Universitatea Politehnica Timişoara
P-ţa Victoriei nr. 2
RO 300006 - Timişoara
Tel: +4 0256 403000
Fax: +4 0256 403021
rector@rectorat.upt.ro
www.upt.ro

Logică digitală

-Curs 6-

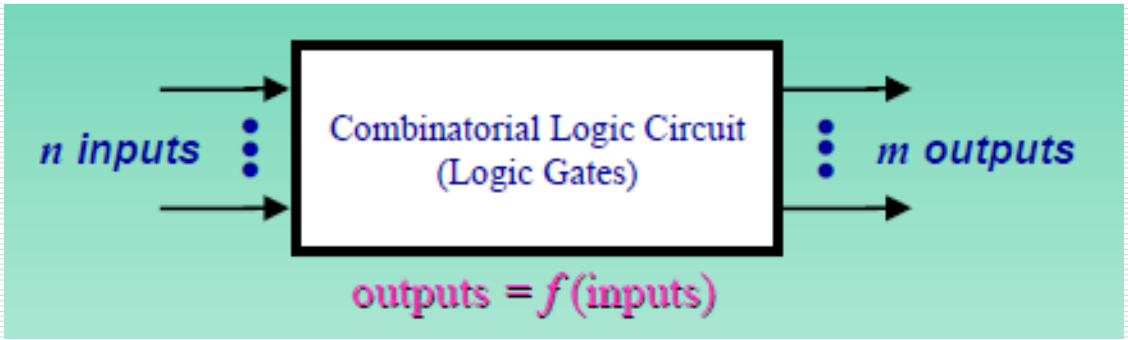
Circuite logice combinaționale

Circuite logice combinaționale

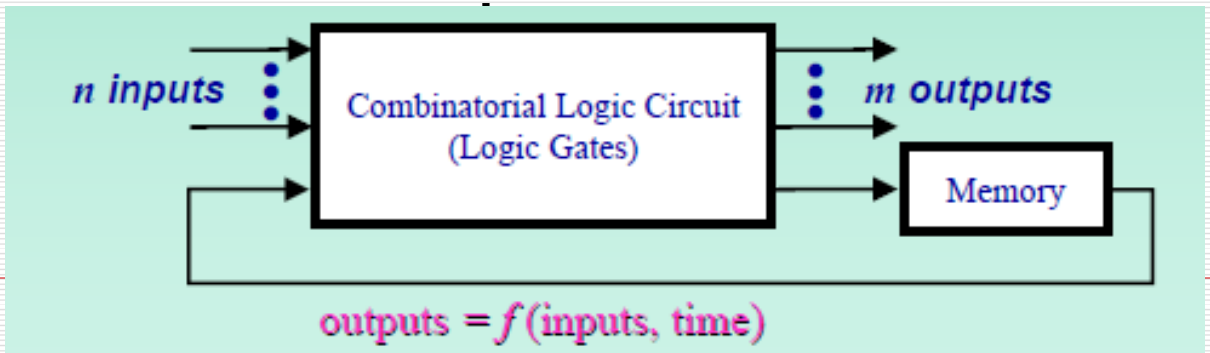
- Circuite de procesare
- Circuite de conversie
- Circuite de interconectare
- Componente universale

Clasificare componente digitale

- Componente combinaționale
  - Ușor de analizat, partiționat, verificat



- Componente secvențiale



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
  - Selecția sursei/destinației
  - Magistrale și interfete magistrală
- Alte componente (blocuri din UC)
  - ROM
  - PLA

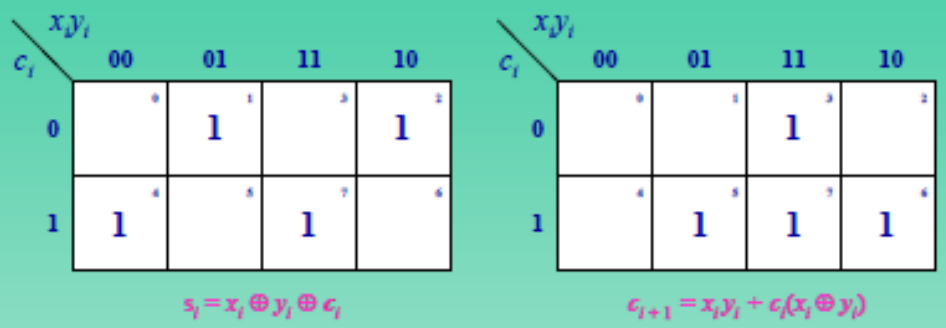
Cuvinte cheie design digital

- Încapsulare
  - Definirea unor componente/blocuri simple
- Iterare
  - Replicarea/Instanțierea componentelor în design
- Ierarhie
  - Realizarea unor blocuri mai mari din blocuri mai mici

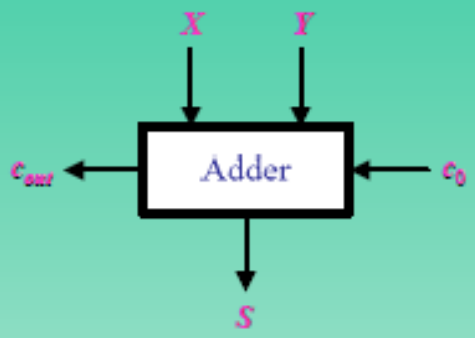
Exemplu – Sumatorul cu propagare serială a transportului

x_i	y_i	c_i	c_{i+1}	s_i
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

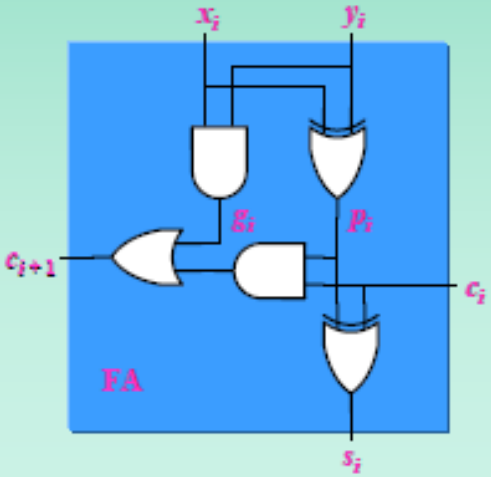
Truth Table



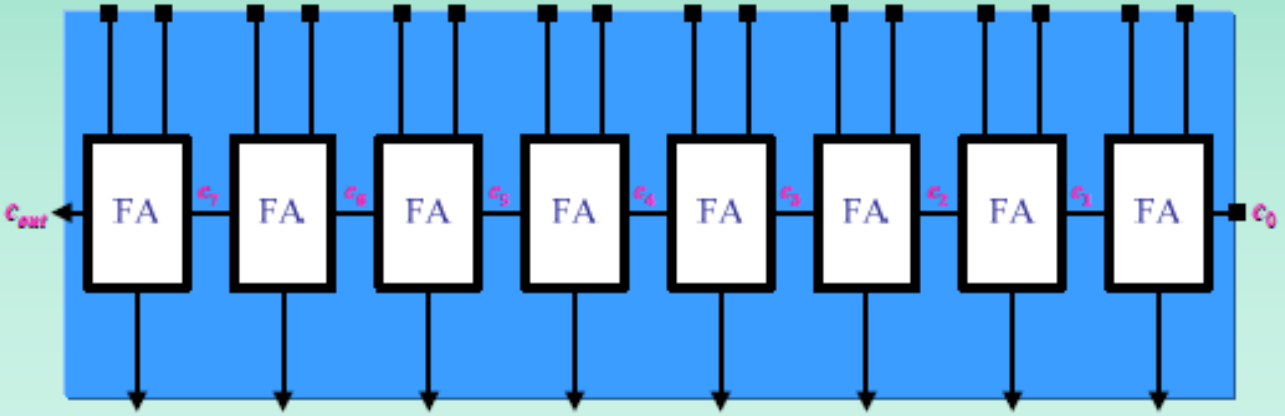
Map Representation



Graphic Symbol



Logic Schematic



8-bit Adder Unit Schematic

Slides by Philip Pham, University of California, Irvine

CLA – Sumatorul cu calculul anticipat al transportului

- Pentru creşterea performanţei sumatoarelor se încearcă calculul transportului în mod anticipat

- transportul de pe un rang să nu depindă de transportul de pe rangul anterior

CLA

$$c_1 = a_0b_0 + c_0(a_0 + b_0) = g_0 + c_0p_0$$

$$c_2 = a_1b_1 + c_1(a_1 + b_1) = g_1 + c_1p_1 = g_1 + (g_0 + c_0p_0)p_1 =$$

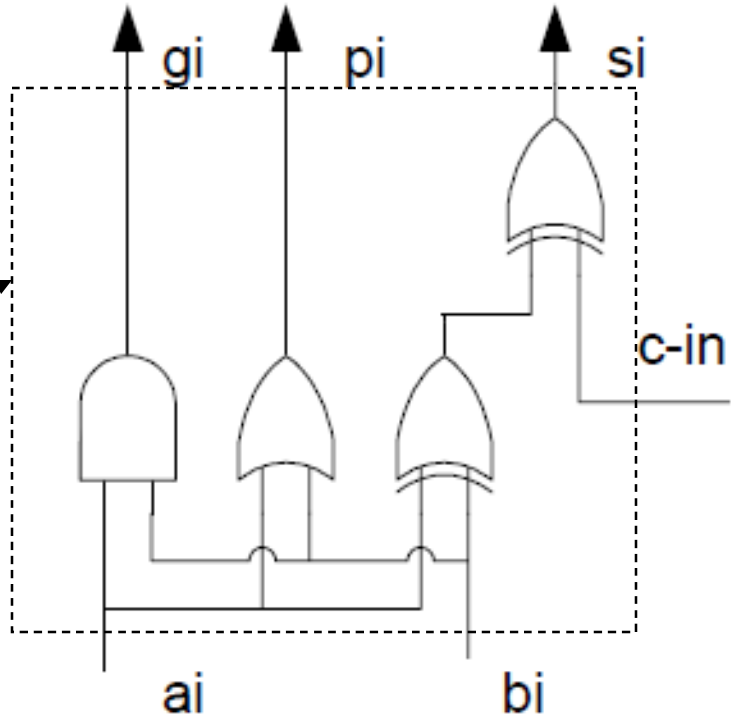
$$= g_1 + g_0p_1 + c_0p_0p_1$$

$$c_3 = g_2 + g_1p_2 + g_0p_1p_2 + c_0p_0p_1p_2$$

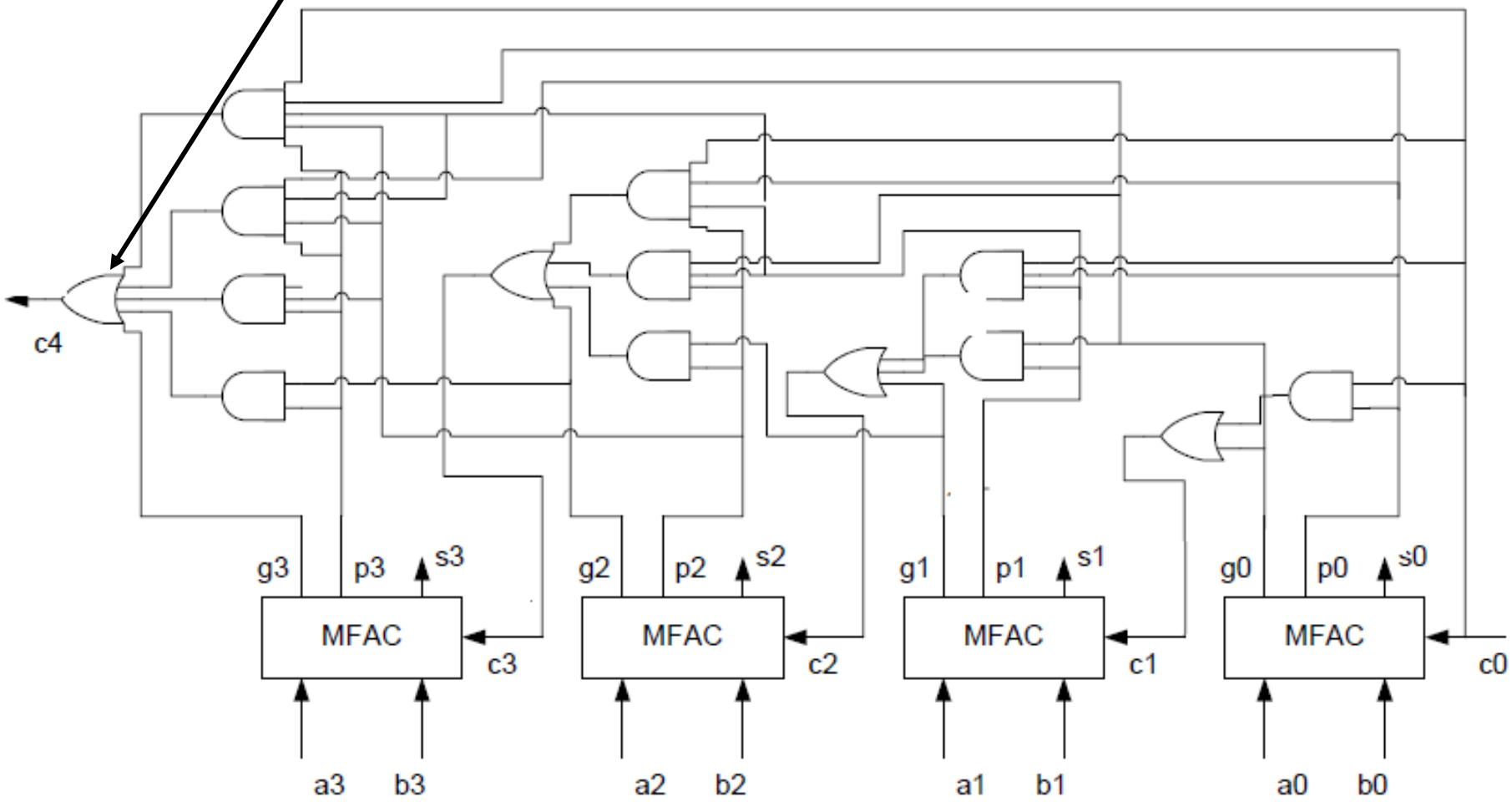
$$c_4 = g_3 + g_2p_3 + g_1p_2p_3 + g_0p_1p_2p_3 + c_0p_0p_1p_2p_3$$

Unde $g_i = a_ib_i$ şi $p_i = a_i + b_i$

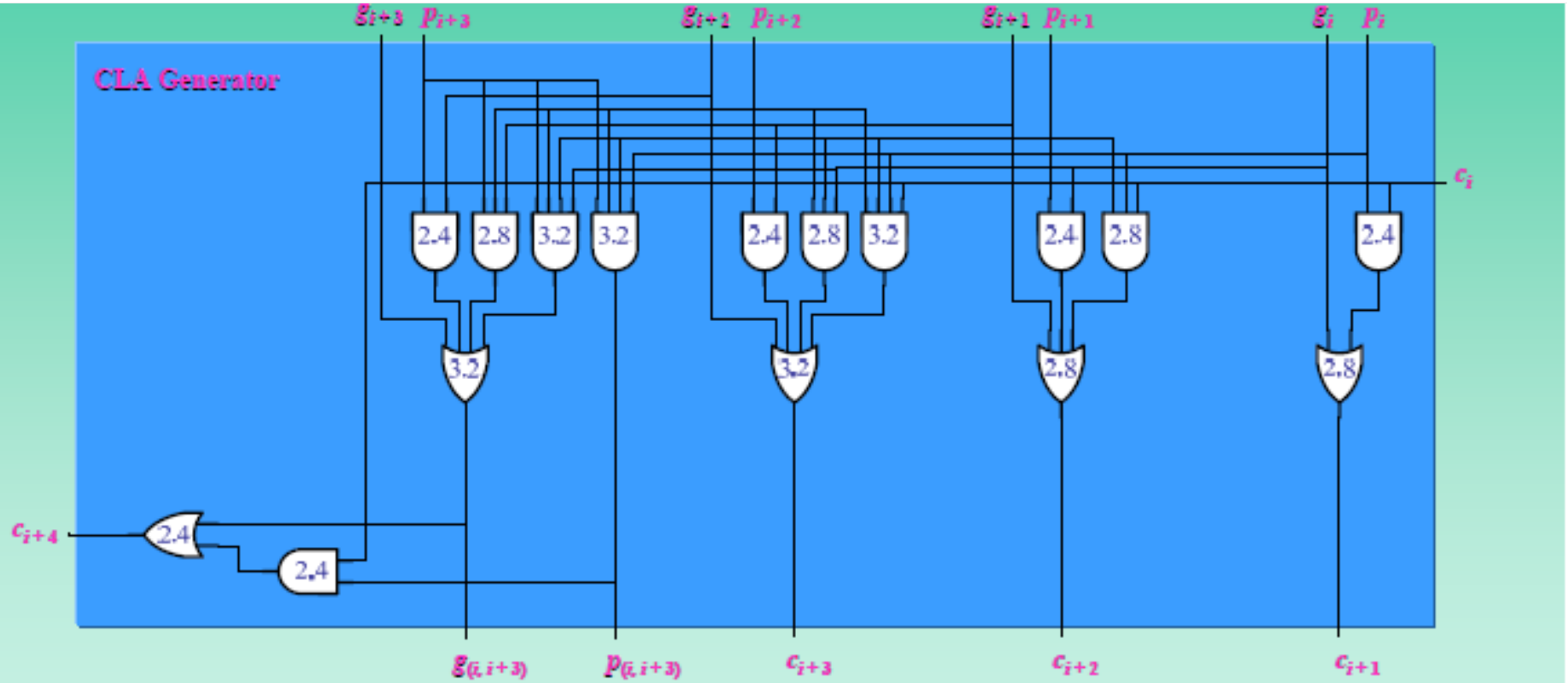
Celula de însumare modificata



$$c_4 = g_3 + g_2p_3 + g_1p_2p_3 + g_0p_1p_2p_3 + c_0p_0p_1p_2p_3$$



CLA Generator



Logic Schematic of CLA

Slides by Philip Pham, University of California, Irvine

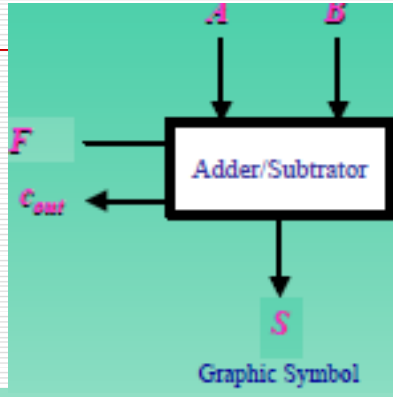
Observații:

- numărul de intrări a anumitor porţi creşte foarte mult, consecințe:
  - întârzierea porţilor crește cu număr de intrări;
  - bibliotecile de porți pot sa nu aibe porți cu atâtea intrări
- Fanout (numărul de porţi comandate de ieşirea unei porţi) creşte considerabil pentru anumite porţi
- Calculul transporturile se face în paralel

Unitate sumator/scăzator C_2

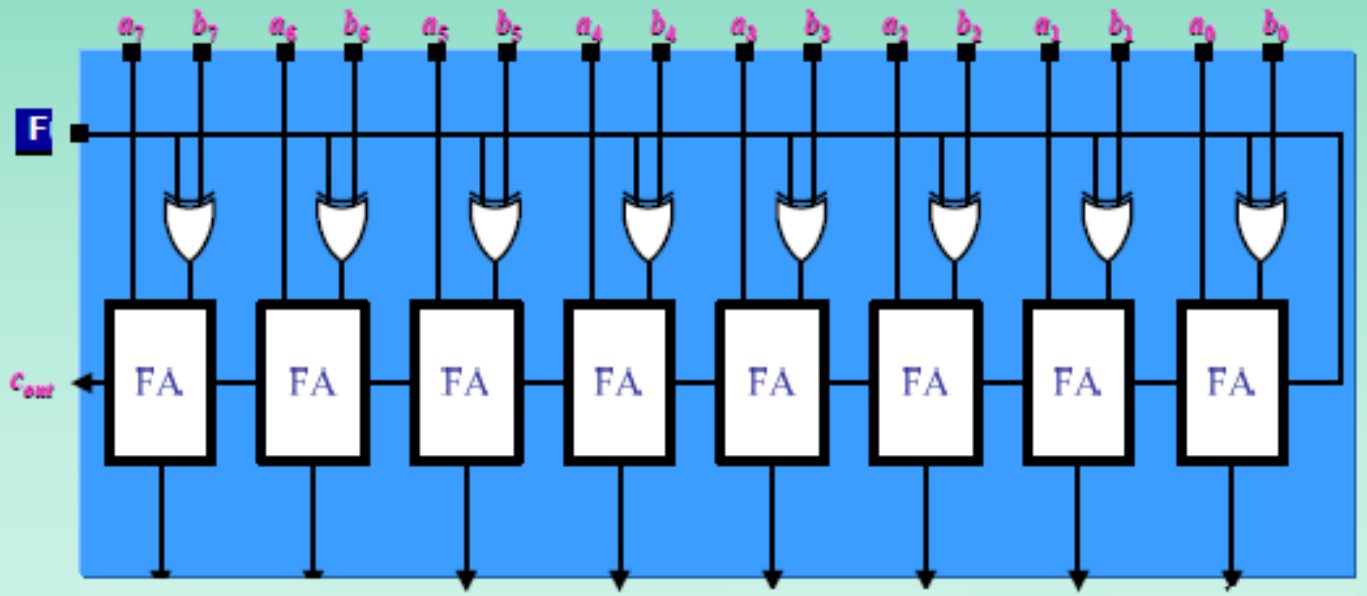
- Scădere

$$A - B = A + B' + 1$$



F	Function	Comment
0	$A + B$	Addition
1	$A + B' + 1$	Subtraction

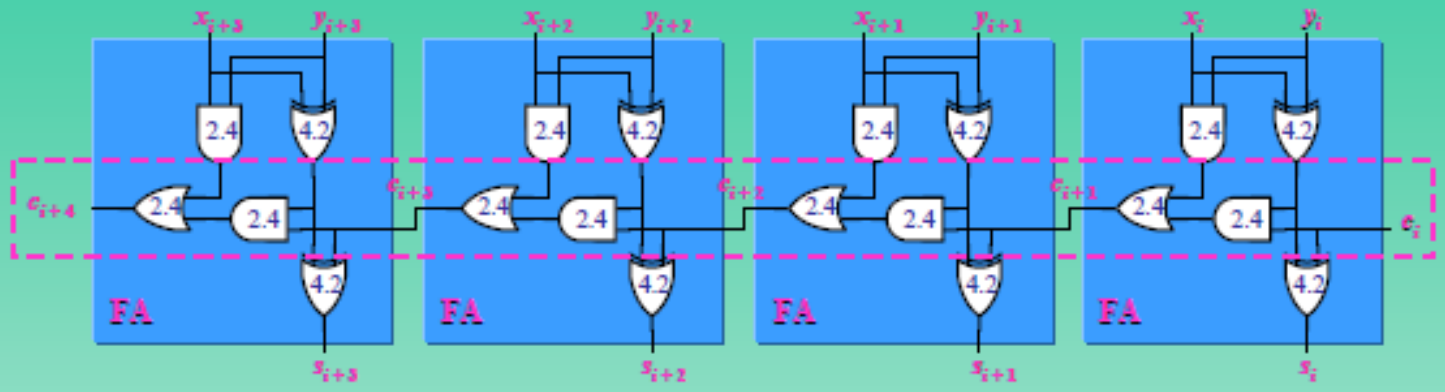
Truth Table



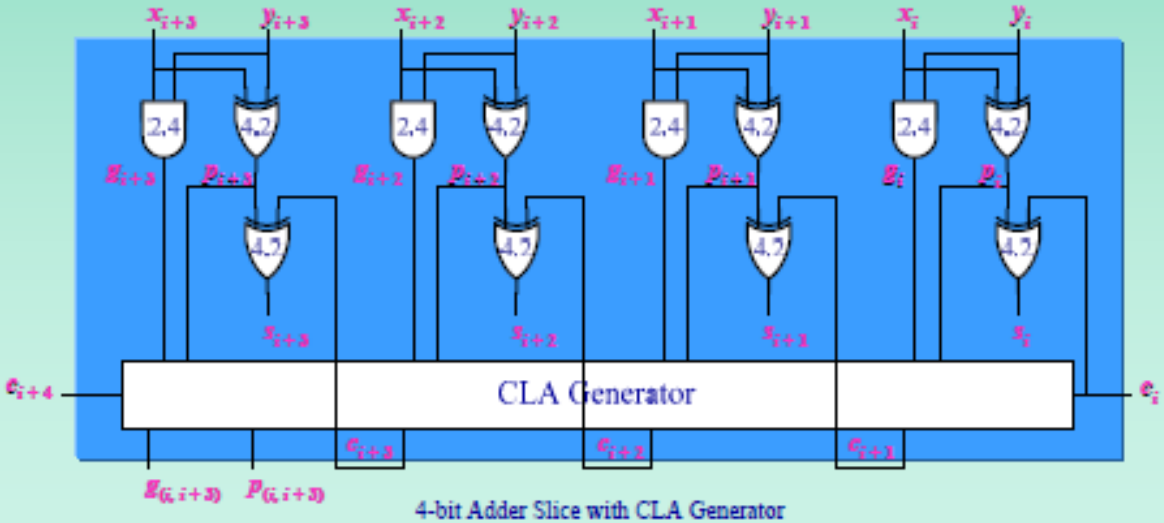
8-bit Adder/Subtractor Unit Schematic

Slides by Philip Pham, University of California, Irvine

Întârzieri pe linia de transport



4-bit Ripple–Carry Adder Slice



4-bit Adder Slice with CLA Generator

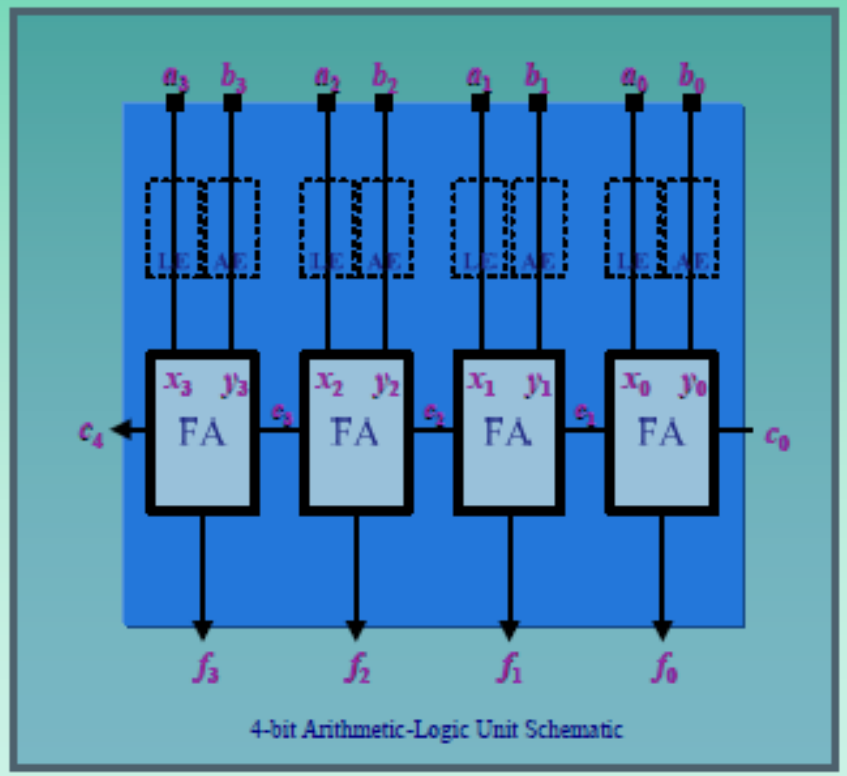
Carry Chains	Delay	
	Ripple	CLA
from $c_i(x_i, y_i)$ to c_{i+1}	4.8 (9.0)	4.8 (9.0)
from $c_i(x_i, y_i)$ to c_{i+2}	9.6 (13.8)	5.6 (9.8)
from $c_i(x_i, y_i)$ to c_{i+3}	14.4 (18.6)	6.4 (10.6)
from $c_i(x_i, y_i)$ to c_{i+4}	19.2 (23.4)	4.8 (13.0)
from $c_i(x_i, y_i)$ to $g_{(i,i+3)}$	N/A	6.4 (10.6)
from $c_i(x_i, y_i)$ to $p_{(i,i+3)}$	N/A	3.2 (7.4)

Ripple and CLA Delays for 4-bit Adder Slice

Unitate Aritmetico-Logică (ALU)

- Realizează operațiile operațiile aritmetice și logice elementare:
  - Aritmetice: adunare, scădere, incrementare, decrementare
  - Logice:ȘI, SAU, Identitate, Negare
- Toate operațiile aritmetice se bazează pe sumator → blocul de bază este sumatorul
- Trebuie configurați corespunzător operanzii → bloc dedicat de extensie op.aritmetice

ALU



4-bit Arithmetic-Logic Unit Schematic

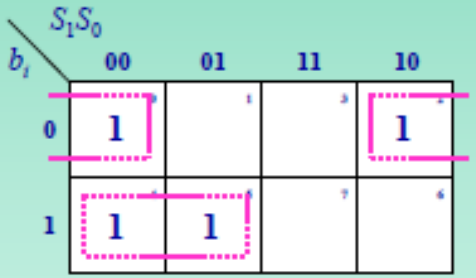
Bloc extensie aritmetică

M	S_1	S_0	Function Name	F	X	Y	c_0
1	0	0	Decrement	$A-1$	A	all 1's	0
1	0	1	Add	$A+B$	A	B'	0
1	1	0	Subtract	$A-B$	A	B'	1
1	1	1	Increment	$A+1$	A	all 0's	1

Functional Table

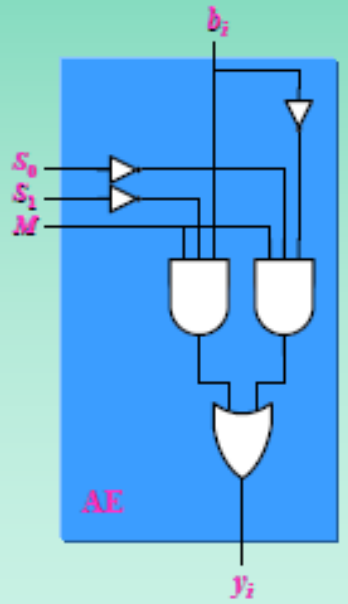
M	S_1	S_0	b_i	y_i
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Truth Table



$y_i = M S'_1 b_i + M S'_0 b'_i$

Map Representation



Logic Schematic

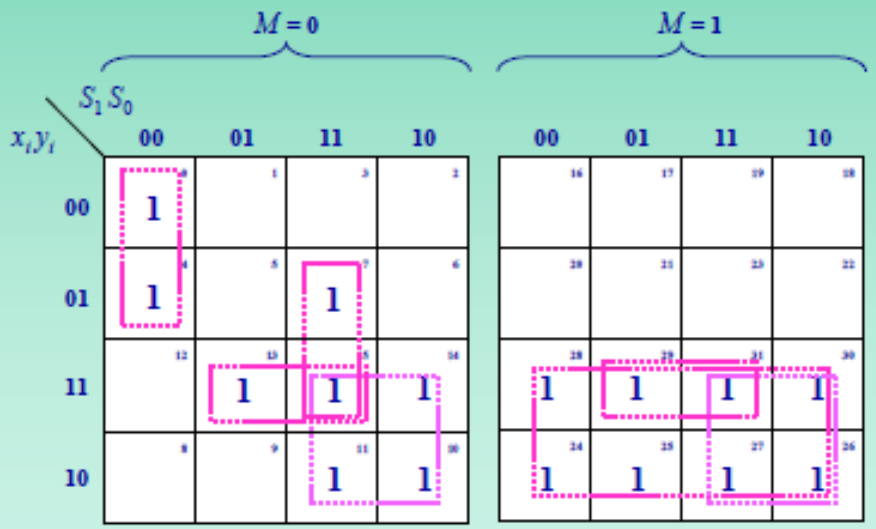
Bloc extensie operații logice

M	S_1	S_0	Function Name	F	X	Y	c_0
0	0	0	Complement	A'	A'	0	0
0	0	1	AND	A AND B	A AND B	0	0
0	1	0	Identity	A	A	0	0
0	1	1	OR	A OR B	A OR B	0	0

Functional Table

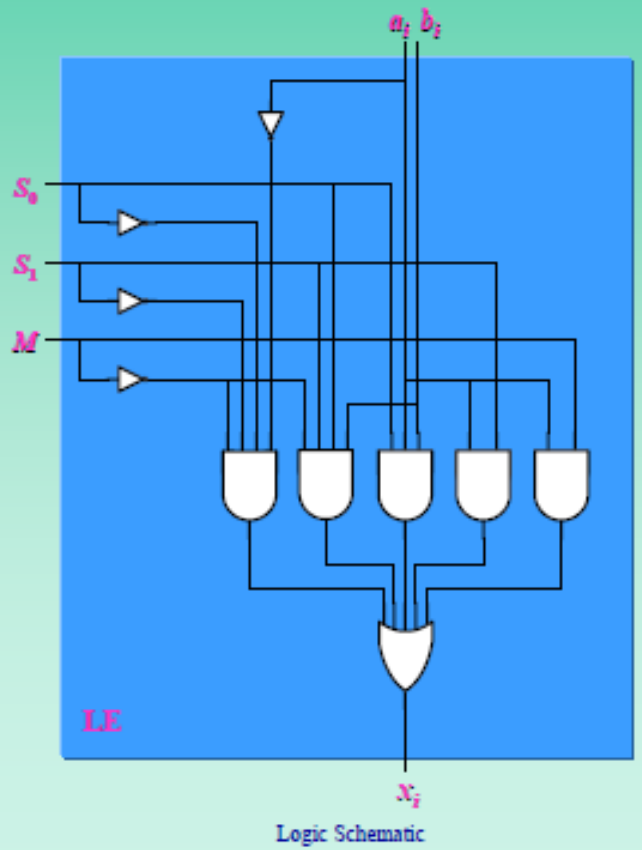
M	S_1	S_0	x_i
0	0	0	a'_i
0	0	1	$a_i b_i$
0	1	0	a_i
0	1	1	$a_i + b_i$
1	X	X	a_i

Truth Table



$$x_i = M' S'_1 S'_0 a'_i + M' S_1 S_0 b_i + S_0 a_i b_i + S_1 a_i + M a_i$$

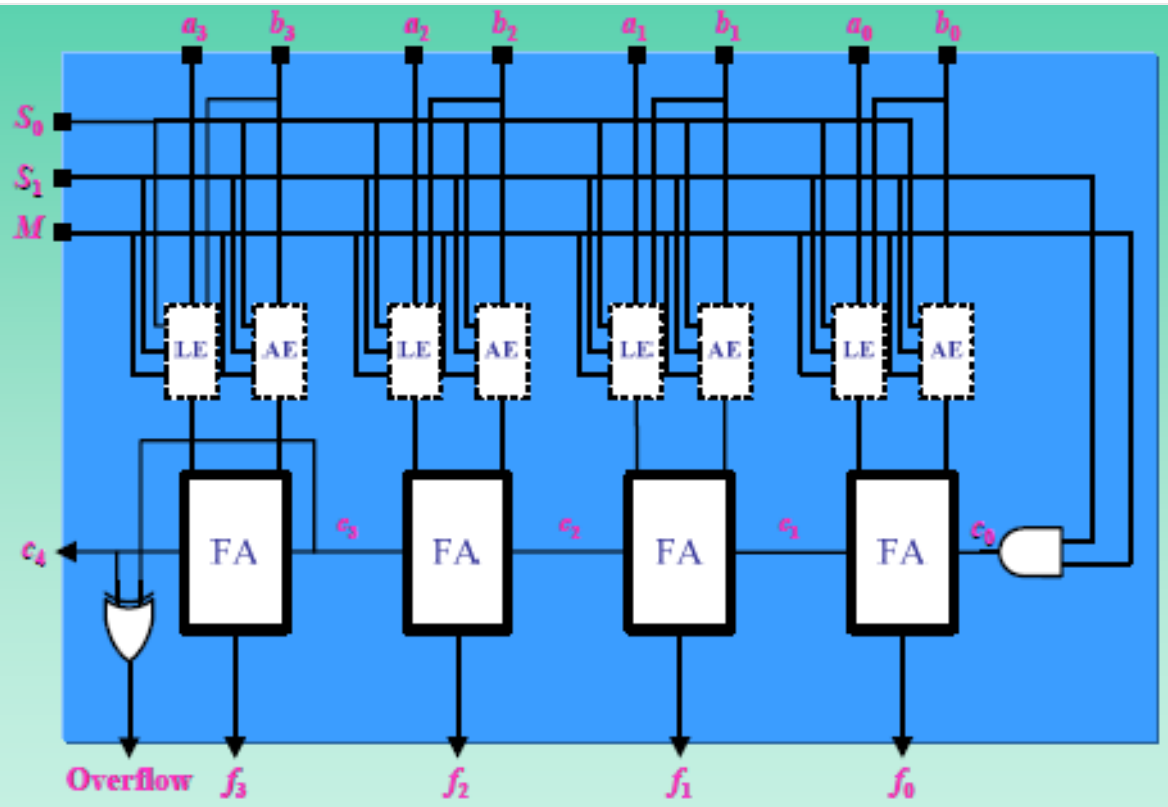
Map Representation



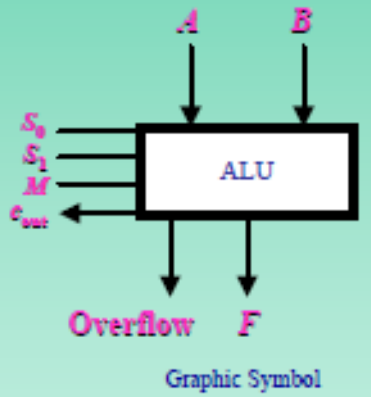
Logic Schematic

Slides by Philip Pham, University of California, Irvine

Arhitectură ALU



4-bit Arithmetic Logic Unit Schematic



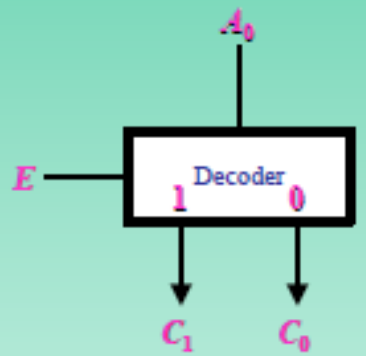
Graphic Symbol

Slides by Philip Pham, University of California, Irvine

Decodificator

- circuite logice combinaţionale ce prezintă un anumit *n* intrări şi până la 2^n ieşiri, care activează **ieşirea (UNA SINGURĂ)** corespunzătoare valoarii combinaţiei vectorului de intrare
- Pot avea intrări de activare, astfel încât ieşirea selectată nu pot fi activată decât dacă intrarile de activare sunt active.
- Pt. n intrări şi cu m ieşiri → decodificator n-la-m.
- Uzual sunt folosite pt. activarea (EN) componentelor

Decodificatorul 1-la-2



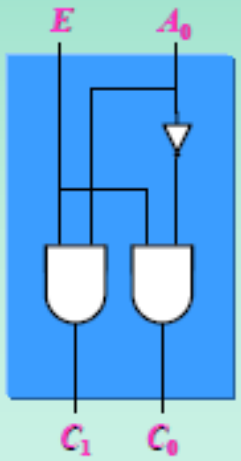
Graphic Symbol

$$C_0 = EA'_0$$
$$C_1 = EA_0$$

Boolean Expression

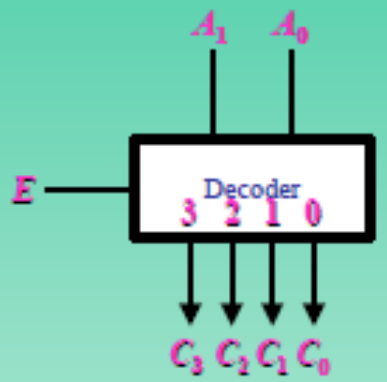
E	A_0	C_1	C_0
1	0	0	1
1	1	1	0
0	X	0	0

Truth Table



Logic Schematic

Decodificatorul 2-la-4



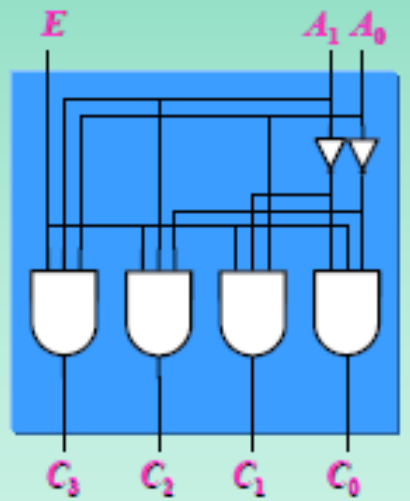
Graphic Symbol

E	A_1	A_0	C_3	C_2	C_1	C_0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0
0	X	X	0	0	0	0

Truth Table

$$C_0 = E_0 A'_1 A'_0$$
$$C_1 = E_0 A'_1 A_0$$
$$C_2 = E_0 A_1 A'_0$$
$$C_2 = E_0 A_1 A_0$$

Boolean Expression

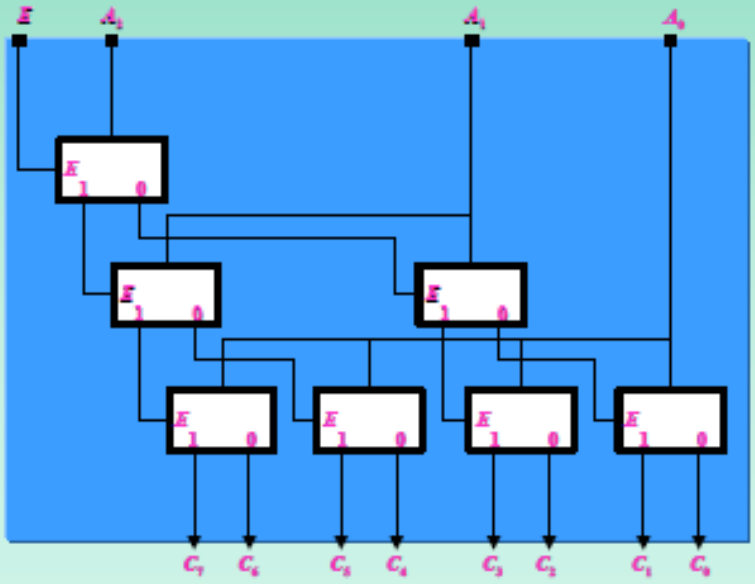


Logic Schematic

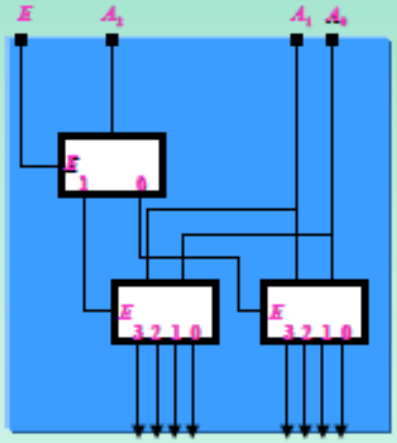
Decodificatorul 3-la-8

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

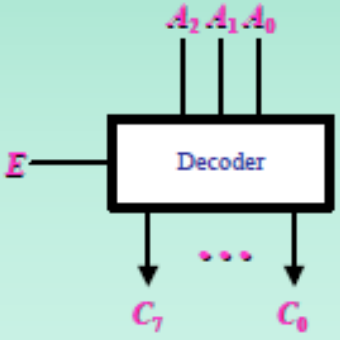
Truth Table



Implementation with 1-to-2 Decoders



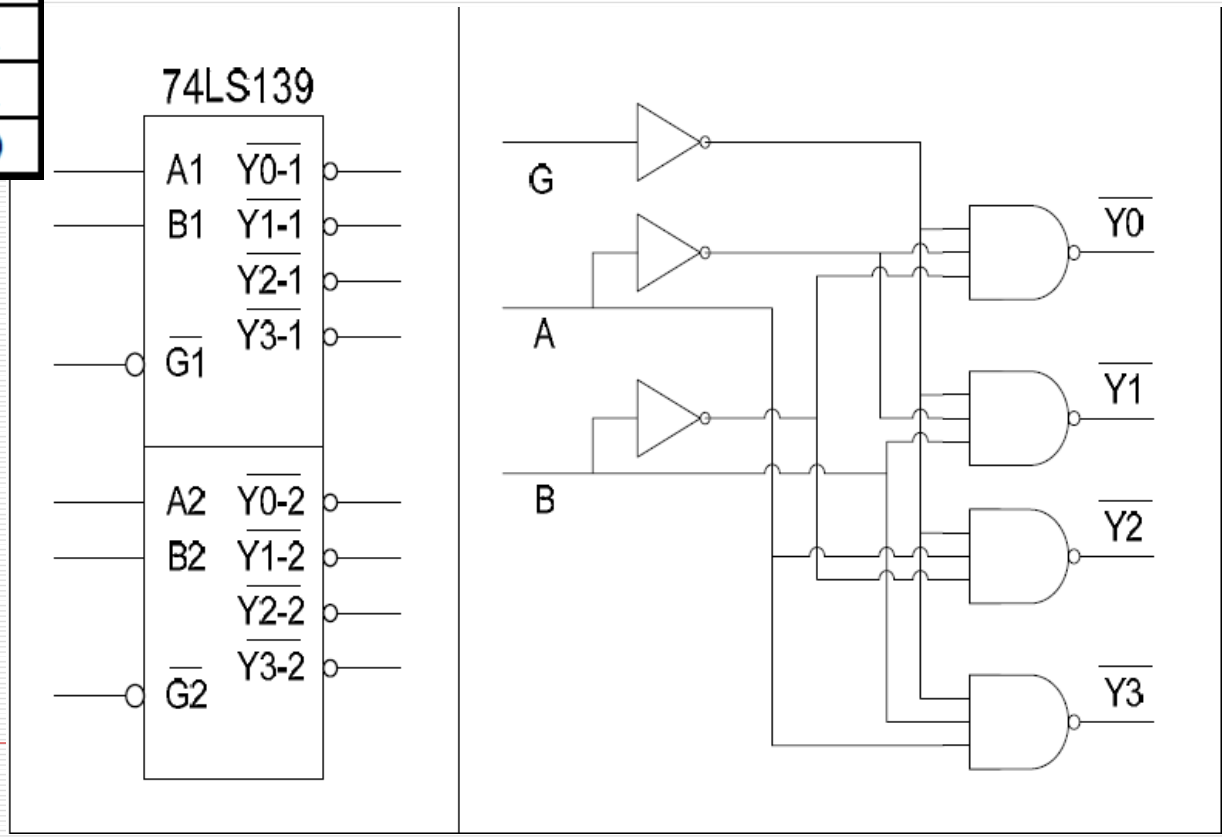
Implementation with 2-to-4 Decoders



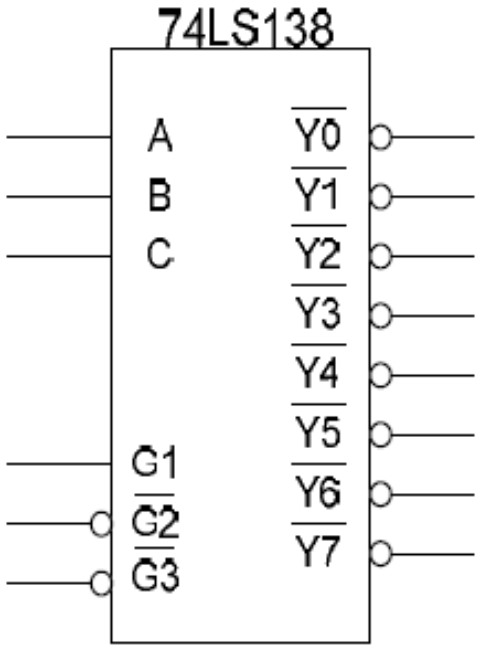
Graphic Symbol

Circuite integrate pe scară medie ce îndeplinesc funcţia 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

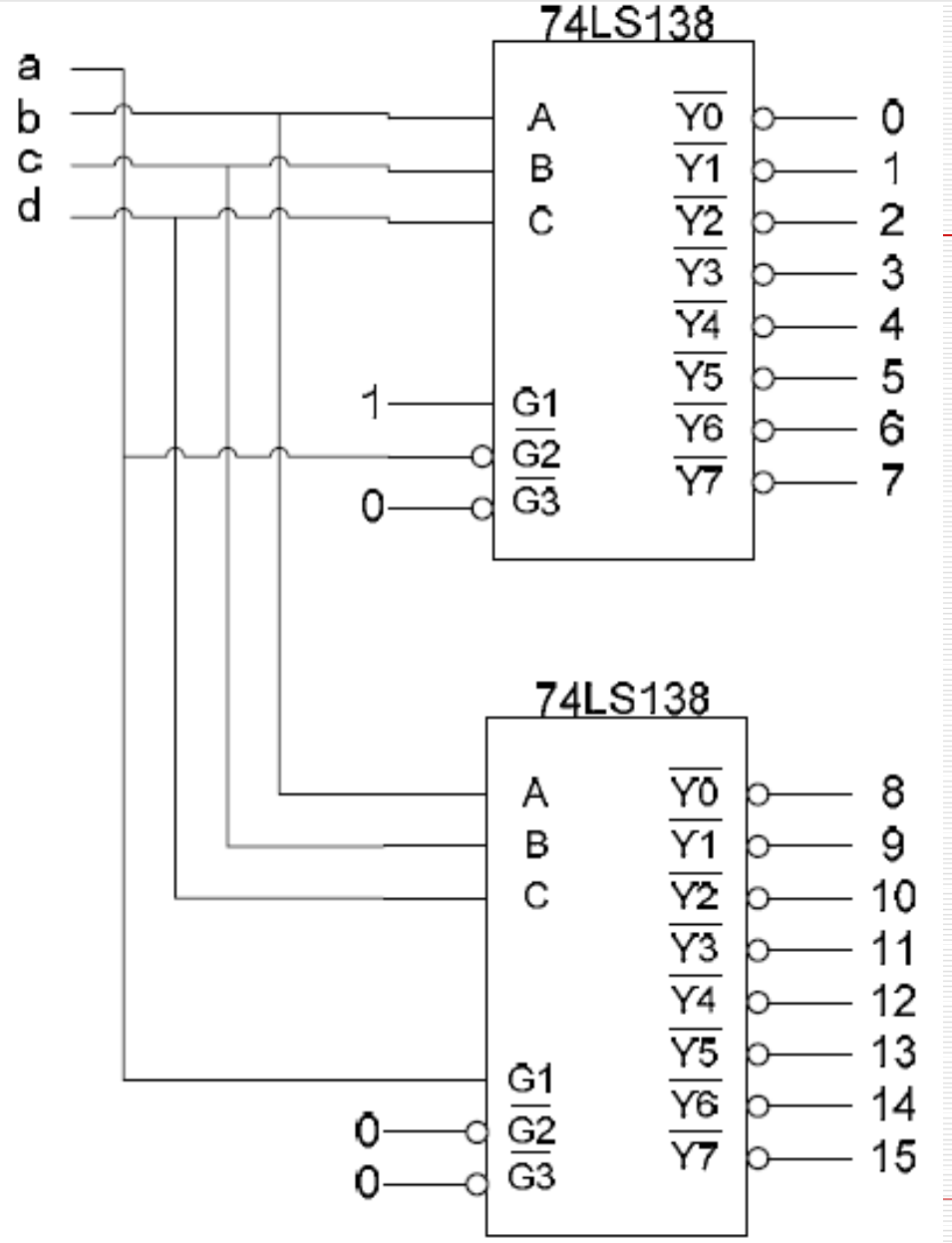


74LS138: decodificator 3-la-8

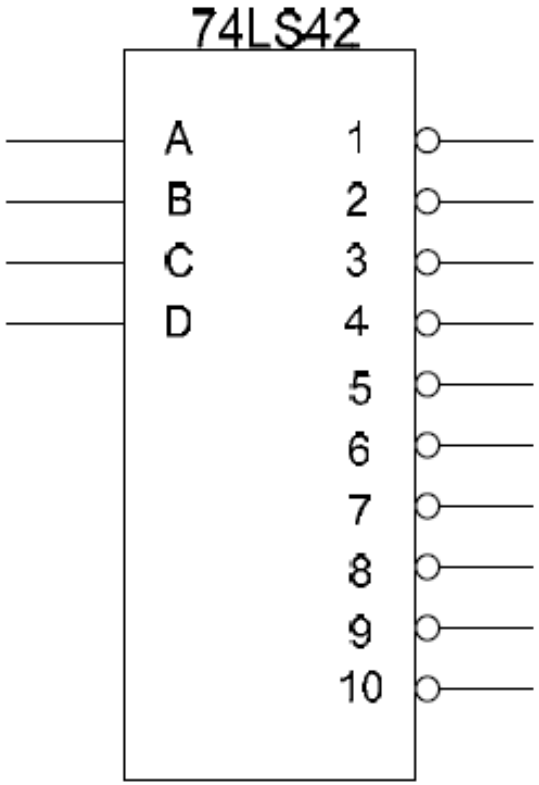


G1	G2	G3	A	B	C	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

DEC 4-la-16


74LS138
a
b
c
d
A
B
C
G1
G2
G3
1
0
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
0
1
2
3
4
5
6
7
74LS138
A
B
C
G1
G2
G3
0
0
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
8
9
10
11
12
13
14
15

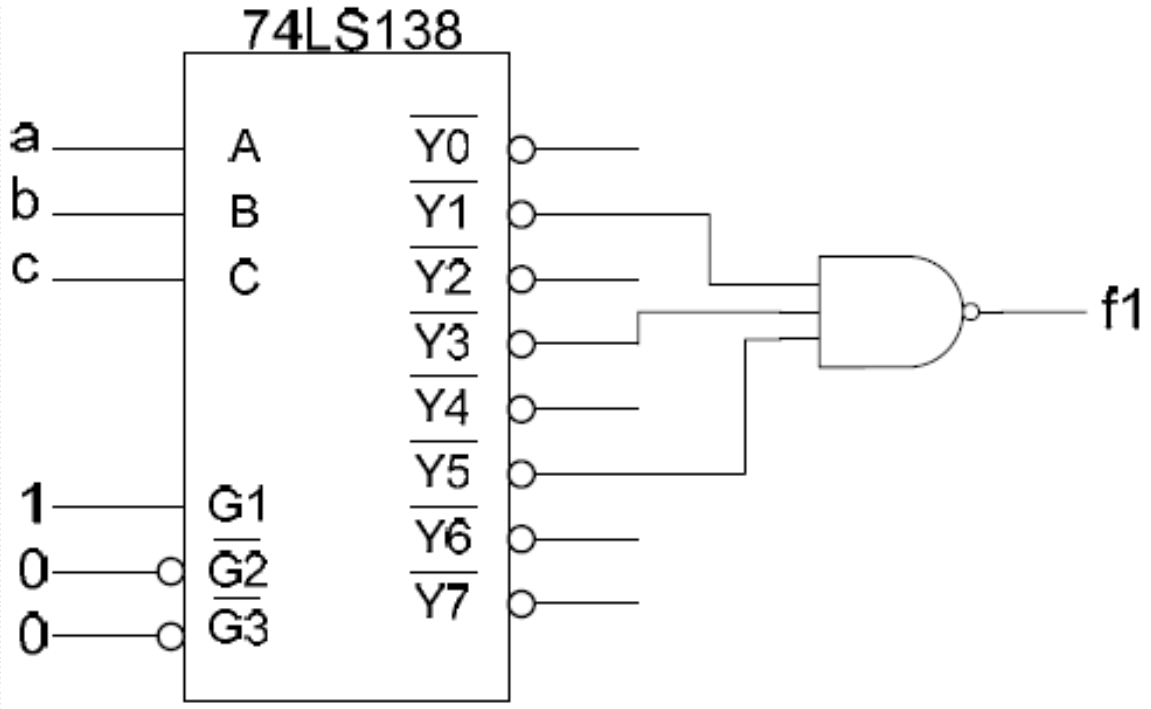
14LS42: Decodificator 4-la-10



A	B	C	B	1	2	3	4	5	6	7	8	9	10
0	0	0	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	1	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	1	1
0	1	0	0	1	1	1	0	1	1	1	1	1	1
0	1	0	1	1	1	1	1	0	1	1	1	1	1
0	1	1	0	1	1	1	1	1	0	1		1	1
0	1	1	1	1	1	1	1	1	1	0		1	1
1	0	0	0	1	1	1	1	1	1	1	0	1	1
1	0	0	1	1	1	1	1	1	1	1	1	0	1
1	0	1	0	1	1	1	1	1	1	1	1	1	0
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

Sinteza funcțiilor logice folosind decodificatoare

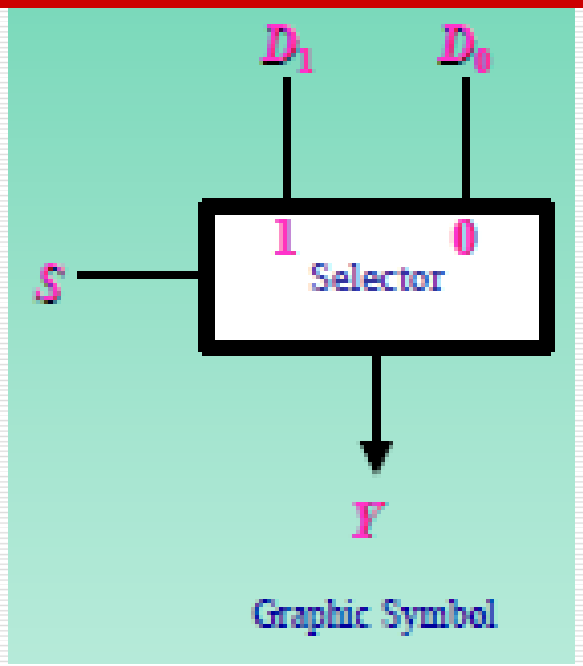
Să se implementeze cu ajutorul unui decodificator 74LS138 funcția logică $f1(a,b,c) = \sum(1,3,5)$



Multiplexor(Selector)

- Multiplexorul este un circuit logic combinaţional ce conectează ieşirea aceestuia la una din cele *n* intrări.
- Selecţia unuia din cele n intrări se face cu ajutorul a $log_2 n$ intrări de selecţie.
- Poate fi privit ca un comutator digital.
- Este folosit la pt.selecția unei singure surse de date din mai multe.

MUX 2-la-1

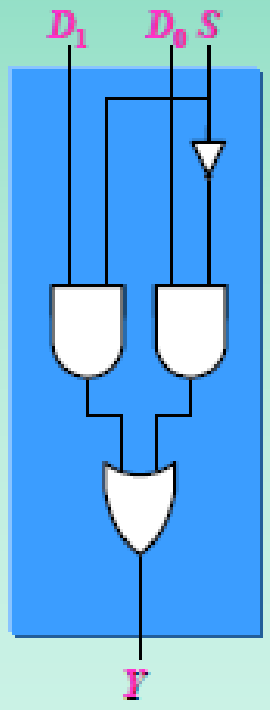


$$Y = S'D_0 + SD_1$$

Boolean Expression

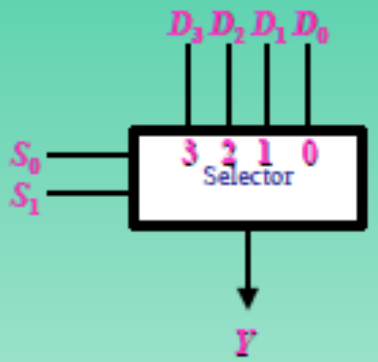
S	Y
0	D_0
1	D_1

Truth Table



Logic Schematic

MUX 4-la-1



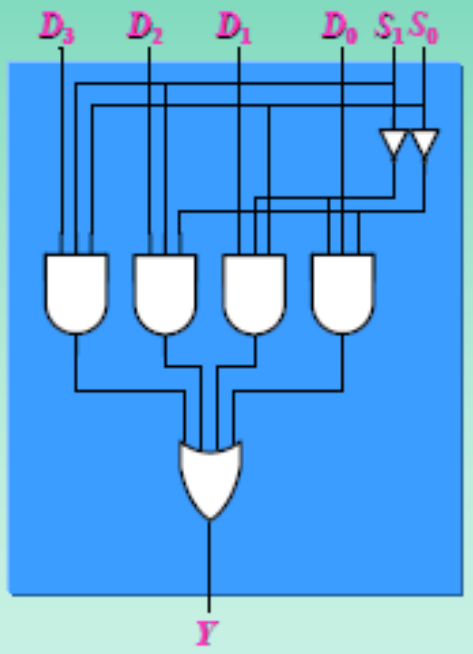
Graphic Symbol

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

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

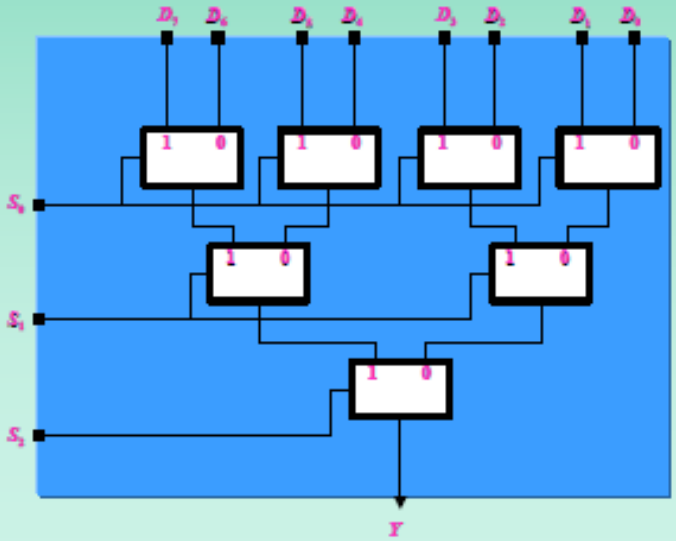


Logic Schematic

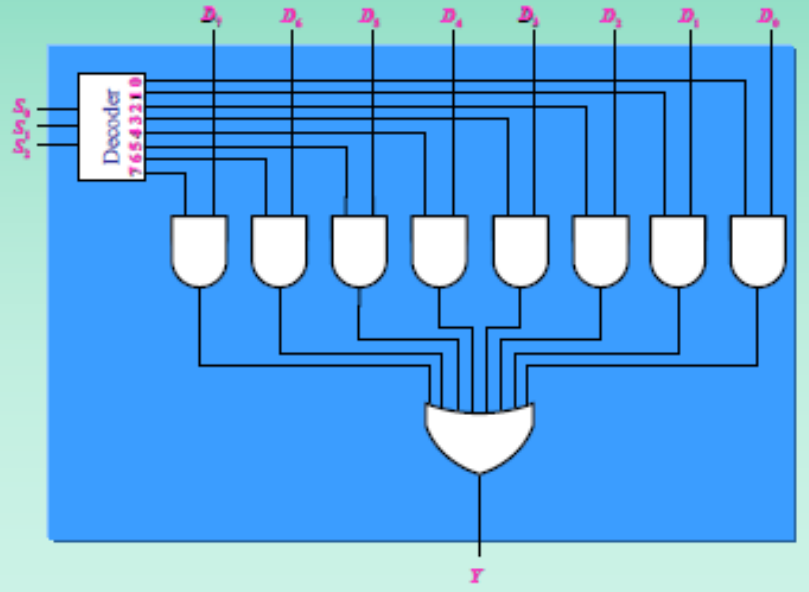
MUX 8-la-1

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



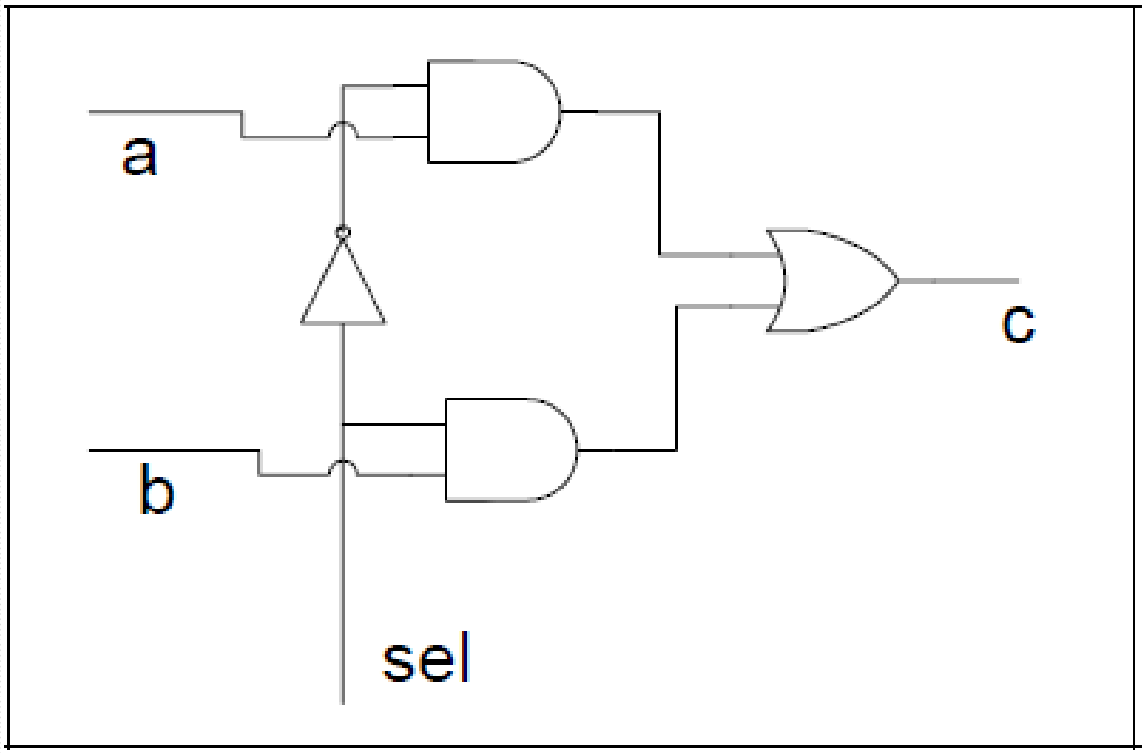
Implementation with 2-to-1 Selectors



Implementation with 3-to-8 Decoder

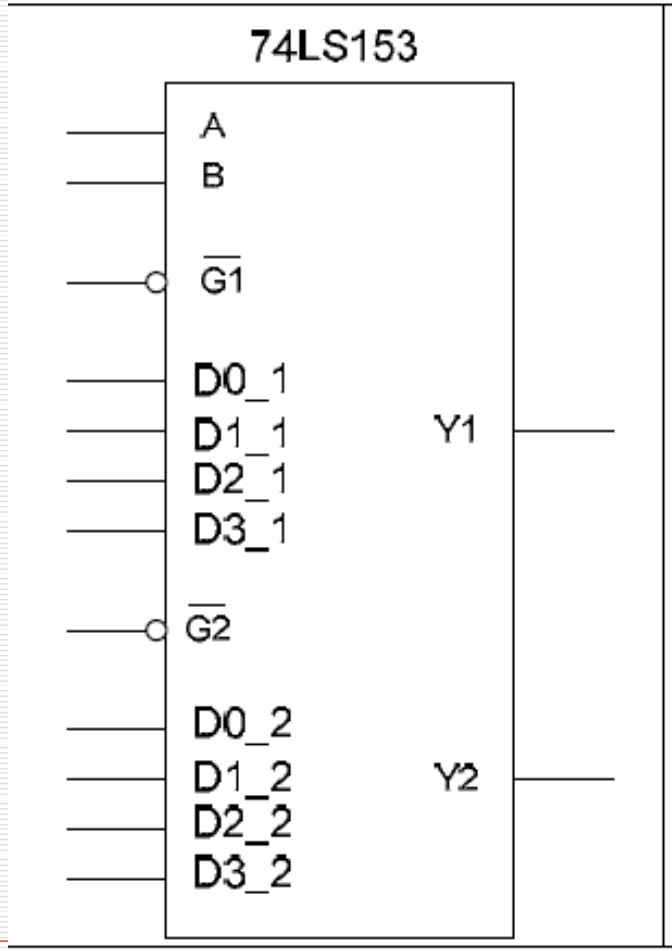
Slides by Philip Pham, University of California, Irvine

MUX 2-la-1 cu linie de validare



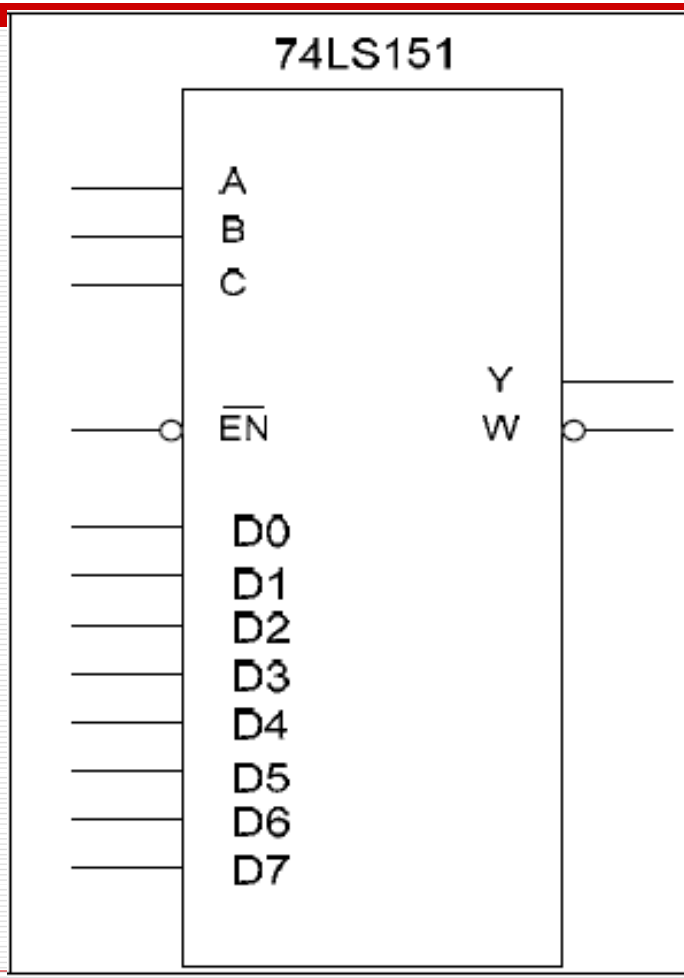
a	b	sel	c
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

74LS153: MUX 4-la-1



A	B	G1	G2	Y1	Y2
x	x	1	1	0	0
0	0	0	0	D0_1	D0_2
0	0	1	0	0	D0_2
0	0	0	1	D0_1	0
0	1	0	0	D1_1	D1_2
0	1	0	1	0	D1_2
0	1	1	0	D1_1	0
1	0	0	0	D2_1	D2_2
1	0	0	1	0	D2_2
1	0	1	0	D2_1	0
1	1	0	0	D3_1	D3_2
1	1	0	1	0	D3_2
1	1	1	0	D3_1	0

74LS151: MUX 8-la-1



A	B	C	EN	Y
x	x	x	1	0
0	0	0	0	D0
0	0	1	0	D1
0	1	0	0	D2
0	1	1	0	D3
1	0	0	0	D4
1	0	1	0	D5
1	1	0	0	D5
1	1	1	0	D7

Întrebări?

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