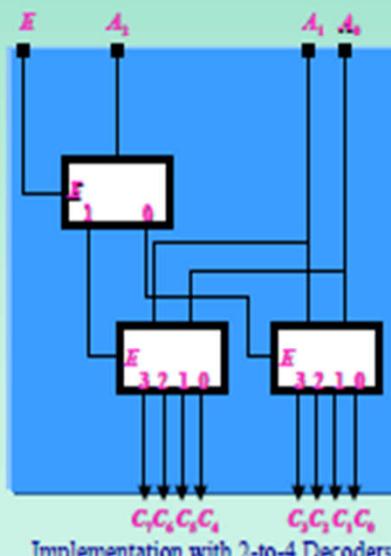
$E$	$A_2$	$A_1$	$A_0$	$C_7$	$C_6$	$C_5$	$C_4$	$C_3$	$C_2$	$C_1$	$C_0$
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0
0	X	X	X	0	0	0	0	0	0	0	0



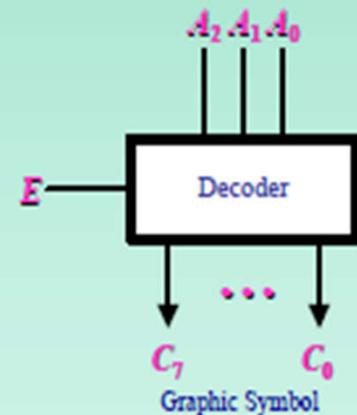
Implementation with 1-to-2 Decoders

Copyright © 2004-2005 by Daniel D. Gajski

Truth Table



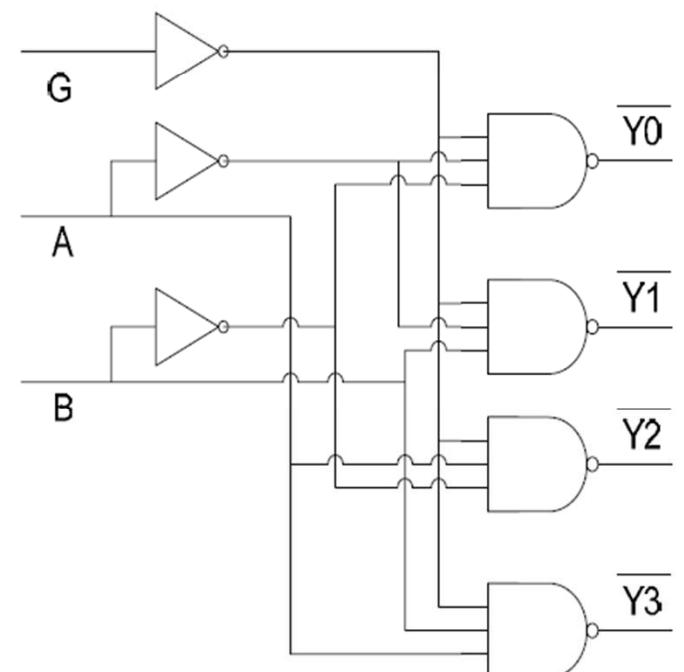
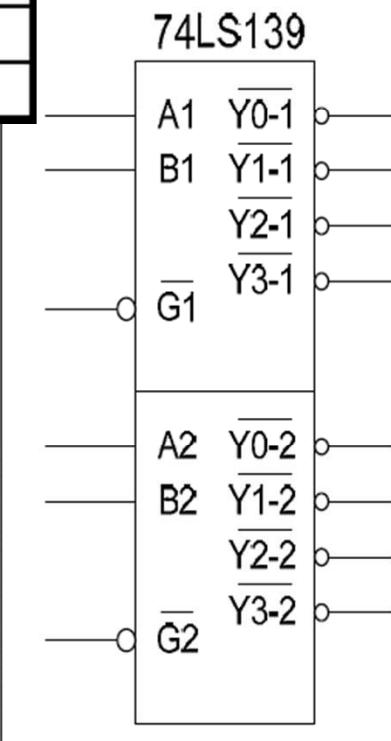
Implementation with 2-to-4 Decoders



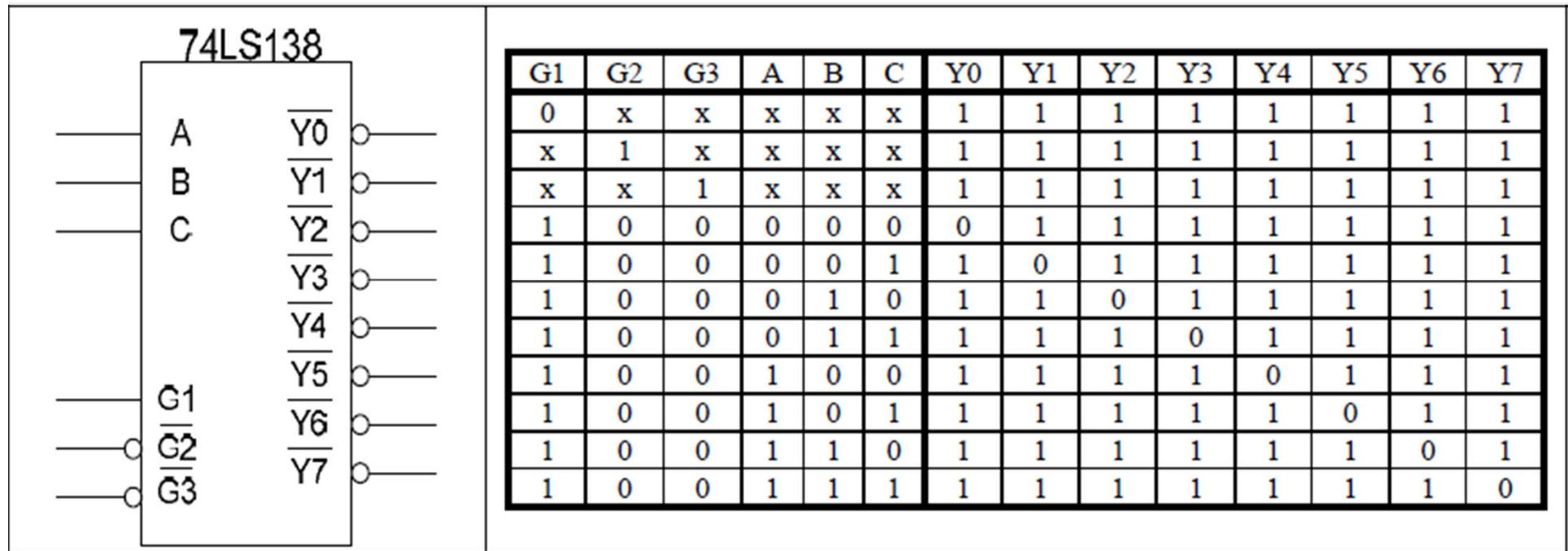
Graphic Symbol

# Circuite integrate pe scară medie ce îndeplinesc funcția de decodificator

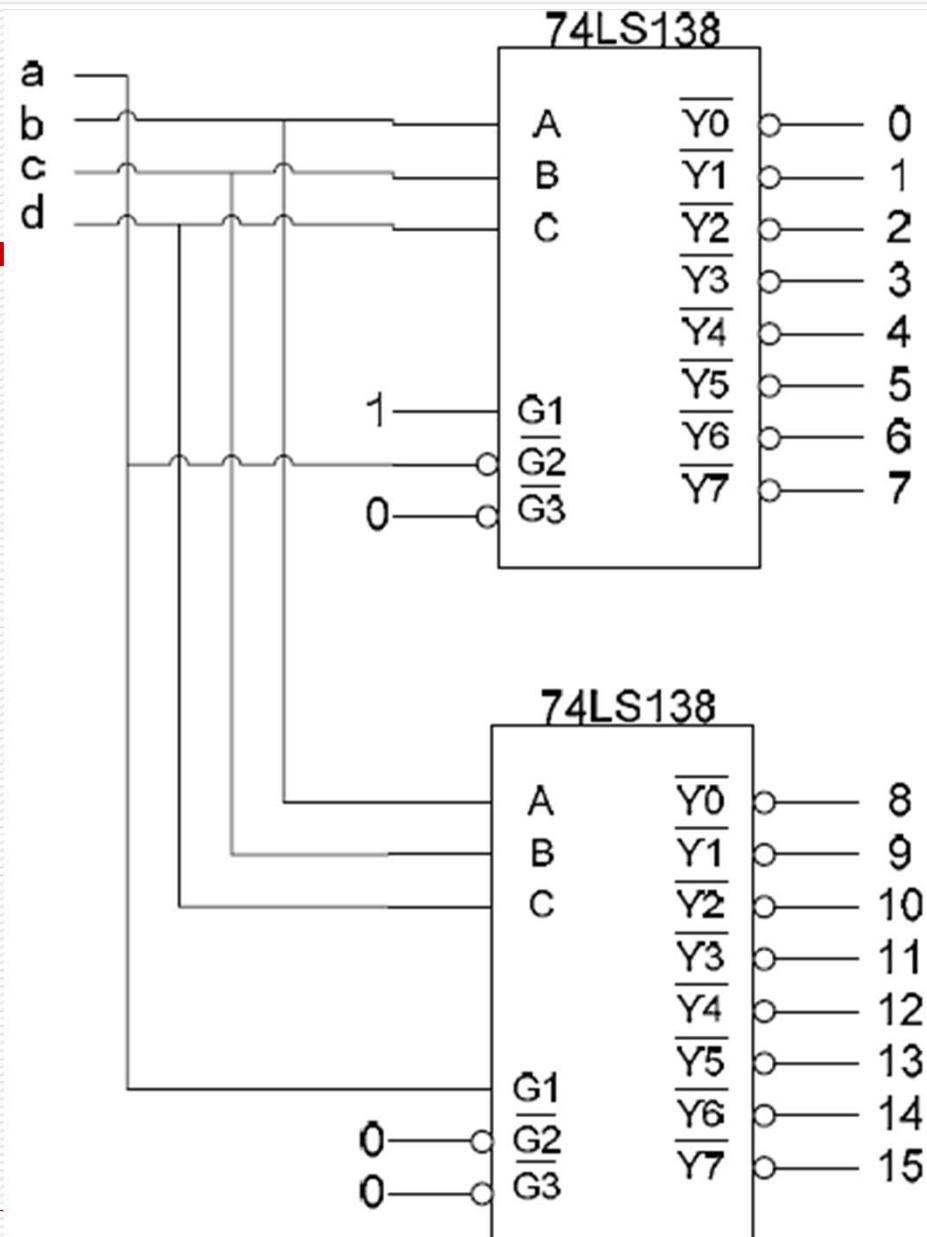
G	A	B	Y0	Y1	Y2	Y3
1	x	x	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



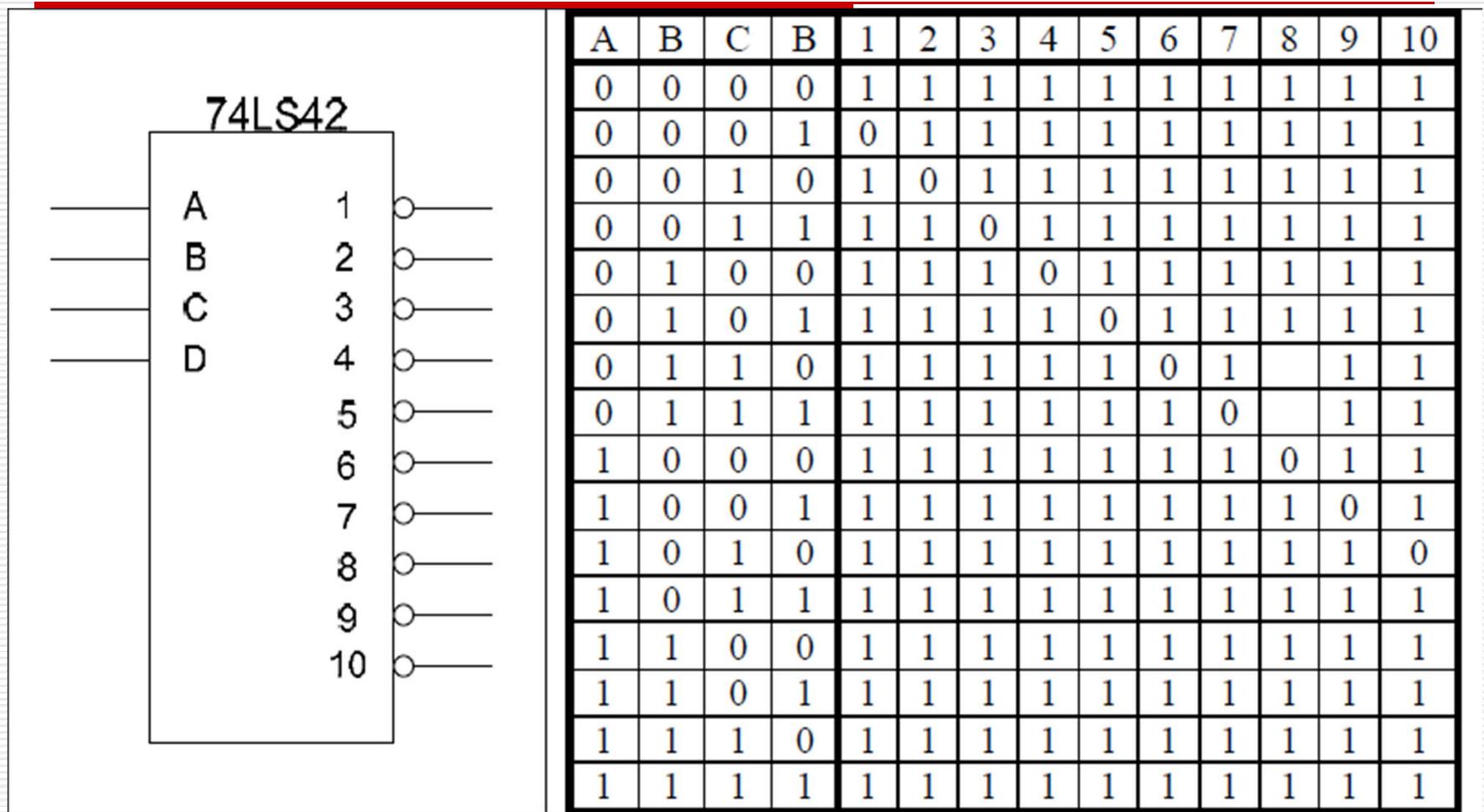
# 74LS138: decodificador 3-1a-8



# DEC 4-la-16



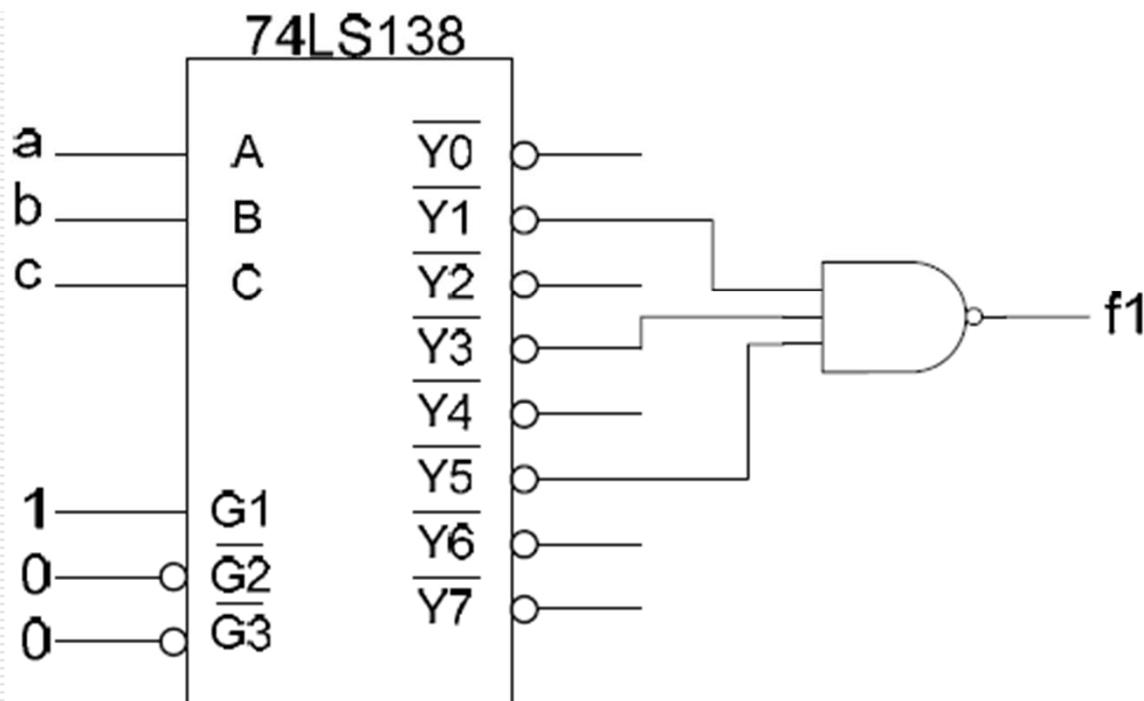
# 14LS42: Decodificador 4-la-10



# Sinteza funcțiilor logice folosind decodificatoare

Să se implementeze cu ajutorul unui decodificator 74LS138 funcția logică

$$f_1(a, b, c) = \sum(1, 3, 5)$$

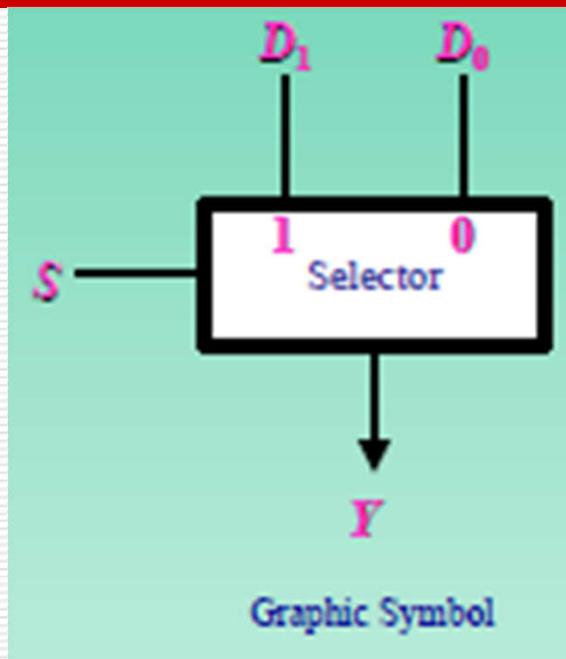


# Multiplexor(Selector)

---

- Multiplexorul este un circuit logic combinațional ce conectează ieșirea acestuia la una din cele  $n$  intrări.
  - Selectia unuia din cele  $n$  intrări se face cu ajutorul a  $\log_2 n$  intrări de selecție.
  - Poate fi privit ca un comutator digital.
  - Este folosit pt. selectia unei singure surse de date din mai multe.
-

# MUX 2-la-1



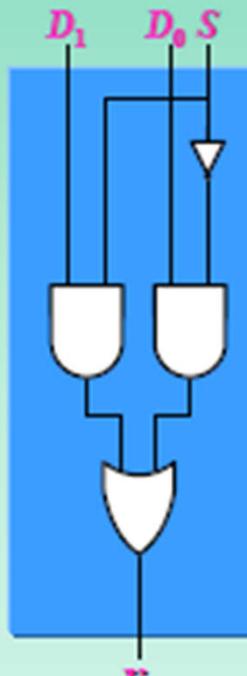
Graphic Symbol

$$Y = S'D_0 + SD_1$$

Boolean Expression

$S$	$Y$
0	$D_0$
1	$D_1$

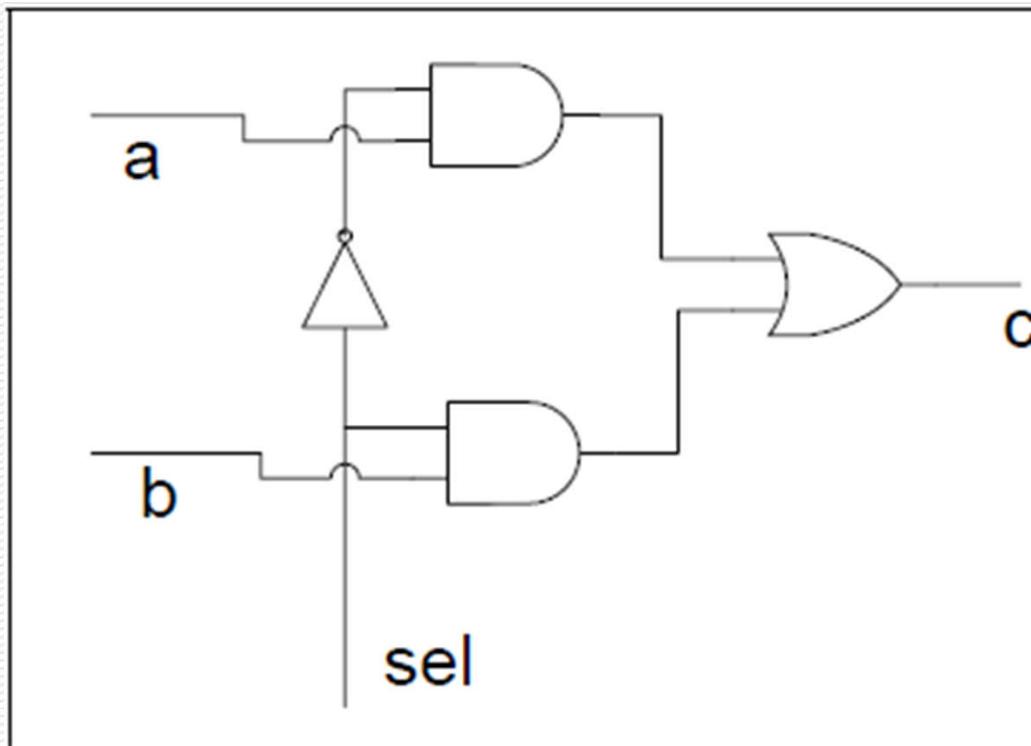
Truth Table



Logic Schematic

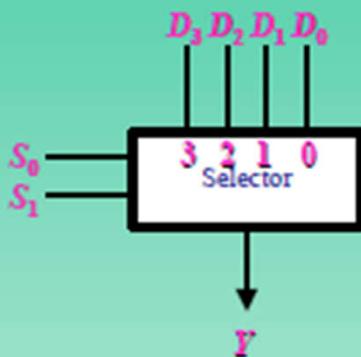
# MUX 2-la-1

---



a	b	sel	c
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

# MUX 4-la-1

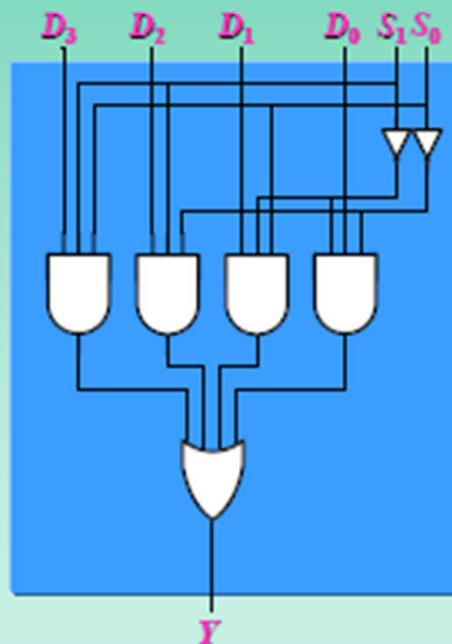


$$Y = S'_1 S'_0 D_0 + S'_1 S_0 D_1 + S_1 S'_0 D_2 + S_1 S_0 D_3$$

Boolean Expression

$S_1$	$S_0$	$Y$
0	0	$D_0$
0	1	$D_1$
1	0	$D_2$
1	1	$D_3$

Truth Table

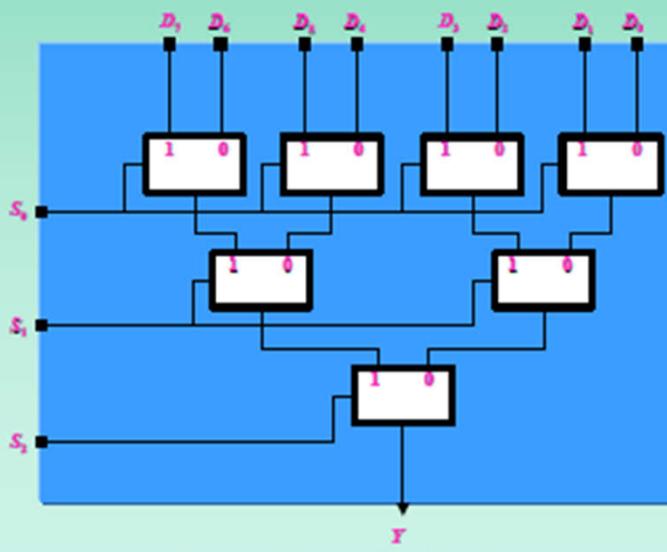


Logic Schematic

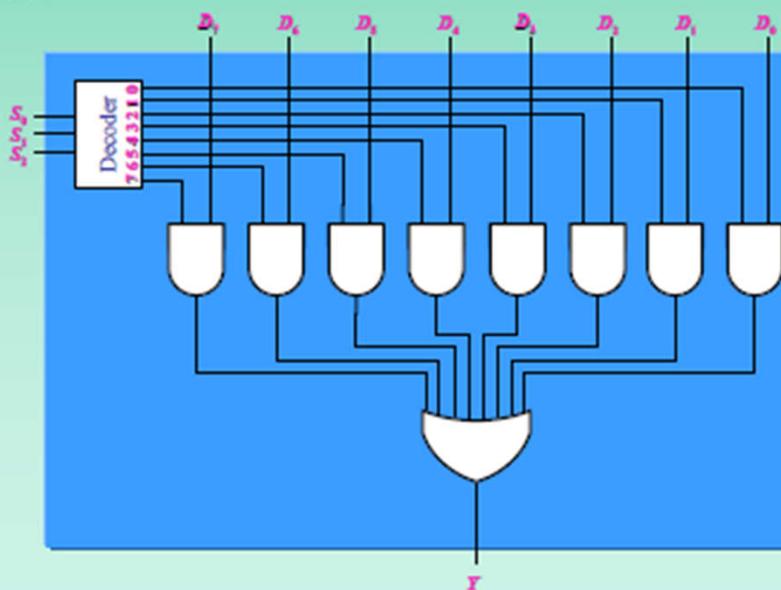
# MUX 8-la-1

$S_2$	$S_1$	$S_0$	$Y$
0	0	0	$D_0$
0	0	1	$D_1$
0	1	0	$D_2$
0	1	1	$D_3$
1	0	0	$D_4$
1	0	1	$D_5$
1	1	0	$D_6$
1	1	1	$D_7$

Truth Table

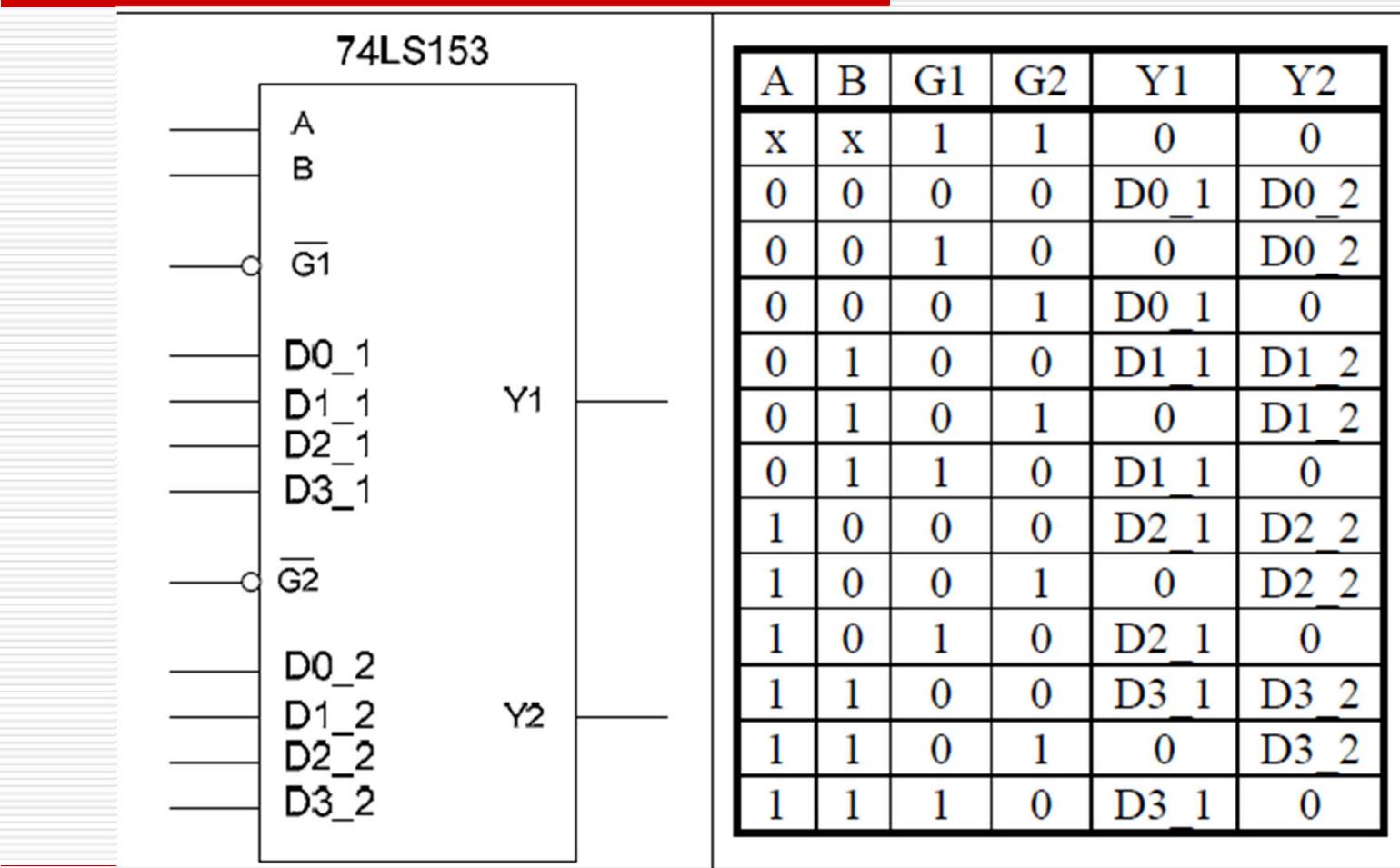


Implementation with 2-to-1 Selectors

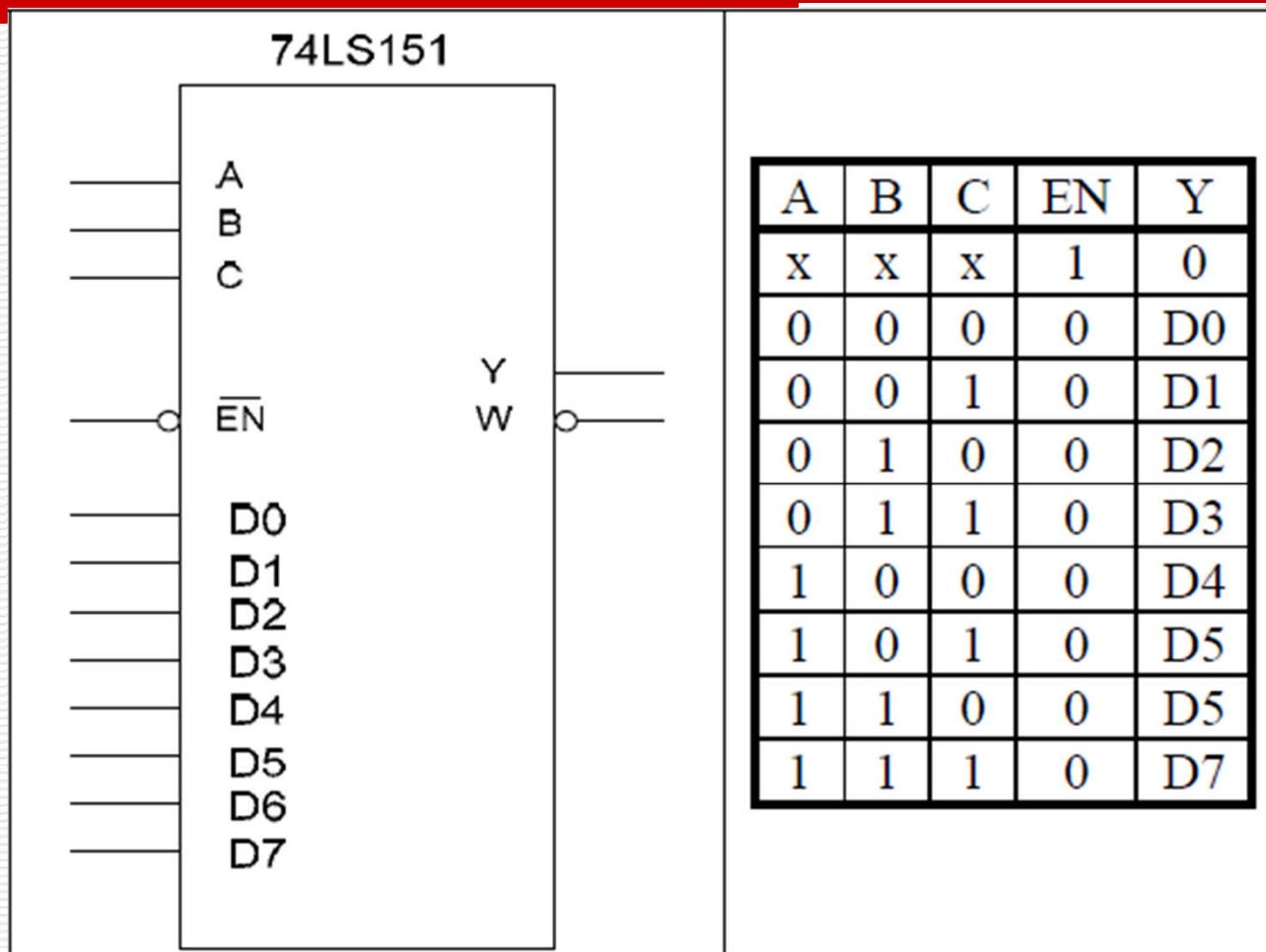


Implementation with 3-to-8 Decoder

# 74LS153: MUX 4-la-1



# 74LS151: MUX 8-la-1

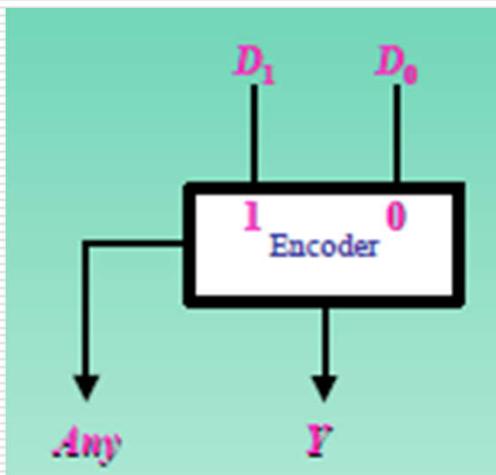


# Codificatoare

---

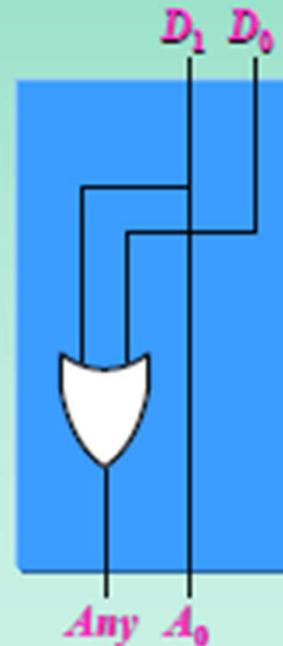
- Circuite combinaționale care realizează într-un sens funcția inversă decodificatoarelor
- Dacă conectăm un decodificator la ieșirea unui codificator nu obținem identitatea!
- Au până la  $2^n$  intrări și un număr de  $n$  ieșiri
- În cele mai multe cazuri valoarea ieșirii unui codificator este dată de indexul celui mai semnificativ bit de intrare activ
- exemplu de aplicație: arbitrarea accesului la o resursă (ex. magistrală, controler întreruperi) → **codificator de prioritate**

# Codificador 2-la-1

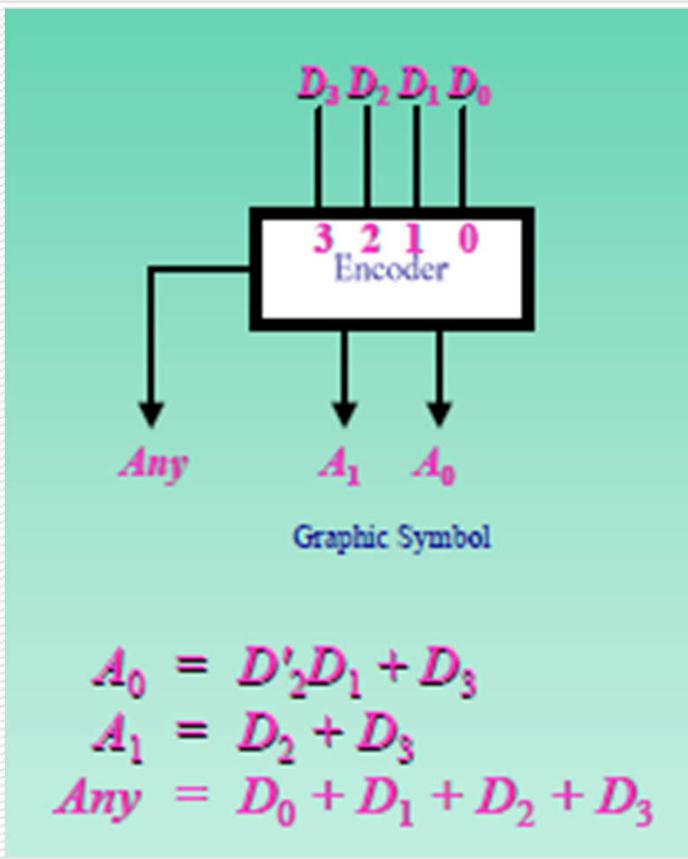


$D_1$	$D_0$	$A_0$	Any
0	0	0	0
0	1	0	1
1	X	1	1

Truth Table

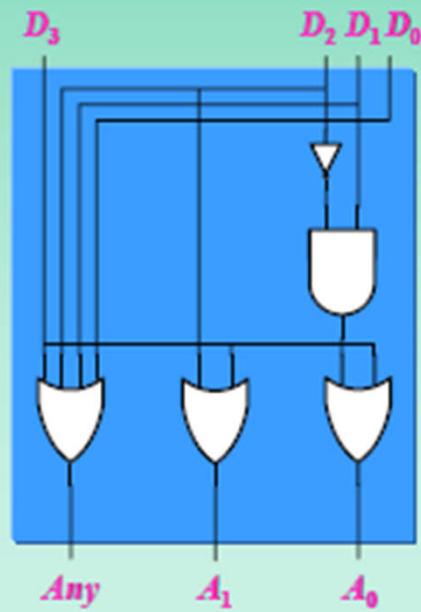


# Codifierator 4-la-2



$D_3$	$D_2$	$D_1$	$D_0$	$A_1$	$A_0$	Any
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

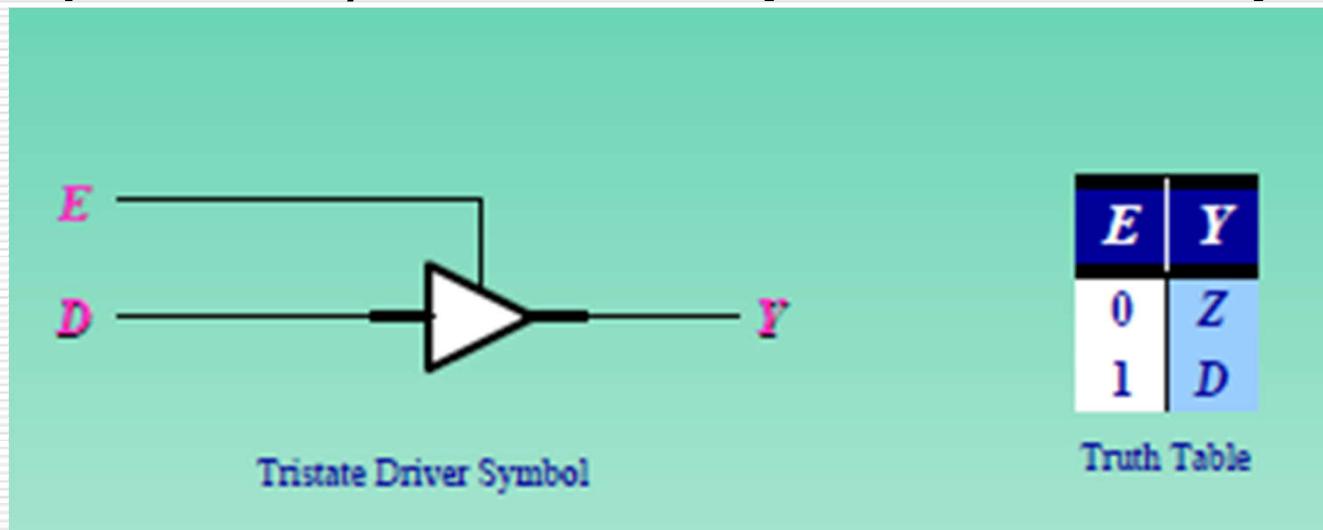
Truth Table



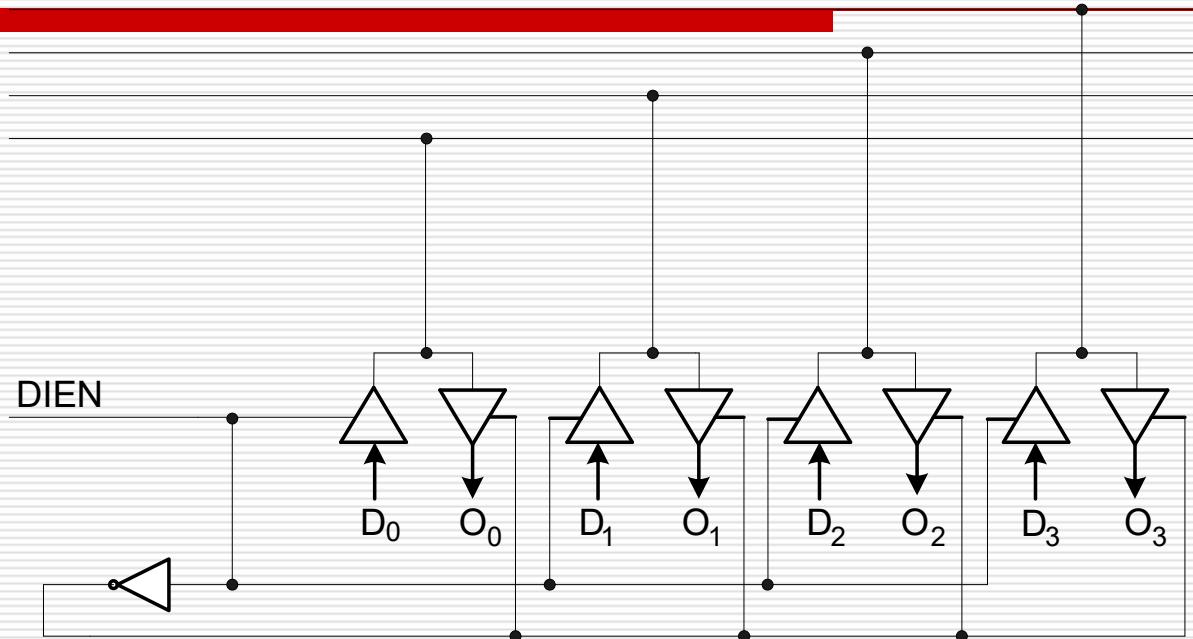
# Magistrale

---

- Poartă cu trei stări (tristate driver) are trei valori de ieșire: 0, 1, Z - impredanță ridicată( $\equiv$  disconnect)



# Magistrale: Utilizarea circuitelor poartă cu trei stări pentru conectarea la magistrala date



$D_i$  – date transmise pe magistrală

$O_i$  – date recepționate de pe magistrală

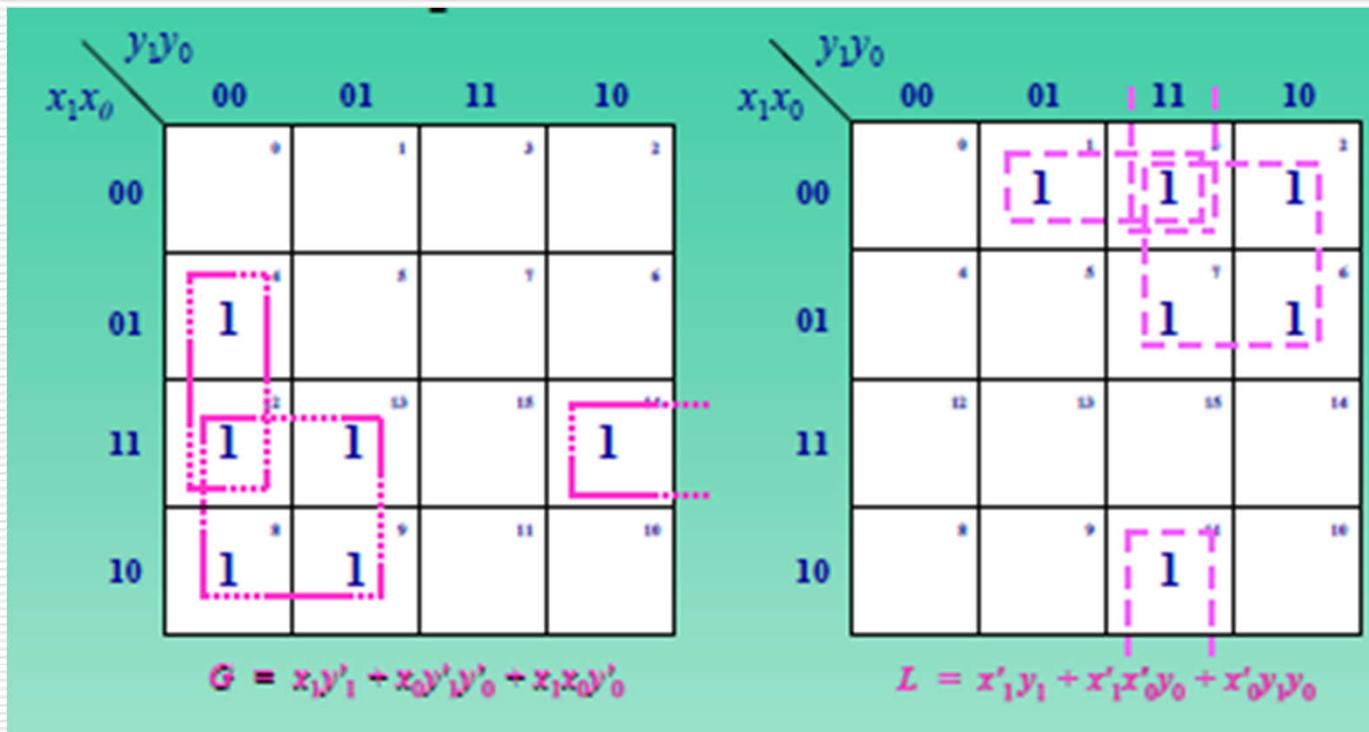
DIEN- comandă de transmisie/recepție date

# Comparatoare: numere 2 biți

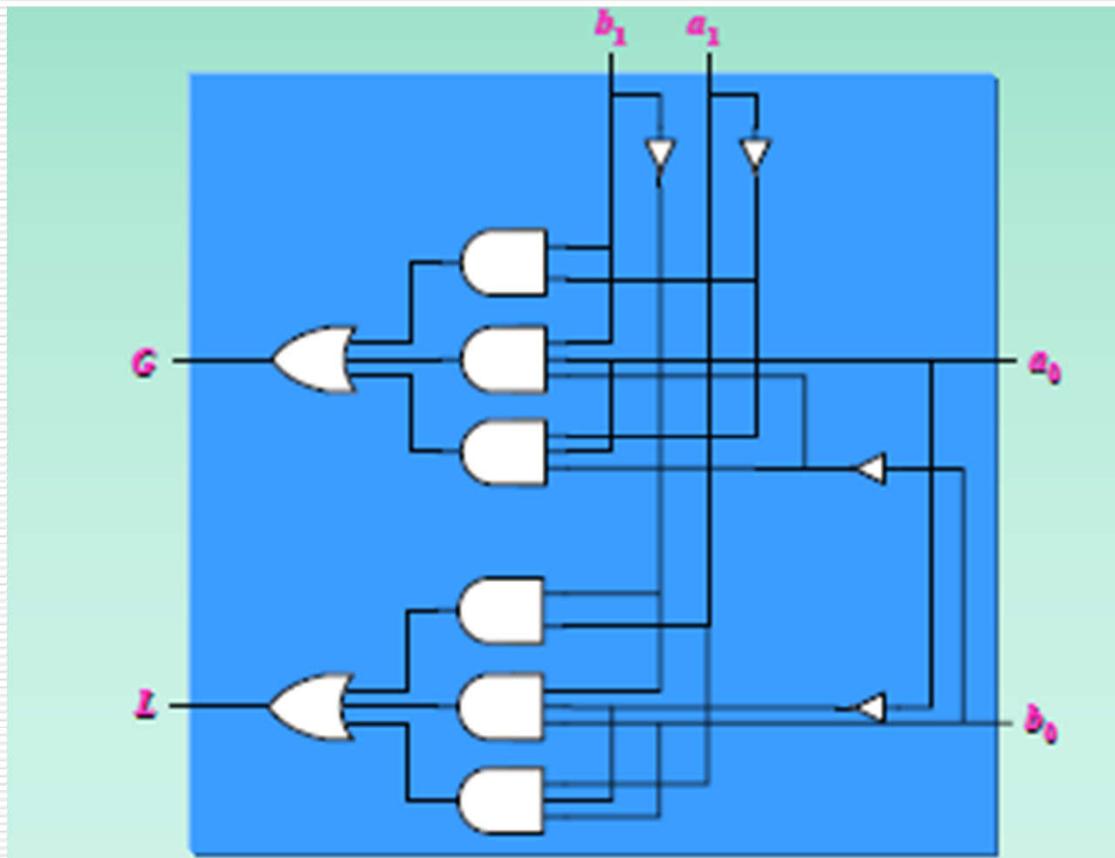
$x_1$	$x_0$	$y_1$	$y_0$	$G$	$L$
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	0
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	0

$G = 1$  when  $X > Y$ ,  
 $L = 1$  when  $X < Y$ ,  
 $G = L = 0$  when  $X = Y$

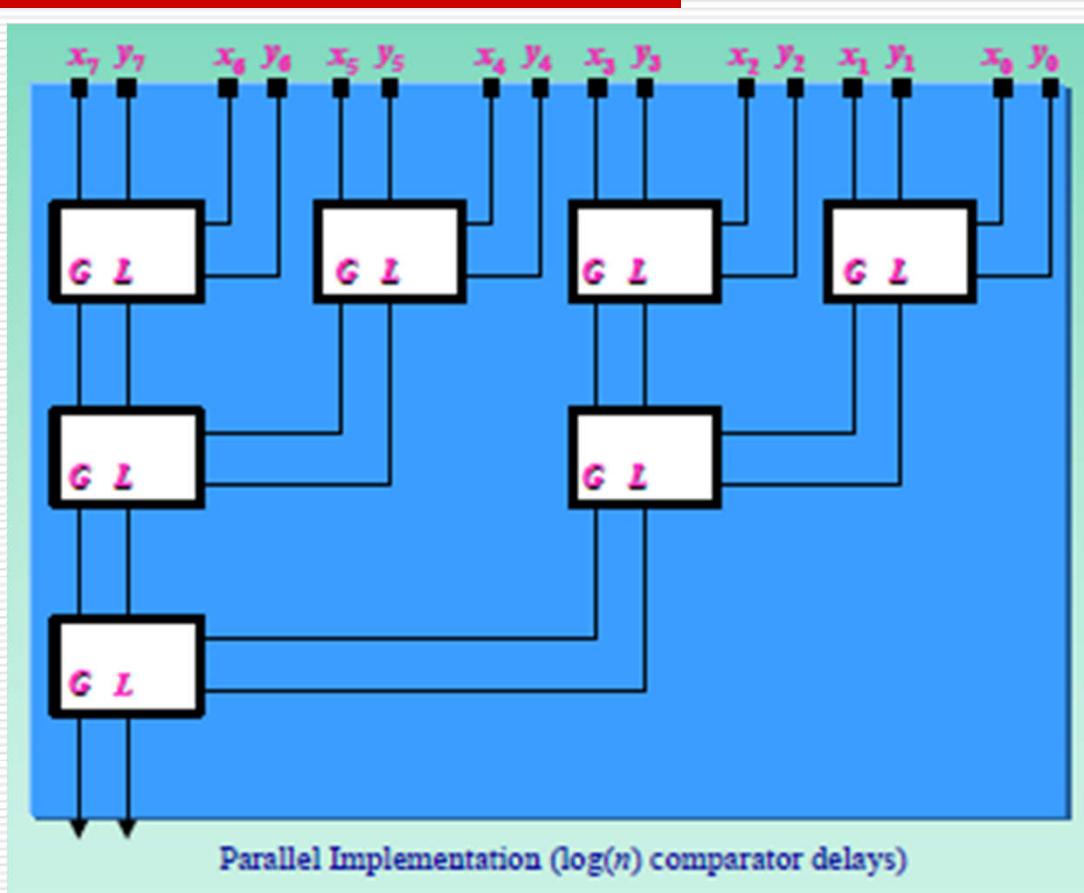
# Comparator



# Comparator



# Comparatoare pe mai mulți biți?



# Memorii ROM, PROM

---

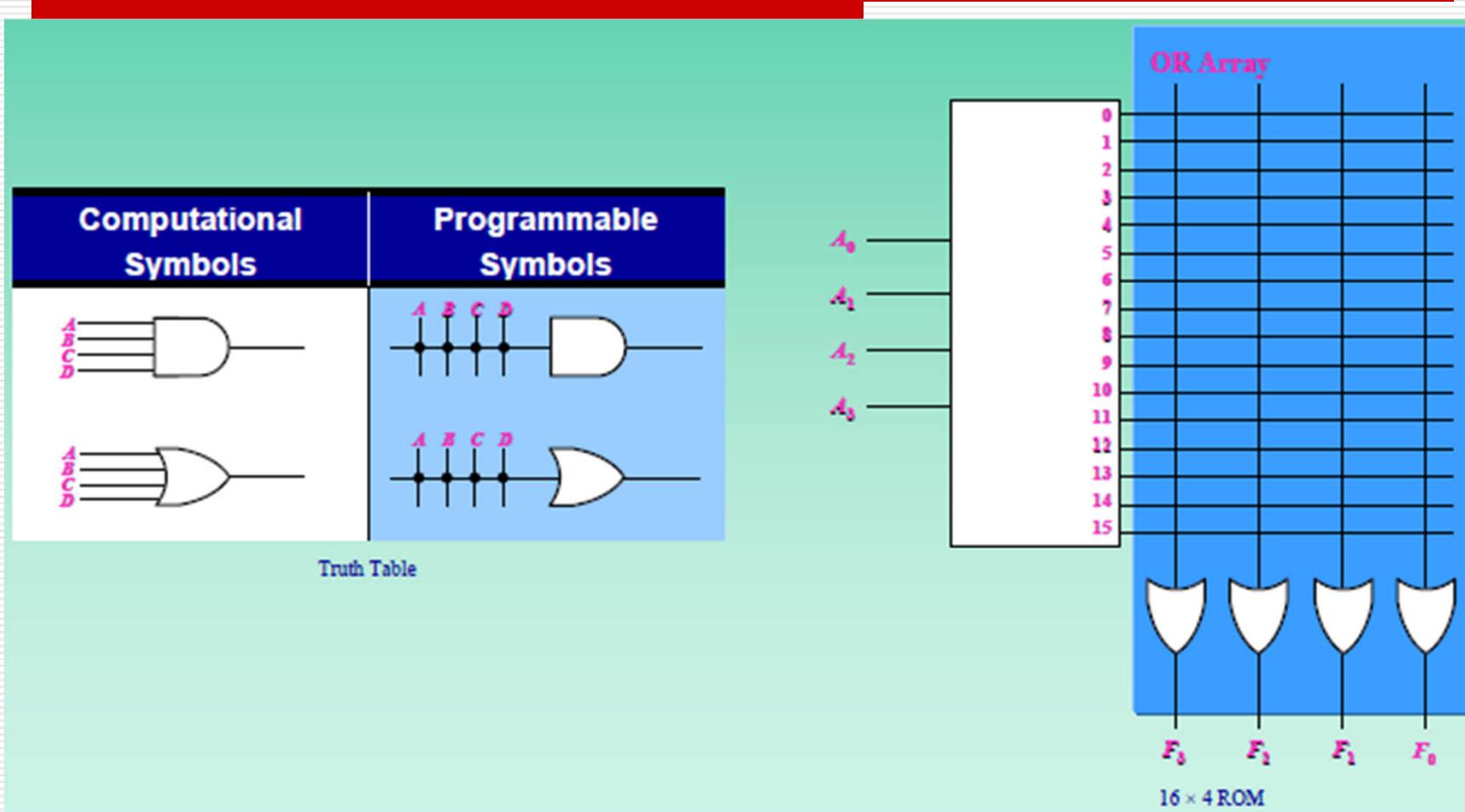
## **Memorii ROM** (read only memory)

- sunt acele memorii utilizate numai pentru citirea informației înscrisă de producător.
- își păstrează nealterata informația înscrisă, la întreruperea alimentării circuitelor, și de aceea se numesc **memorii nevolatile**.

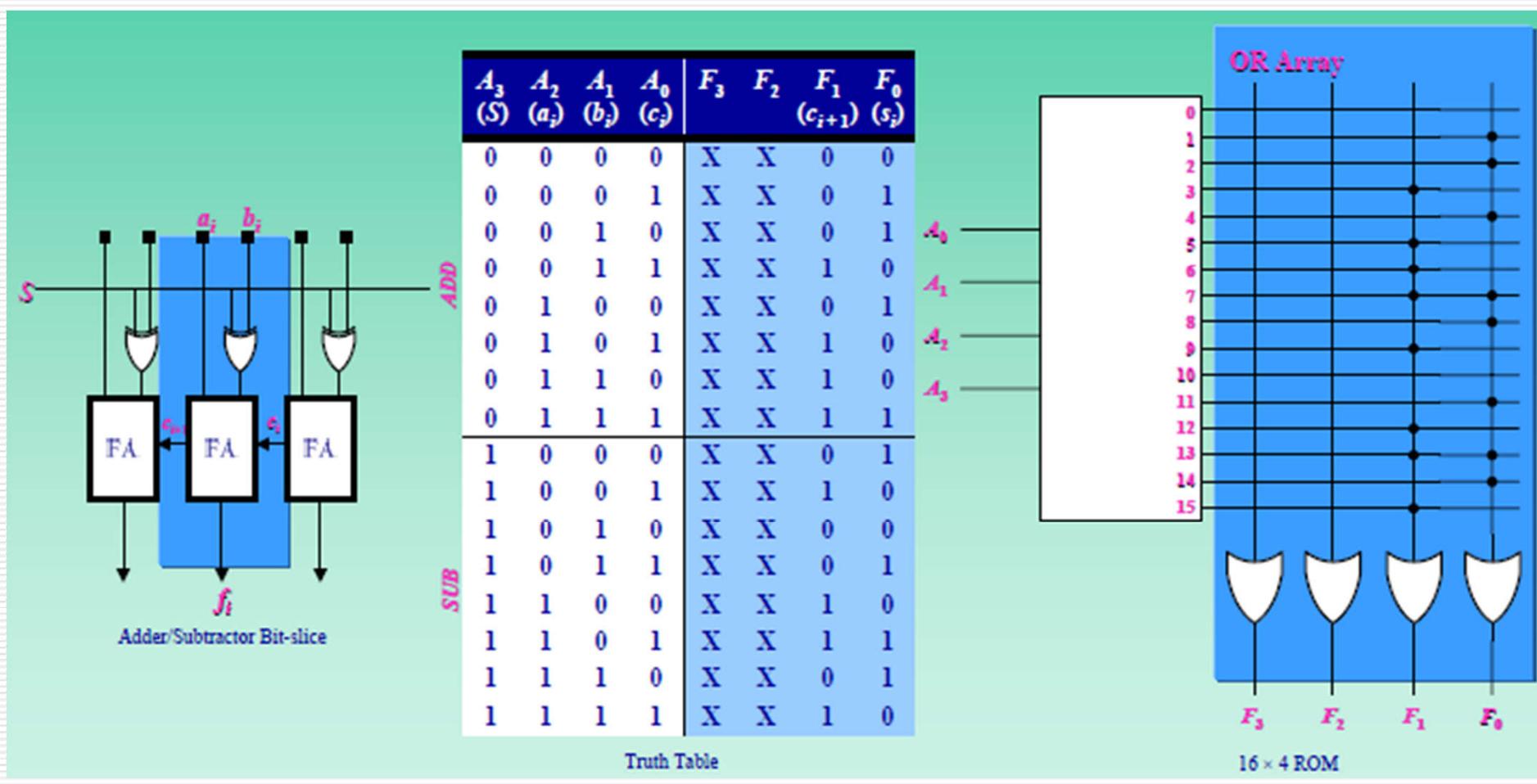
## **Memoriile PROM** (programmable read only memory)

- păstrează caracteristicile memorii ROM, cu deosebirea ca pot fi programate și reprogramate de utilizator
- Funcție de tipul lor pot fi programate cu radiații ultraviolete (EPROM) sau electric, prin aplicarea unor impulsuri de tensiune.

# Memorii ROM



# Sumator/scăzător pe bit folosind 16x4 ROM



## Implementarea funcțiilor folosind PLA-uri

---

- Implementarea folosind ROM-uri nu este eficientă în situația în care funcția are puțini mintermi.
- PLA-ul urmărește același principiu ca și ROM-ul cu obs.:
  - Are un număr de linii decodificate mai mic decât  $2^n$  ( $m$ ) → se pretează pentru situația în care funcția este sumă de un număr mic de mintermi;
  - Poate fi folosită ieșirea complementată în caz contrar

# FAC implementat cu PLA

	$x_i y_i$	00	01	11	10
$c_i$	0	*	1	*	1
1	1	*	*	1	*

$$s_i = x_i \oplus y_i \oplus c_i$$

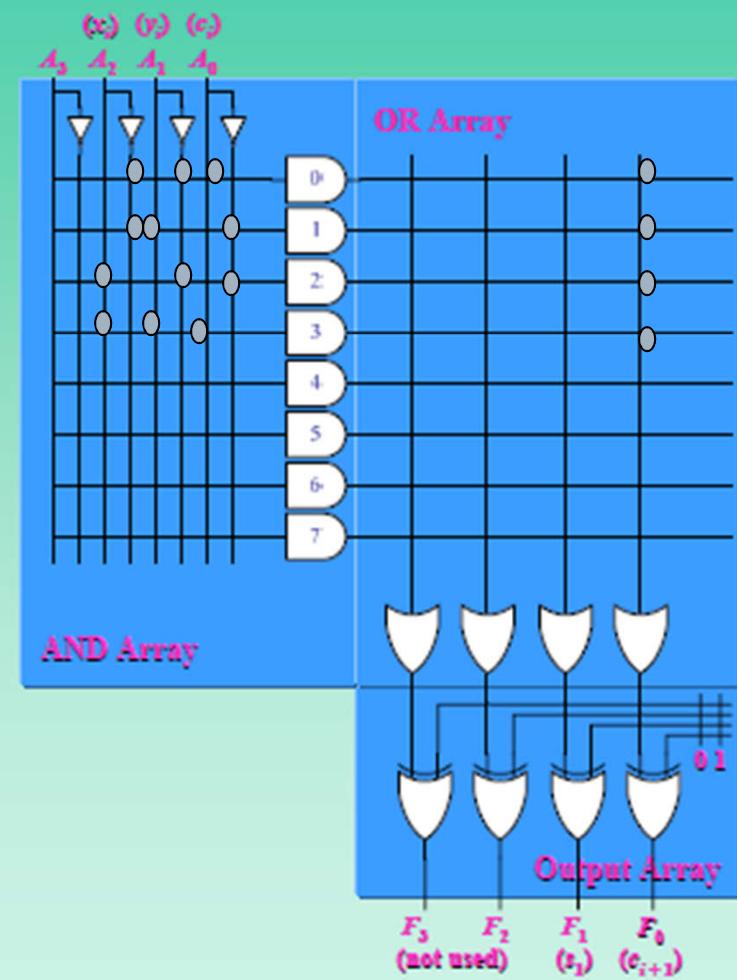
	$x_i y_i$	00	01	11	10
$c_i$	0	*	*	1	*
1	1	*	1	1	1

$$c_{i+1} = x_i y_i + c_i(x_i \oplus y_i)$$

Map Representation

$A_3$ ( $x_i$ )	$A_2$ ( $y_i$ )	$A_1$ ( $c_i$ )	$A_0$ ( $c_i$ )	$F_3$	$F_2$	$F_1$ ( $c_{i+1}$ )	$F_0$ ( $s_i$ )
X	0	0	0	X	X	0	0
X	0	0	1	X	X	0	1
X	0	1	0	X	X	0	1
X	0	1	1	X	X	1	0
X	1	0	0	X	X	0	1
X	1	0	1	X	X	1	0
X	1	1	0	X	X	1	0
X	1	1	1	X	X	1	1

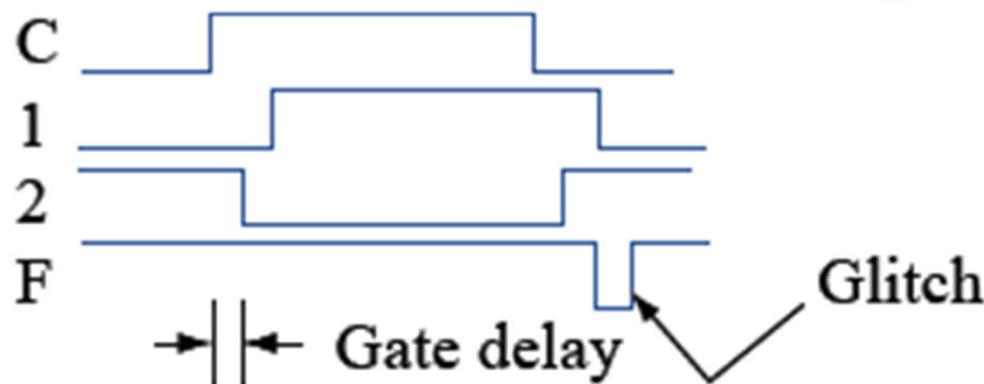
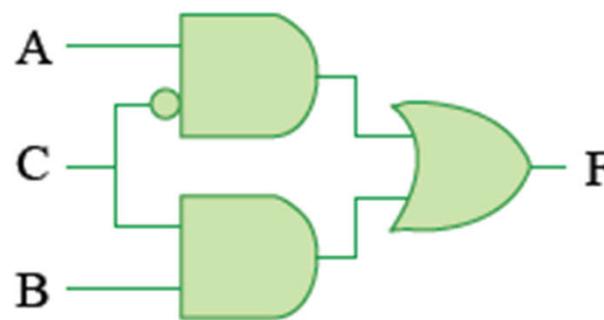
Truth Table



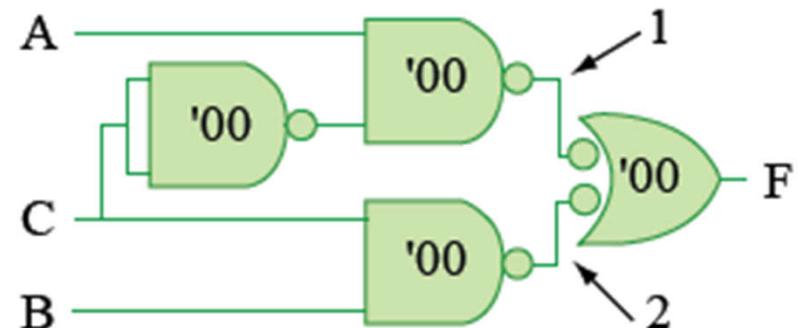
PLA Implementation

# Hazardurile la circuitele combinaționale ( $A=1$ , $B=1$ )

□ Fie funcția:  $F = AC' + BC$



Implemented with MSI gates:



AB		00	01	11	10
C	0	0	0	1	1
	1	0	1	1	0

Figure by MIT OpenCourseWare

# Hazardurile la circuitele combinaționale

---

- Glitch rezultatul diferențelor de timpi de propagare între diferite căi paralele din design.
  - Este asociat situațiilor în care funcția saltă între diferitele grupuri de termen produs din harta Karnaugh.
  - O modalitate de înlăturare a lor – acoperirea cu un nou termen produs în forma finală a funcției.
-

# Hazardurile la circuitele combinaționale

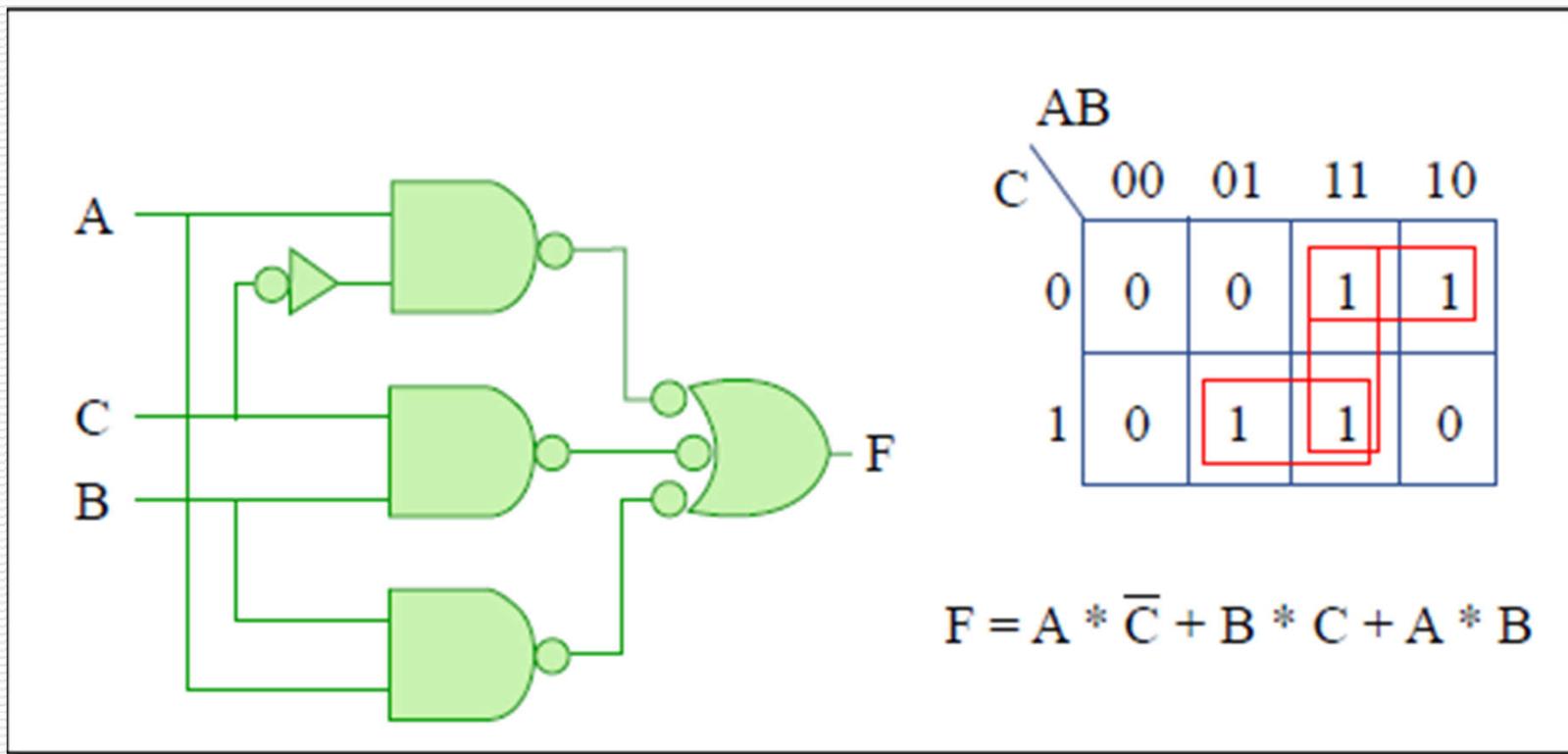


Figure by MIT OpenCourseWare.

# Întrebări?

---

**Enough Talking Let's Get To It  
!!Brace Yourselves!!**

