Proiectare logică

Curs 6

Metode de proiectare cu circuite MSI și LSI. Hazardul combinațional

Cristian Vancea

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

Cuprins

- Proiectare CLC cu circuite MSI – Exerciții
- Circuite LSI (Large Scale Integration) uzuale
- Proiectare CLC cu circuite LSI
- Circuite VLSI (Very Large Scale Integration) uzuale
- Hazardul combinațional

Circuite logice combinaționale

Sinteza funcțiilor booleene cu DCD și MUX – Exerciții

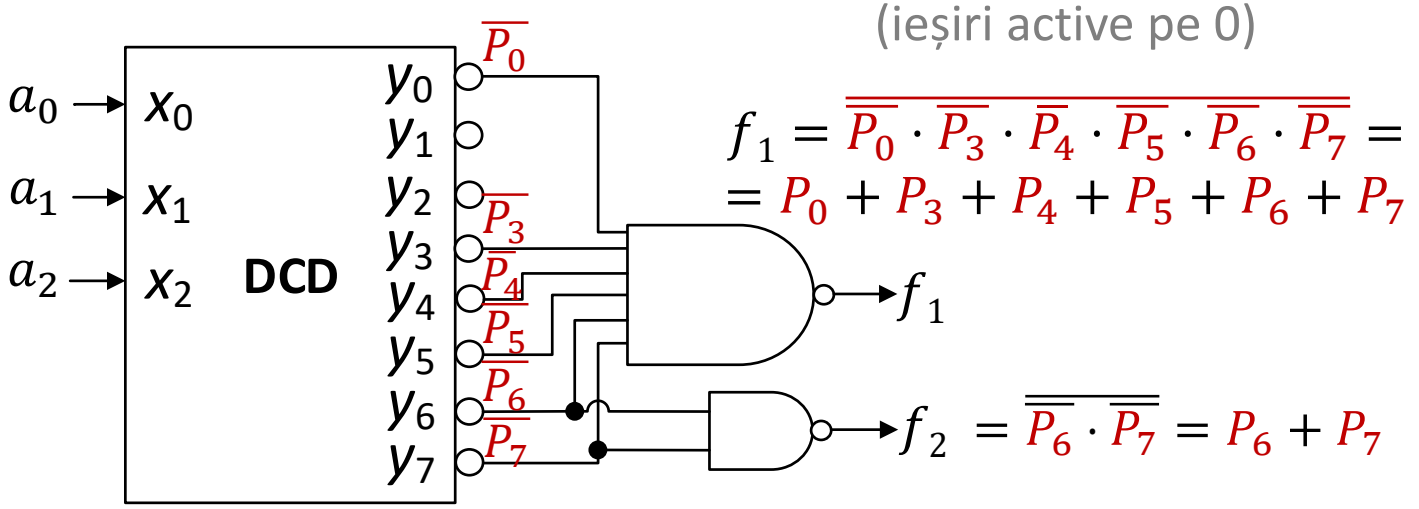
$\text{Ex}_1: n = 3 \quad f_1 = (a_0 \cdot a_1) + (a_0 \oplus \overline{a_1}) + a_2$

$f_2 = \sum(6, 7)$

	a_2	a_1	a_0	f_1
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

DCD cu ieșiri în logica negativă

(ieșiri active pe 0)

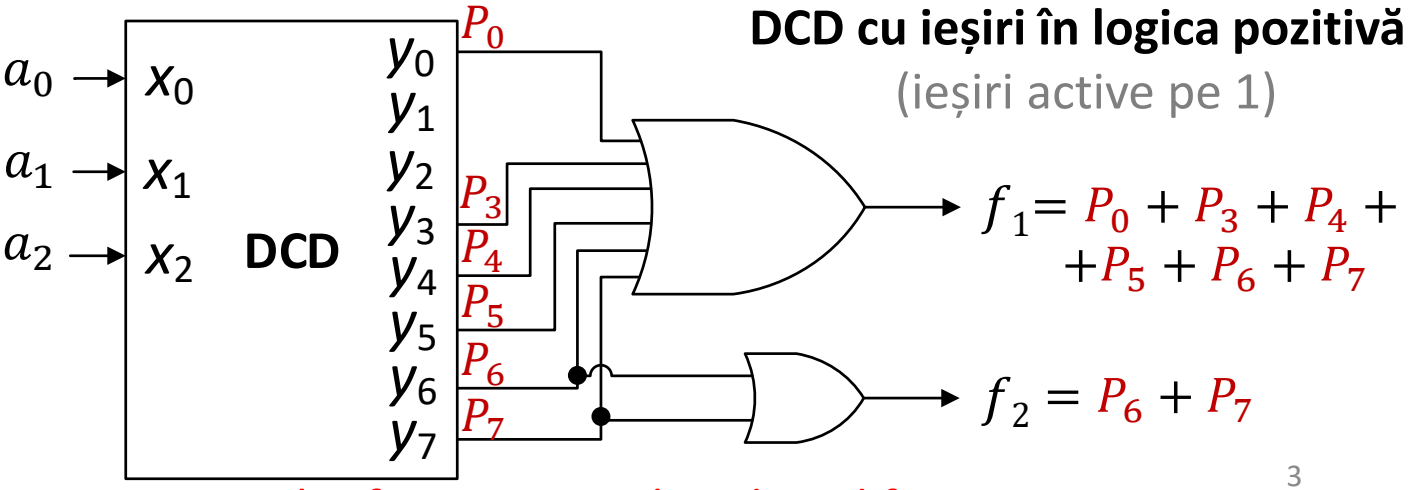


$f_1 = \overline{\overline{P_0} \cdot \overline{P_3} \cdot \overline{P_4} \cdot \overline{P_5} \cdot \overline{P_6} \cdot \overline{P_7}} =$

$= P_0 + P_3 + P_4 + P_5 + P_6 + P_7$

DCD cu ieșiri în logica pozitivă

(ieșiri active pe 1)



Notă: Se pot implementa mai multe funcții cu același decodificator.

Circuite logice combinaționale

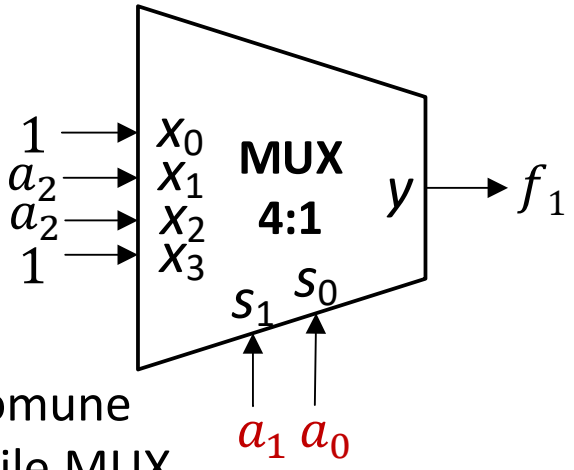
Sinteza funcțiilor booleene cu DCD și MUX – Exerciții

Ex$_2$: $n = 3$ $f_1 = (a_0 \cdot a_1) + (a_0 \oplus \overline{a_1}) + a_2$

	a_2	a_1	a_0	f_1
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

a_2 \ a_1a_0	00	01	11	10
0	1	0	1	0
1	1	1	1	1

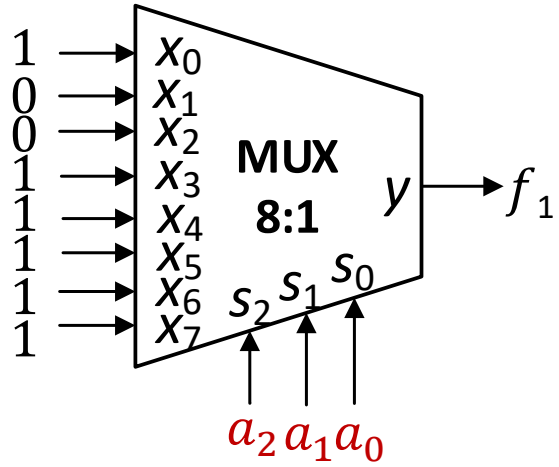
Implementare cu MUX 4:1



Obs: Se grupează după valorile comune ale variabilelor aplicate pe selecțiile MUX.

a_2 \ a_1a_0	00	01	11	10
0	1	0	1	0
1	1	1	1	1

Implementare cu MUX 8:1



Obs: Când toate variabilele se aplică pe selecțiile MUX fiecare celulă reprezintă o grupare aparte.

Circuite logice combinaționale

Sinteza funcțiilor booleene cu DCD și MUX – Exerciții

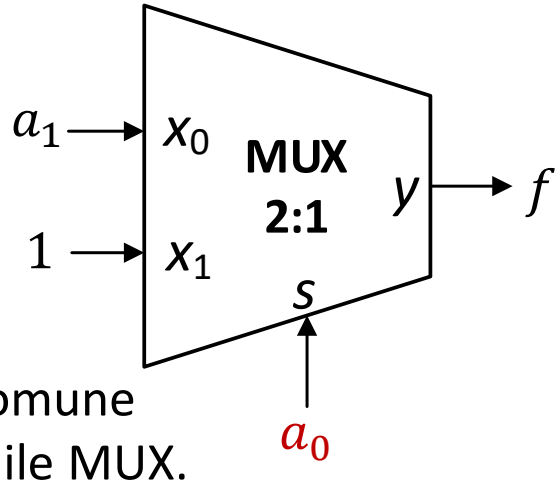
Ex$_3$: $n = 2 \;\; f = (\overline{a_0} + a_1) \oplus \overline{a_1}$

a_1 \ a_0	0	1
0	0	1
1	1	1

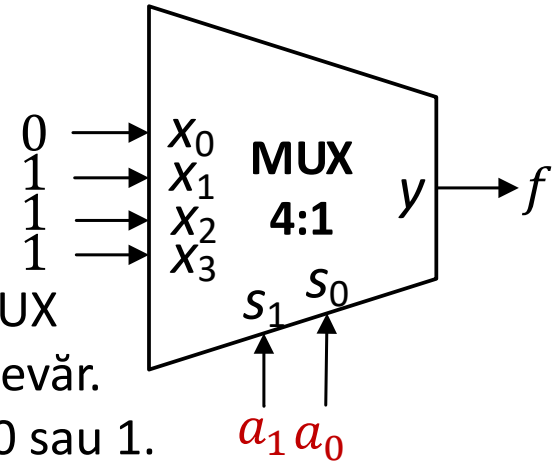
	a_1	a_0	f
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

Obs: Se grupează după valorile comune ale variabilelor aplicate pe selecțiile MUX.

Implementare cu MUX 2:1



Implementare cu MUX 4:1



Obs$_1$: Când toate variabilele se aplică pe selecțiile MUX datele de intrare în MUX se pot lua din tabelul de adevăr.

Obs$_2$: Pe liniile de date se aplică întotdeauna numai 0 sau 1.

Circuite logice combinaționale

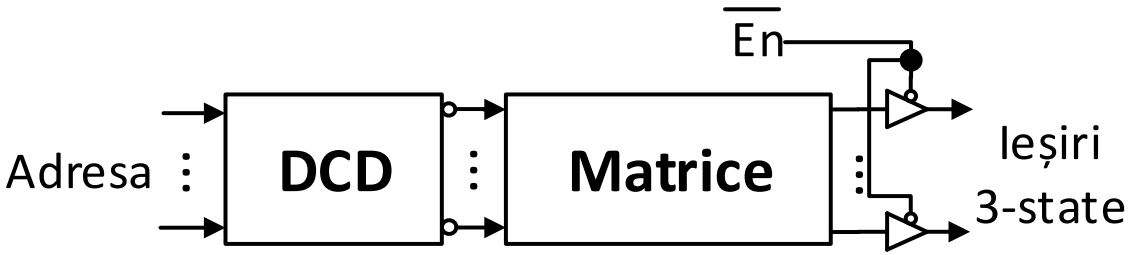
Circuite LSI (Large Scale Integration) uzuale

- LSI au peste 500 tranzistoare integrate.
- Exemple de circuite LSI :
  - Memorii ROM (Read Only Memory)
  - Unități PLA (Programmable Logic Array)

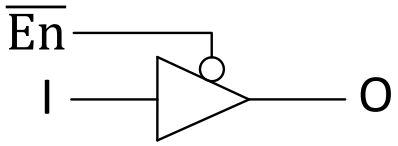
Circuite logice combinaționale

Circuite LSI uzuale – Memorii ROM

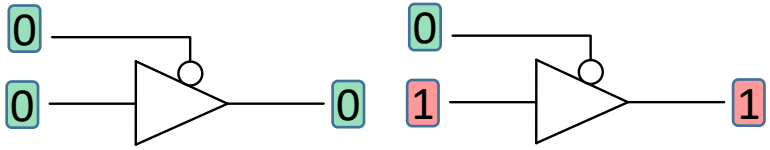
- Sunt memorii nevolatile – conținutul stabilit din fabricație nu se modifică în lipsa alimentării; Read Only Memory (ROM) - la funcționare conținutul doar se citește.
- Conțin 2 niveluri de circuite:
  - Nivelul 1 – decodificator (DCD);
  - Nivelul 2 – matrice de porți logice și conexiuni care implementează cuvintele de memorie. *1 rând din matrice ⇔ 1 cuvânt de memorie.*
- Intrările ROM coincid cu intrările DCD și codurile aplicate se numesc adrese. Pentru fiecare adresă se activează o linie de ieșire a DCD, care generează amplasarea la ieșirea ROM a unui cuvânt de memorie asociat.
- Ieșirile ROM sunt three-state (o linie poate fi 0, 1 sau Z – înaltă impedanță) => se pot conecta împreună ieșirile mai multor memorii: doar o memorie va fi activă restul având ieșirile în înaltă impedanță.
- Are o intrare de activare numită $\overline{\mathrm{En}}$ (Enable) sau $\overline{\mathrm{CS}}$ (Chip Select). Memoria inactivă ($\overline{\mathrm{En}} = 1$) are ieșirile în înaltă impedanță (Z).



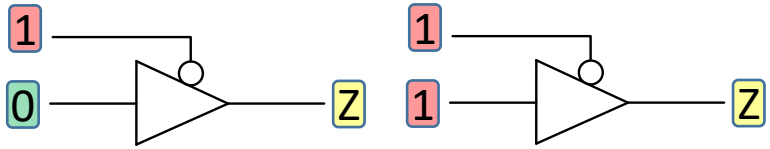
Buffere 3-state



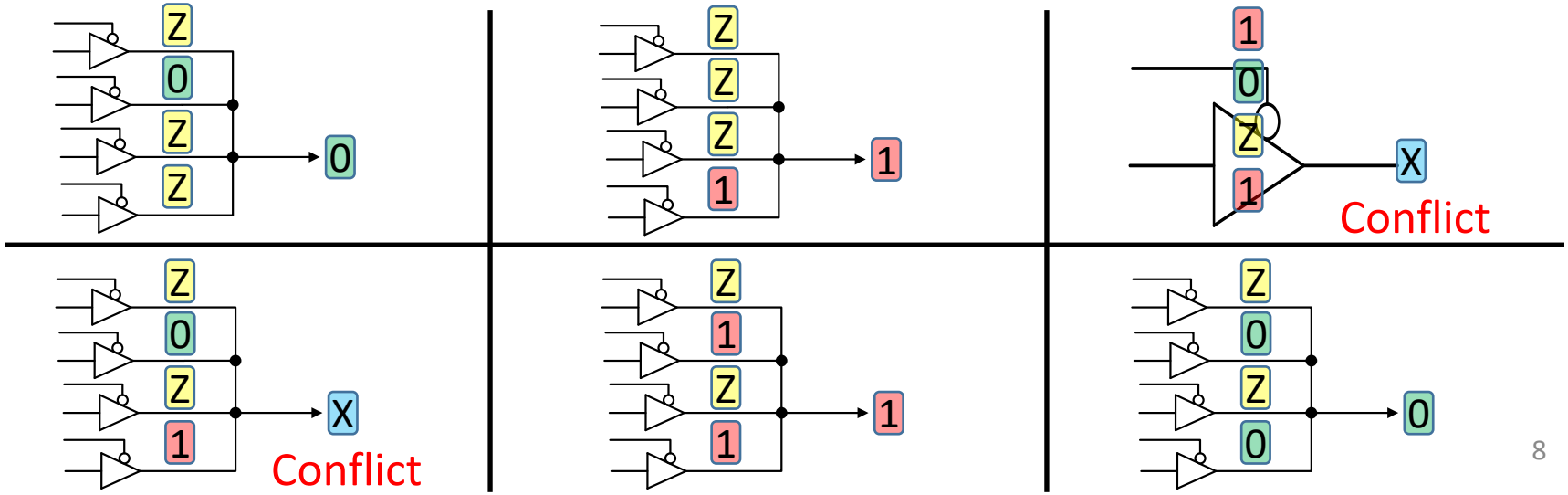
- Dacă este activ ($\overline{\text{En}} = 0$) permite trecerea semnalului de la intrarea I la ieșirea O.



- Dacă este în înaltă impedanță ($\overline{\text{En}} = 1$) nu permite trecerea semnalului între intrarea I și ieșirea O.



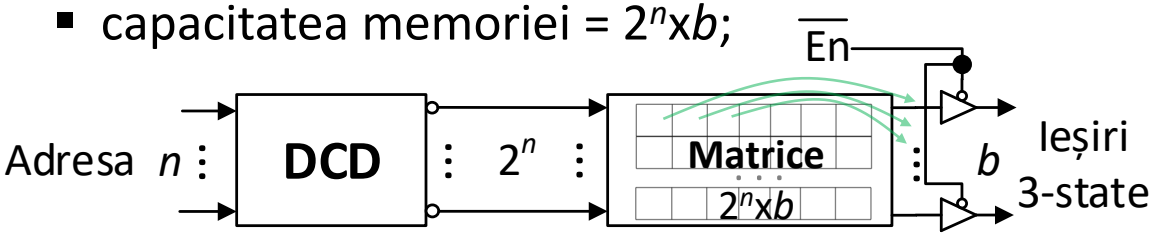
- Se pot conecta mai multe ieșiri cu buffer 3-state. Se vor evita conflictele (0 și 1 concomitent) pe linia comună. Exemple de situații posibile:



Circuite logice combinaționale

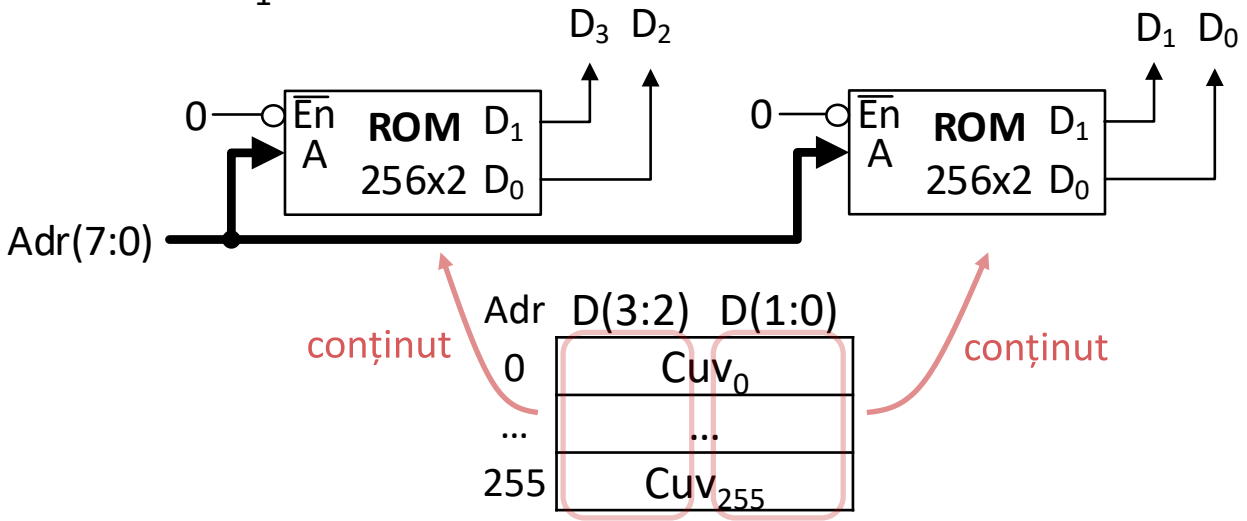
Circuite LSI uzuale – Memorii ROM

- Organizarea memoriei:
  - n – numărul de biți de adresă => numărul de cuvinte de memorie = 2^n;
  - b – numărul de biți ai unui cuvânt de memorie (de obicei putere a lui 2);
  - capacitatea memoriei = 2^nxb;



  - se poate mări capacitatea folosind mai multe memorii:

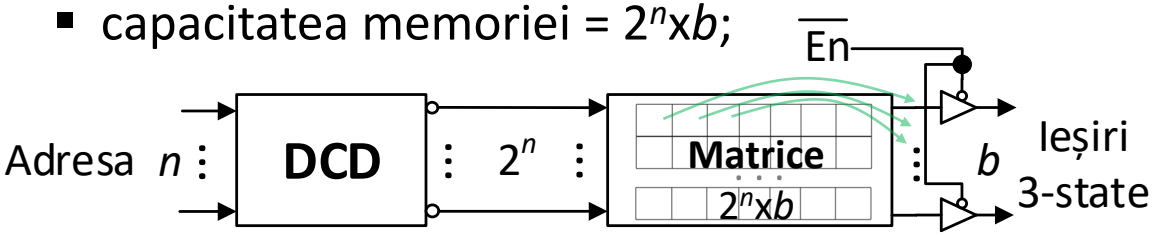
Ex$_1$: Mărirea cuvântului de memorie de la 2 la 4 biți la ROM cu 256 cuvinte



Circuite logice combinaționale

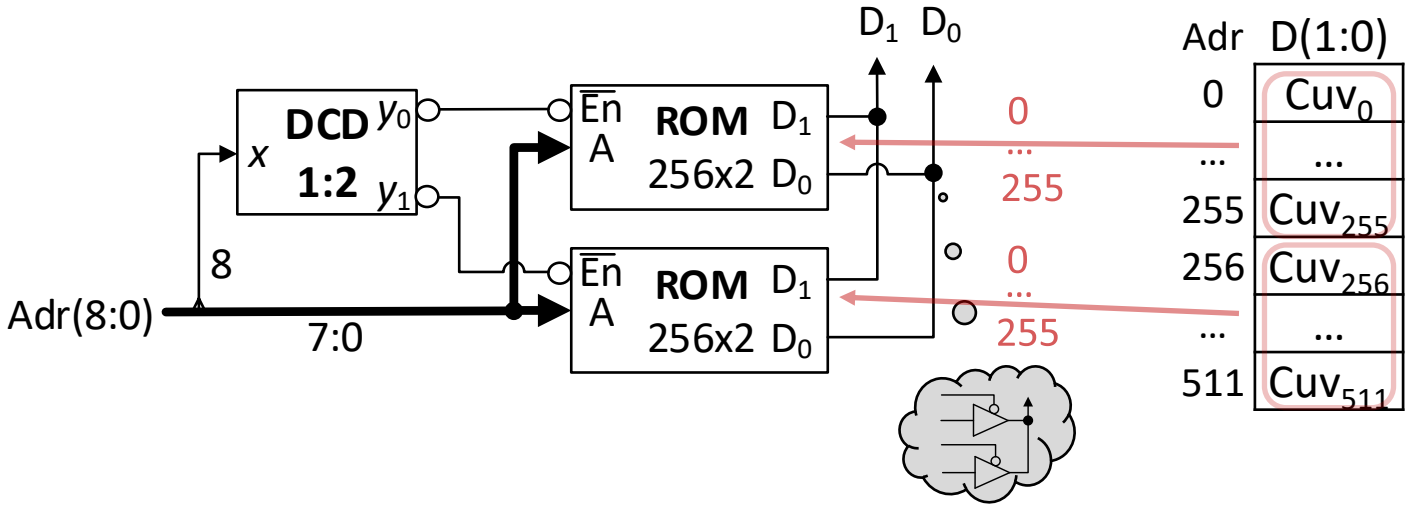
Circuite LSI uzuale – Memorii ROM

- Organizarea memoriei:
  - n – numărul de biți de adresă => numărul de cuvinte de memorie = 2^n;
  - b – numărul de biți ai unui cuvânt de memorie (de obicei putere a lui 2);
  - capacitatea memoriei = 2^nxb;



  - se poate mări capacitatea folosind mai multe memorii:

Ex$_2$: Dublarea numărului de cuvinte de la 256 la 512 la ROM cu 2 biți / ieșire

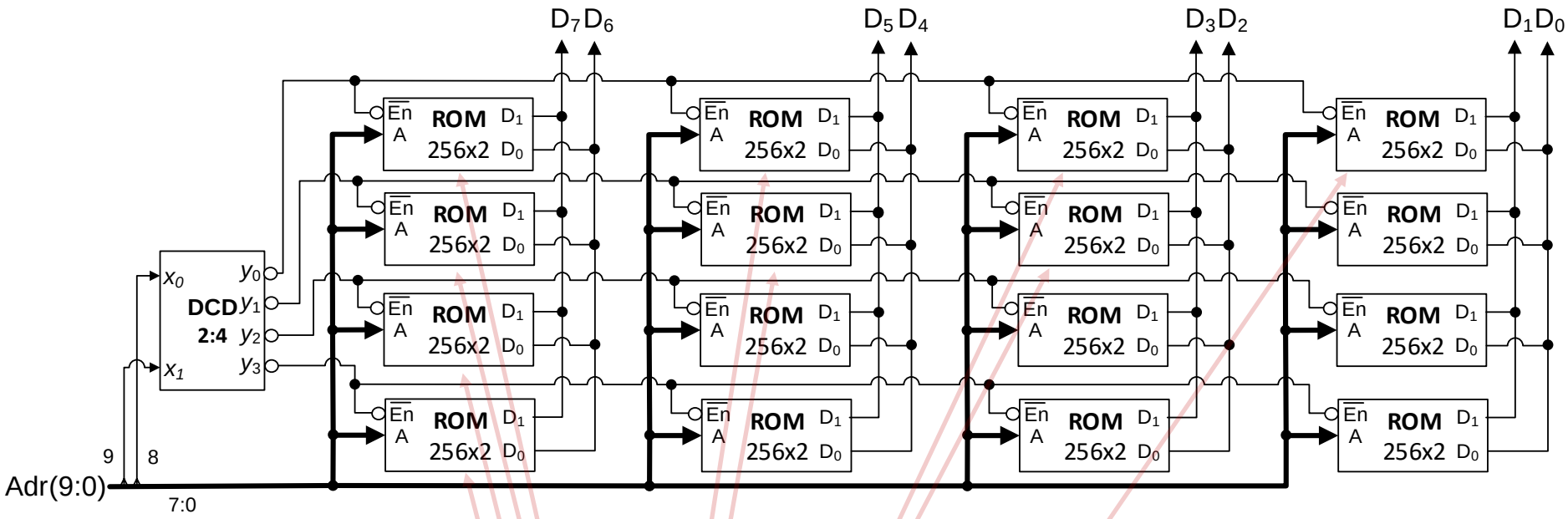


B10	B2
0	000000000
1	000000001
...	
255	011111111
256	100000000
257	100000001
...	
255	111111111

Circuite logice combinaționale

Circuite LSI uzuale – Memorii ROM

Ex$_3$: 4 x numărul de cuvinte + 4 x numărul de biți / cuvânt => ROM 1024x8 (1KB)



Adr	D(7:6)	D(5:4)	D(3:2)	D(1:0)
0…255				
256…511				
512…767				
768…1023				

Distribuirea conținutului memoriei de 1KB (harta de biți) pe unități ROM 256x2

Circuite logice combinaționale

Circuite LSI uzuale – Memorii ROM

Aplicații specifice:

- Memorie de (micro)instrucțiuni sau de date;
- Conversie de coduri;
- Stocare coduri de caractere;
- Generare secvențe de impulsuri predefinite;
- Implementare CLC cu număr mare de variabile de intrare și ieșire.

Circuite logice combinaționale

Sinteza funcțiilor booleene cu memorii ROM

- Nu necesită minimizarea funcțiilor; are la bază aducerea funcțiilor de implementat la forma canonică disjunctivă (FCD) deoarece:
  - Variabilele de intrare se aplică pe liniile de adresă ale memoriei => la nivelul DCD se implementează toți mintermii ca intrări ale matricei.
  - Conținutul matricei este configurat încât să determine disjuncția SAU a mintermilor necesari în implementarea funcției. Fiecare bit de ieșire va reprezenta o funcție de implementat.
- Etape:
  1. Stabilirea dimensiunii memoriei potrivite; dacă este necesar se pot utiliza mai multe memorii de dimensiune redusă pentru obținerea unei capacități suficiente.
  2. Stabilirea tabelului de adevăr al memoriei ⇔ configurarea matricei de cuvinte de memorie (harta de biți) în conformitate cu funcțiile de implementat.

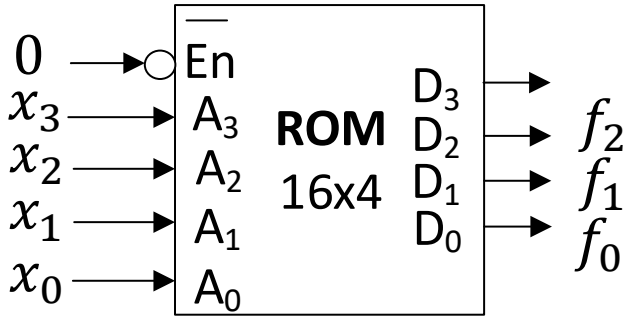
Circuite logice combinaționale

Sinteza funcțiilor booleene cu memorii ROM

Ex$_1$: $n = 4$ $f_0 = \sum(0,1,2,3,6,9,11,12,14)$, $f_1 = \sum(1,2,3,5,7)$, $f_2 = \sum(11)$

1. Alegerea capacității memoriei

$2^n \times b = 2^4 \times 4 = 16 \times 4$



Obs: Chiar dacă avem 3 funcții, b trebuie ales ca putere a lui 2. Deci $b = 4$ ieșiri => o ieșire va fi ignorată (în acest caz D$_3$).

Circuite logice combinaționale

Sinteza funcțiilor booleene cu memorii ROM

Ex$_1$: $n = 4$ $f_0 = \sum(0,1,2,3,6,9,11,12,14)$, $f_1 = \sum(1,2,3,5,7)$, $f_2 = \sum(11)$

2. Configurarea tabelului de adevăr al memoriei (harta de biți):

- se pune 1 pe coloana asociată funcției în dreptul mintermilor constituenți și 0 în rest;
- se poate înscrie orice în coloana pentru ieșirea ignorată D_3.

Obs: Fiecare rând asociat unei adrese va reprezenta conținutul unui cuvânt de memorie.

Adrese				Cuvinte de memorie			
A_3	A_2	A_1	A_0	D_3	$D_2=f_2$	$D_1=f_1$	$D_0=f_0$
0	0	0	0	X	0	0	1
0	0	0	1	X	0	1	1
0	0	1	0	X	0	1	1
0	0	1	1	X	0	1	1
0	1	0	0	X	0	0	0
0	1	0	1	X	0	1	0
0	1	1	0	X	0	0	1
0	1	1	1	X	0	1	0
1	0	0	0	X	0	0	0
1	0	0	1	X	0	0	1
1	0	1	0	X	0	0	0
1	0	1	1	X	1	0	1
1	1	0	0	X	0	0	1
1	1	0	1	X	0	0	0
1	1	1	0	X	0	0	1
1	1	1	1	X	0	0	0

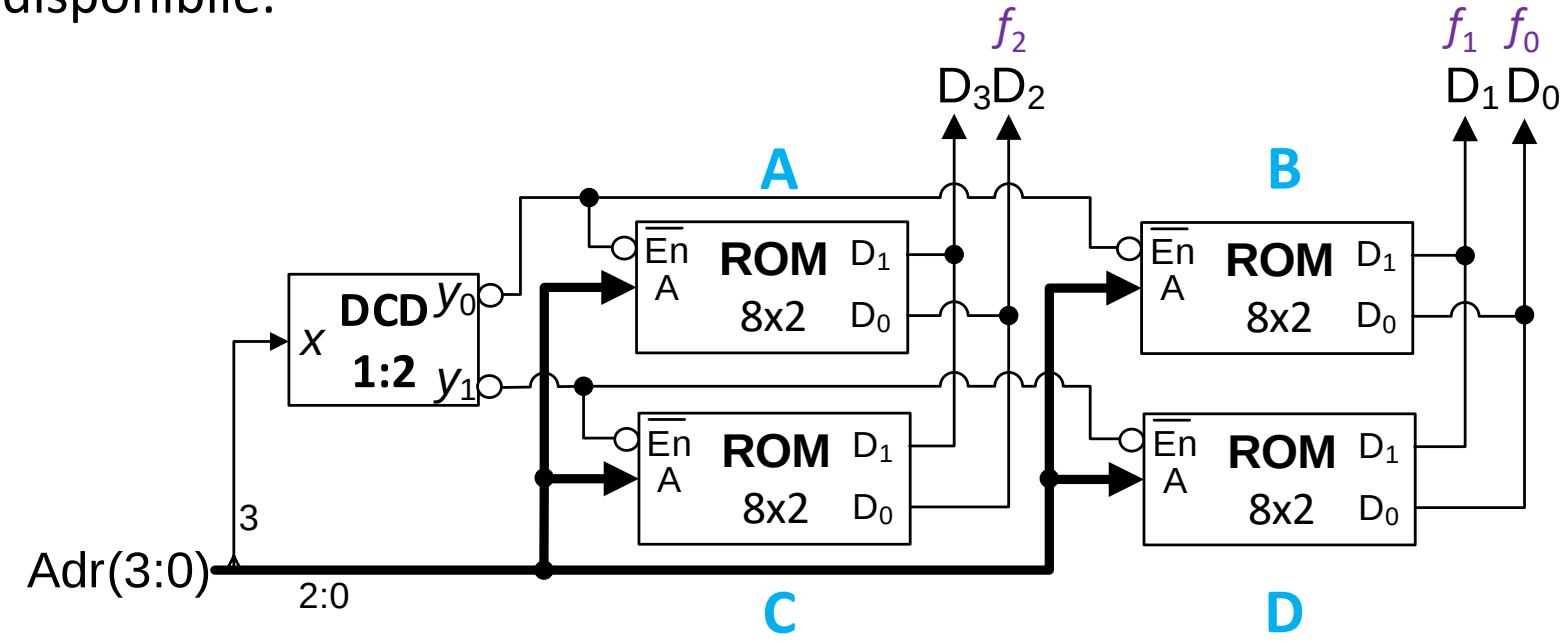
Circuite logice combinaționale

Sinteza funcțiilor booleene cu memorii ROM

Ex$_1$: $n = 4$ $f_0 = \sum(0,1,2,3,6,9,11,12,14)$, $f_1 = \sum(1,2,3,5,7)$, $f_2 = \sum(11)$

Dacă nu avem la dispoziție memoria necesară ci doar memorii mai mici (ex. ROM 8x2) sunt necesari pașii 3 și 4.

Pasul 3. Se construiește memoria necesară ROM 16x4 cu memoriile ROM 8x2 disponibile.



Circuite logice combinaționale

Sinteza funcțiilor booleene cu memorii ROM

Ex$_1$: $n = 4$ $f_0 = \sum(0,1,2,3,6,9,11,12,14)$, $f_1 = \sum(1,2,3,5,7)$, $f_2 = \sum(11)$

Pasul 4. Se distribuie conținutul memoriei mari ROM 16x4 pe memoriile disponibile și se construiește harta de biți pentru fiecare ROM 8x2 utilizat.

A_3	A_2	A_1	A_0	D_3	$D_2=f_2$	$D_1=f_1$	$D_0=f_0$
0	0	0	0	X	0	0	1
0	0	0	1	X	0	1	1
0	0	1	0	X	0	1	1
0	0	1	1	X	0	1	1
0	1	0	0	X	0	0	0
0	1	0	1	X	0	1	0
0	1	1	0	X	0	0	1
0	1	1	1	X	0	1	0
1	0	0	0	X	0	0	0
1	0	0	1	X	0	0	1
1	0	1	0	X	0	0	0
1	0	1	1	X	1	0	1
1	1	0	0	X	0	0	1
1	1	0	1	X	0	0	0
1	1	1	0	X	0	0	1
1	1	1	1	X	0	0	0

A

A_2	A_1	A_0	D_1	D_0
0	0	0	X	0
0	0	1	X	0
0	1	0	X	0
0	1	1	X	0
1	0	0	X	0
1	0	1	X	0
1	1	0	X	0
1	1	1	X	0

B

A_2	A_1	A_0	D_1	D_0
0	0	0	0	1
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

C

A_2	A_1	A_0	D_1	D_0
0	0	0	X	0
0	0	1	X	0
0	1	0	X	0
0	1	1	X	1
1	0	0	X	0
1	0	1	X	0
1	1	0	X	0
1	1	1	X	0

D

A_2	A_1	A_0	D_1	D_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	1
1	0	1	0	0
1	1	0	0	1
1	1	1	0	0

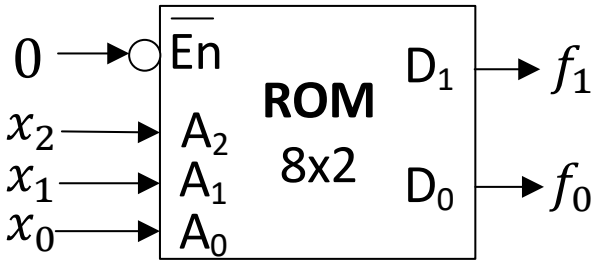
Circuite logice combinaționale

Sinteza funcțiilor booleene cu memorii ROM

Ex$_2$: $n = 3 \;\; f_0 = x_2 + (x_1 \cdot \overline{x_0}), f_1 = x_1 \oplus x_2$

$$f_0 \overset{\text{FCD}}{\Longrightarrow} \sum(2,4,5,6)\,, f_1 \overset{\text{FCD}}{\Longrightarrow} \sum(2,3,4,5)$$

1. Alegerea capacității memoriei
$2^n \times b = 2^3 \times 2 = 8 \times 2$



2. Configurarea tabelului de adevăr

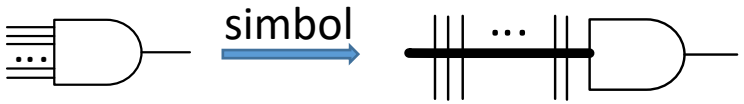
A_2	A_1	A_0	$D_1=f_1$	$D_0=f_0$
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	0	1
1	1	1	0	0

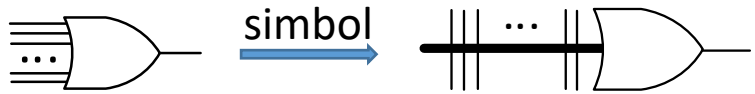
x_2	x_1	x_0	f_1	f_0
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	0	1
1	1	1	0	0

Circuite logice combinaționale

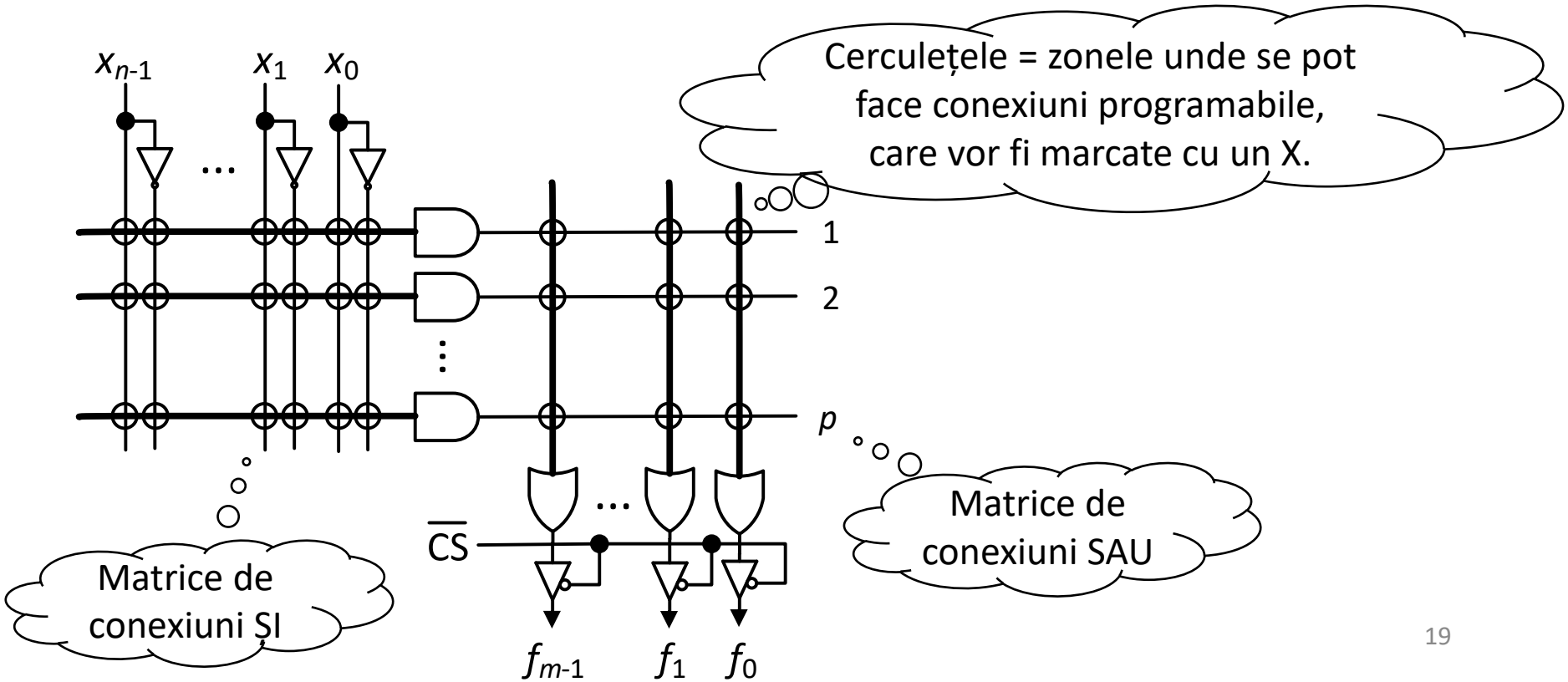
Circuite LSI uzuale – Unități PLA

- Conțin 2 niveluri de logică programabilă: un șir de porți ȘI, un șir de porți SAU. Porțile ȘI și SAU au mai multe intrări:





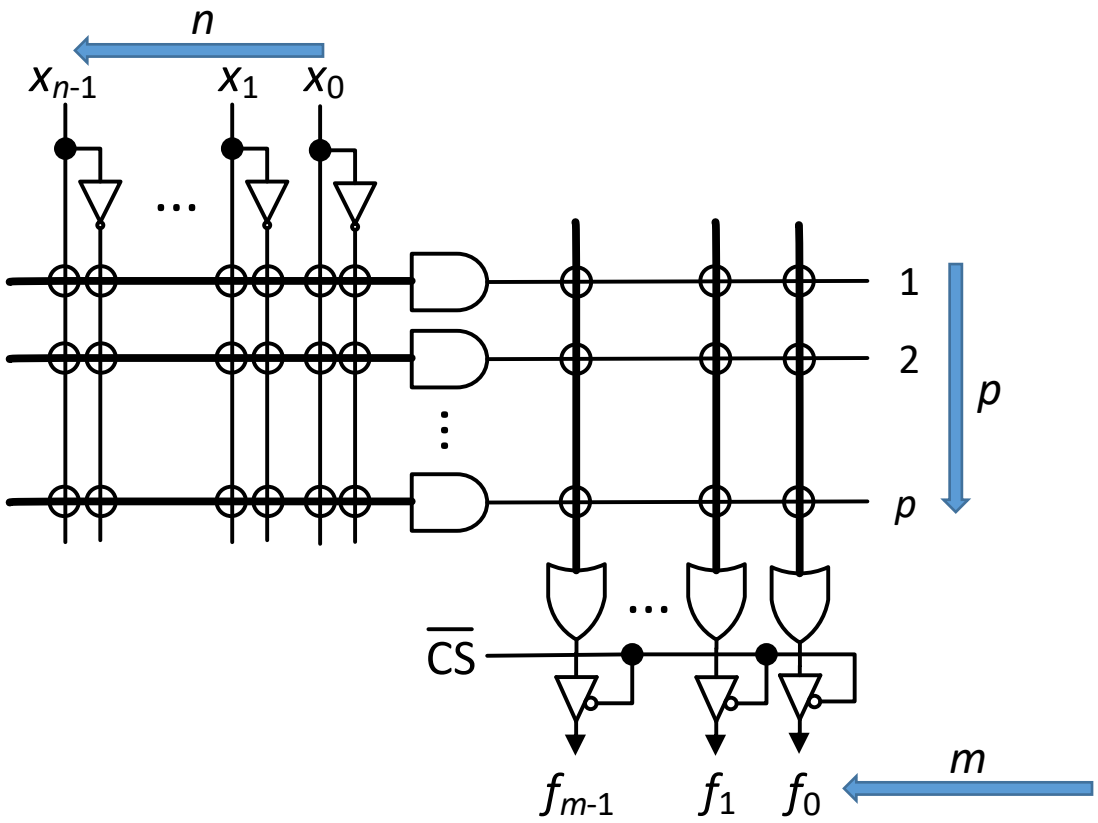
- Ieșirile PLA sunt three-state. Există intrare de activare numită $\overline{CS}$ (Chip Select). Circuitul inactiv ($\overline{CS} = 1$) are ieșirile în înaltă impedanță (Z).



Circuite logice combinaționale

Circuite LSI uzuale – Unități PLA

- Structura PLA:
  - *n* linii de intrare;
  - *p* porți ȘI cu *n* intrări fiecare => maxim *p* termeni elementari;
  - *m* linii de ieșire => *m* porți SAU cu maxim *p* intrări fiecare.



Circuite logice combinaționale

Circuite LSI uzuale – Unități PLA

Aplicații specifice:

- Microprogramare;
- Conversie de coduri;
- Generare coduri de caractere;
- Implementare seturi de funcții.

Circuite logice combinaționale

Sinteza funcțiilor booleene cu unități PLA

- Necesită aducerea funcțiilor la forma disjunctivă minimă (FDM) pentru a reduce numărul de conexiuni necesare. Ca urmare:
  - La nivelul matricei de conexiuni ȘI se generează termenii elementari.
  - La nivelul matricei de conexiuni SAU se realizează disjuncția SAU a termenilor elementari.
- Etape:
  1. Exprimarea funcțiilor în forma disjunctivă minimă (FDM) – "sumă de produse".
  2. Realizarea legăturilor la nivelul matricelor de conexiuni ȘI și SAU.

Obs: Dacă funcțiile au termeni elementari comuni, aceștia se implementează o singură dată.

Circuite logice combinaționale

Sinteza funcțiilor booleene cu unități PLA

Ex$_1$: $n = 4$ $f_0 = \sum(0,1,2,3,6,9,11,12,14)$, $f_1 = \sum(1,2,3,5,7)$, $f_2 = \sum(11)$

1. Minimizarea funcțiilor la FDM

f_0:

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

FDM $f_0 = (\overline{x_3} \cdot \overline{x_2}) + (\overline{x_2} \cdot x_0) + (x_3 \cdot x_2 \cdot \overline{x_0}) + (x_2 \cdot x_1 \cdot \overline{x_0})$

f_1:

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

FDM $f_1 = (\overline{x_3} \cdot x_0) + (\overline{x_3} \cdot \overline{x_2} \cdot x_1)$

f_2:

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

FDM $f_2 = x_3 \cdot \overline{x_2} \cdot x_1 \cdot x_0$

Circuite logice combinaționale

Sinteza funcțiilor booleene cu unități PLA

Ex$_1$: $n = 4$ $f_0 = \sum(0,1,2,3,6,9,11,12,14)$, $f_1 = \sum(1,2,3,5,7)$, $f_2 = \sum(11)$

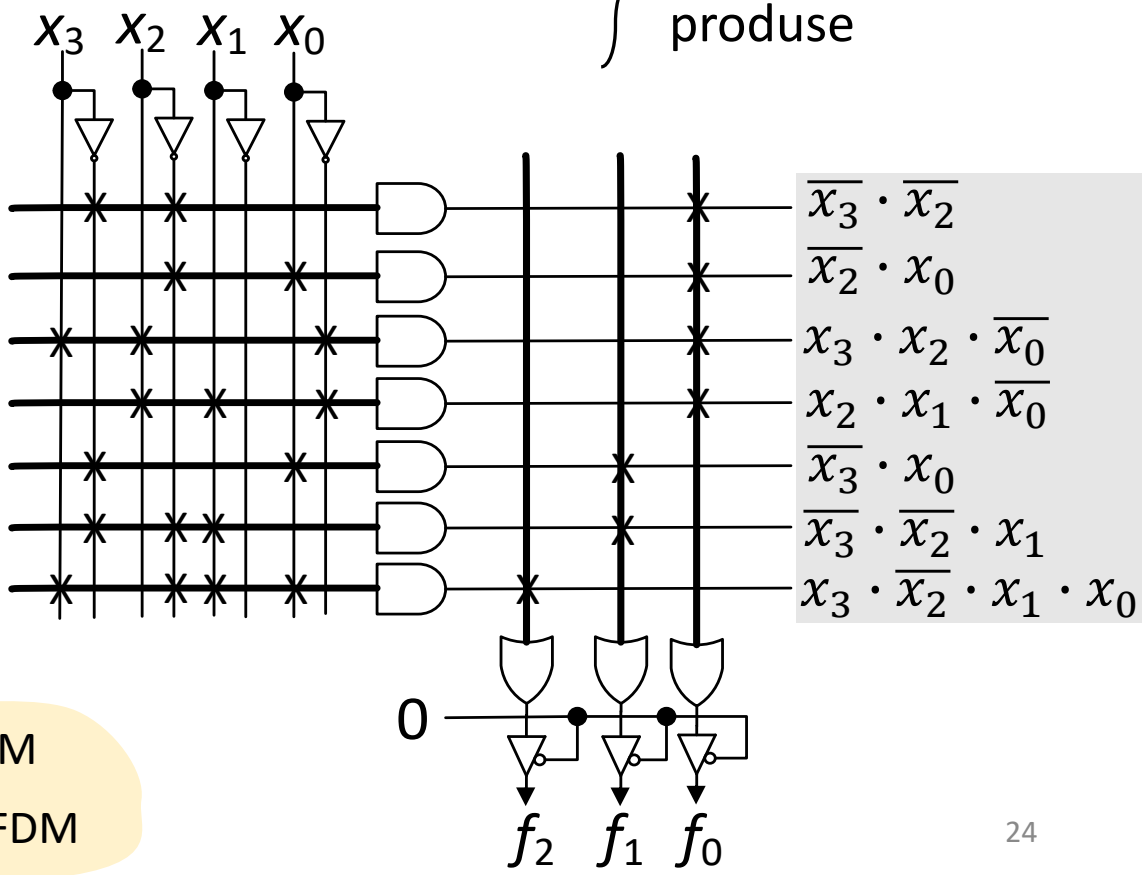
2. Realizarea conexiunilor

$f_0 = (\overline{x_3} \cdot \overline{x_2}) + (\overline{x_2} \cdot x_0) + (x_3 \cdot x_2 \cdot \overline{x_0}) + (x_2 \cdot x_1 \cdot \overline{x_0})$

$f_1 = (\overline{x_3} \cdot x_0) + (\overline{x_3} \cdot \overline{x_2} \cdot x_1)$

$f_2 = x_3 \cdot \overline{x_2} \cdot x_1 \cdot x_0$

} FDM = sumă de produse



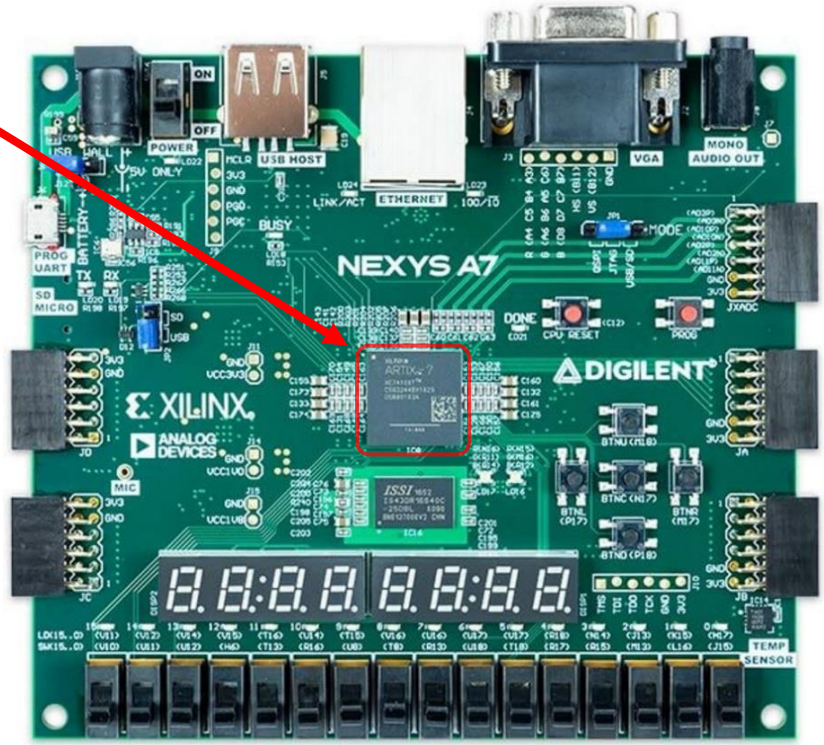
$f_1 = (\overline{x_3} \cdot x_0) + (\overline{x_3} \cdot \overline{x_2} \cdot x_1)$ FDM

$= \overline{x_3} \cdot (x_0 + \overline{x_2} \cdot x_1)$ Nu este FDM

Circuite logice combinaționale

Circuite VLSI (Very Large Scale Integration) uzuale

- Prezintă un grad extins de integrare.
- Exemple de circuite VLSI:
  - CPLD (Complex Programmable Logic Device);
  - FPGA (Field Programmable Gate Array).



https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual

Circuite logice combinaționale

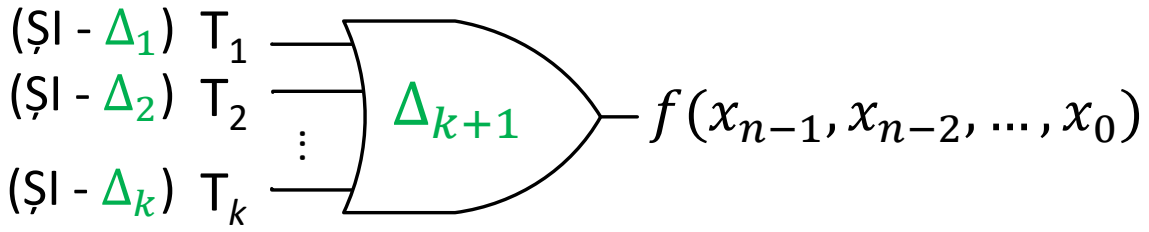
Hazardul combinațional

Ex$_1$: $n = 4 \;\; f = \sum(1,3,5,7,8,9,12,13)$

x_3x_2 \ 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

$$f \overset{\text{FDM}}{\Longrightarrow} \underbrace{\overline{x_3} \cdot x_0}_{T_1} + \underbrace{x_3 \cdot \overline{x_1}}_{T_2}$$

- Obs: Dacă $T_1, T_2, \ldots, T_k$ sunt termeni elementari obținuți după minimizare în FDM, atunci f se implementează cu SAU peste aceștia:



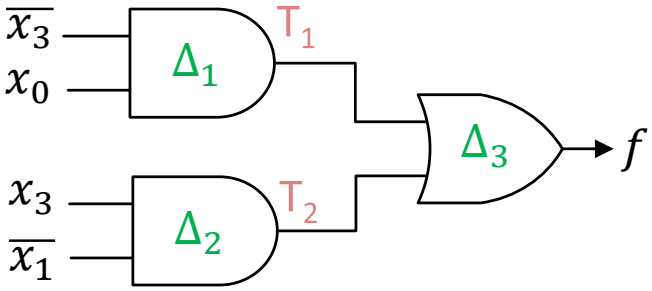
- Orice modificare a variabilelor de intrare x_i poate să genereze modificarea unuia sau mai multor termeni elementari T_j cu întârzieri diferite Δ_j.

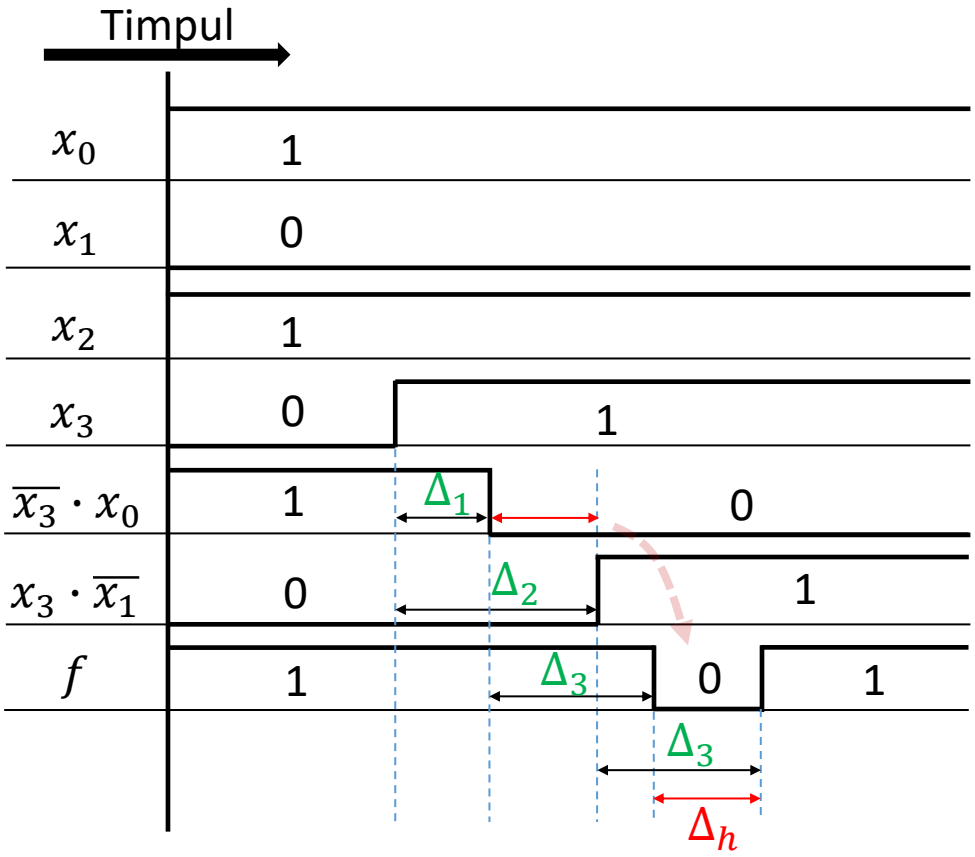
Circuite logice combinaționale

Hazardul combinațional

$\text{Ex}_1: n = 4 \;\; f = \sum(1,3,5,7,8,9,12,13) =$

$\overset{\text{FDM}}{\Longrightarrow} \overline{x_3} \cdot x_0 + x_3 \cdot \overline{x_1}$ (T_1: $\overline{x_3} \cdot x_0$, T_2: $x_3 \cdot \overline{x_1}$)





grupuri adiacente

x_3x_2 \ 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

Δ_h - perioada de hazard combinațional

Obs: Hazardul apare când grupurile asociate termenilor elementari sunt vecine pe linii sau coloane în Diagrama Karnaugh și nu se intersectează.

Circuite logice combinaționale

Hazardul combinațional

Definiție: **Hazardul combinațional** – comportare temporară greșită a CLC în care ieșirea are valoare necorespunzătoare cu modificările aduse intrărilor, datorită întârzierilor din circuit.

Tipuri:

- Static – când se modifică o variabilă –> se poate elimina.
- De funcție – când se modifică mai multe variabile –> eliminare dificilă/imposibilă.

Circuite logice combinaționale

Hazardul combinațional

Soluție pentru eliminarea hazardului static

Pas$_1$: Se identifică toate grupările utilizate pentru FDM care sunt vecine și nu se intersectează (adiacente).

Ex$_1$: $n = 4 \;\; f = \sum(1,3,5,7,8,9,12,13)$

x_3x_2 \ 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

Pas$_2$: Dacă există astfel de grupări se adaugă un număr minim de grupări dreptunghiulare maximale de 1 care realizează unirea lor. Grupările trebuie să conțină un număr de celule putere a lui 2.

Se adaugă termenii corespunzători la FDM.

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

x_3x_2 \ 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

Circuite logice combinaționale

Hazardul combinațional

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

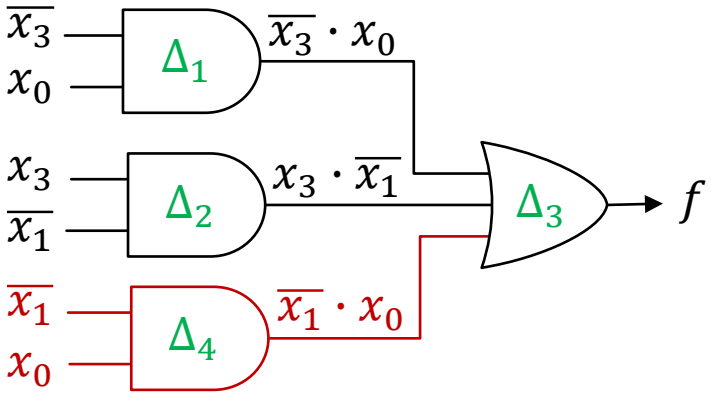
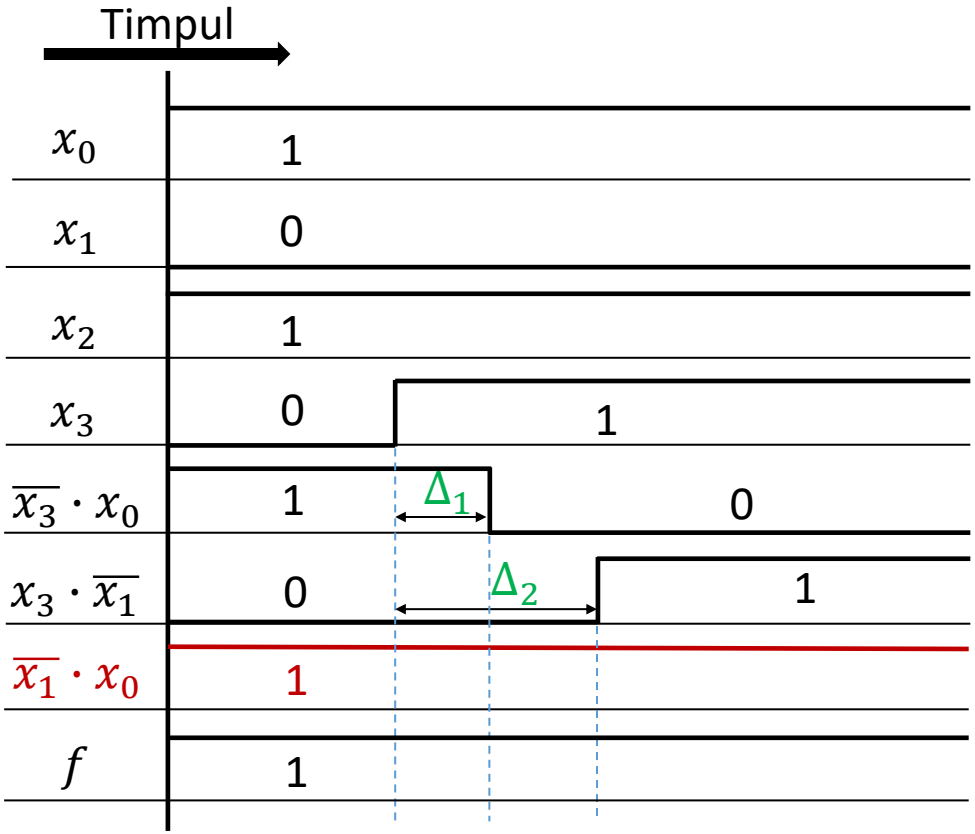


Diagrama de timp pentru circuitul fără hazard



=> Hazardul static nu mai apare!

Circuite logice combinaționale

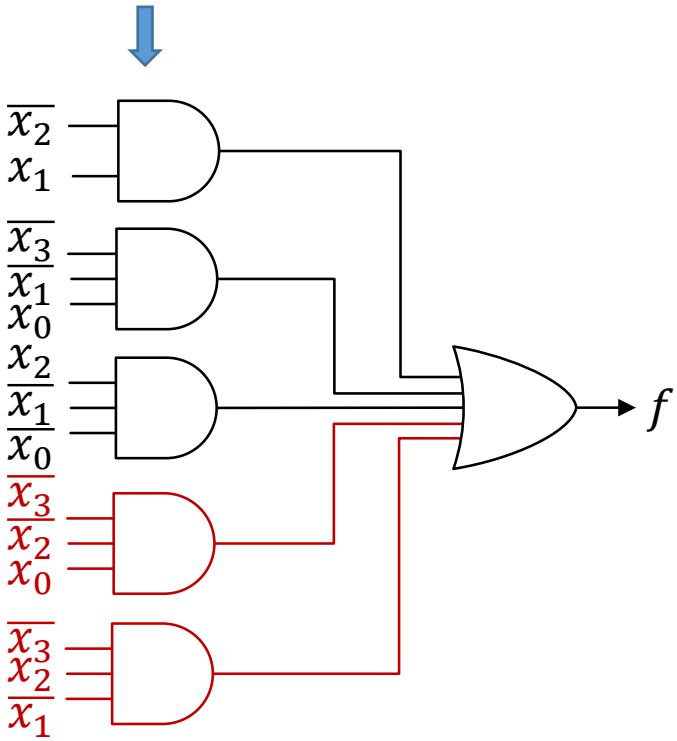
Hazardul combinațional

$\text{Ex}_2: n = 4 \;\; f = \sum(1,2,3,4,5,10,11,12) =$

$\overset{\text{FDM}}{\Longrightarrow} \overline{x_2} \cdot x_1 + \overline{x_3} \cdot \overline{x_1} \cdot x_0 + x_2 \cdot \overline{x_1} \cdot \overline{x_0} +$

$+ \overline{x_3} \cdot \overline{x_2} \cdot x_0 + \overline{x_3} \cdot x_2 \cdot \overline{x_1} =$

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



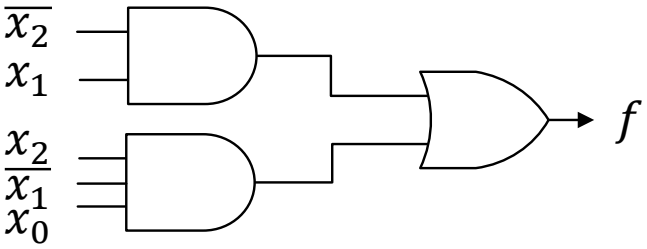
Circuite logice combinaționale

Hazardul combinațional

Ex$_3$: $n = 4 \;\; f = \sum(2,3,5,10,11,13) =$

$$\overset{\text{FDM}}{\Longrightarrow} \overline{x_2} \cdot x_1 + x_2 \cdot \overline{x_1} \cdot x_0$$

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



FDM nu prezintă hazard static deoarece nu există grupări vecine.

Circuite logice combinaționale

Hazardul combinațional

Soluție alternativă - se întârzie citirea rezultatului un interval de timp suficient de mare încât valorile pe liniile de ieșire să se stabilizeze:

- Lentă;
- Sigură – rezolvă hazardul combinațional static și de funcție;
- Consum redus de resurse (nu apar termeni suplimentari).