Arhitectura sistemelor de calcul

- Prelegerea 7 -

0-DS

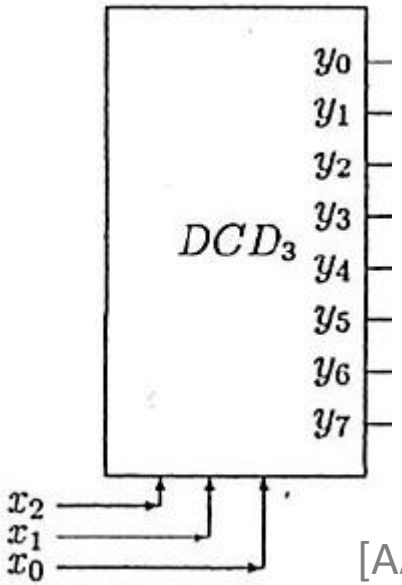
Ruxandra F. Olimid

Facultatea de Matematică şi Informatică
Universitatea din Bucureşti

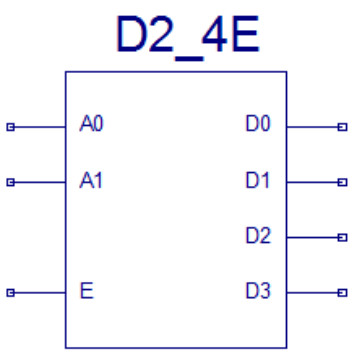
Cuprins

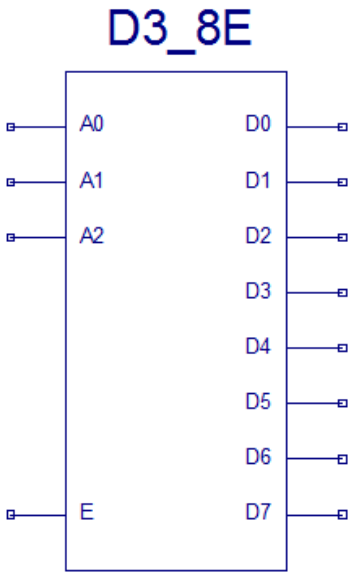
Decodificatoare

- *Scop:* standardizarea şi determinarea semnalului de intrare
- *Abreviere:* DCD (decodificator); EDCD (decodificator elementar)
- *Mod de funcţionare*: o singură ieşire este activă la un moment dat; aceasta determină semnalul de intrare
- *Reprezentare schematică:*



[AAT]

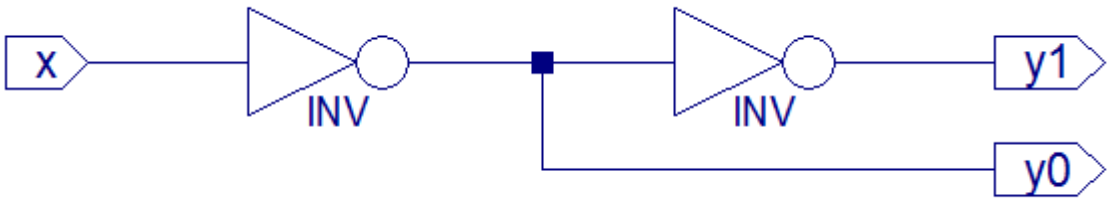




[Xilinx - ISE]

Decodificatoare

- *EDCD:* 1 singură intrare; 2 ieşiri
  - ✓ Ieşirea y_0 devine activă dacă intrarea are valoarea 0
  - ✓ Ieşirea y_1 devine activă dacă intrarea are valoarea 1



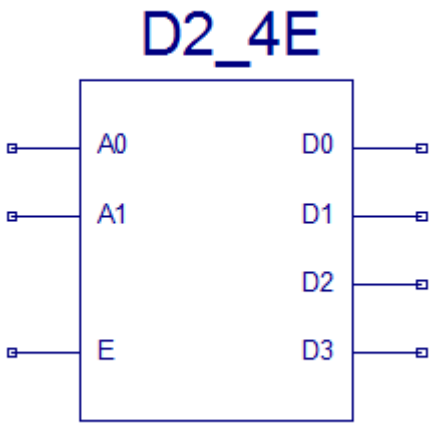
[Xilinx - ISE]

Decodificatoare

- *Def.:* Un DCD_n este un circuit combinational cu intrarea $X = \{0,1\}^n$ şi ieşirea Y pe 2^n biţi astfel încât:
  - ✓ Fiecare ieşire este activă pentru o singură configuraţie la intrare
  - ✓ Fiecare configuraţie la intrare activează un singur bit de ieşire

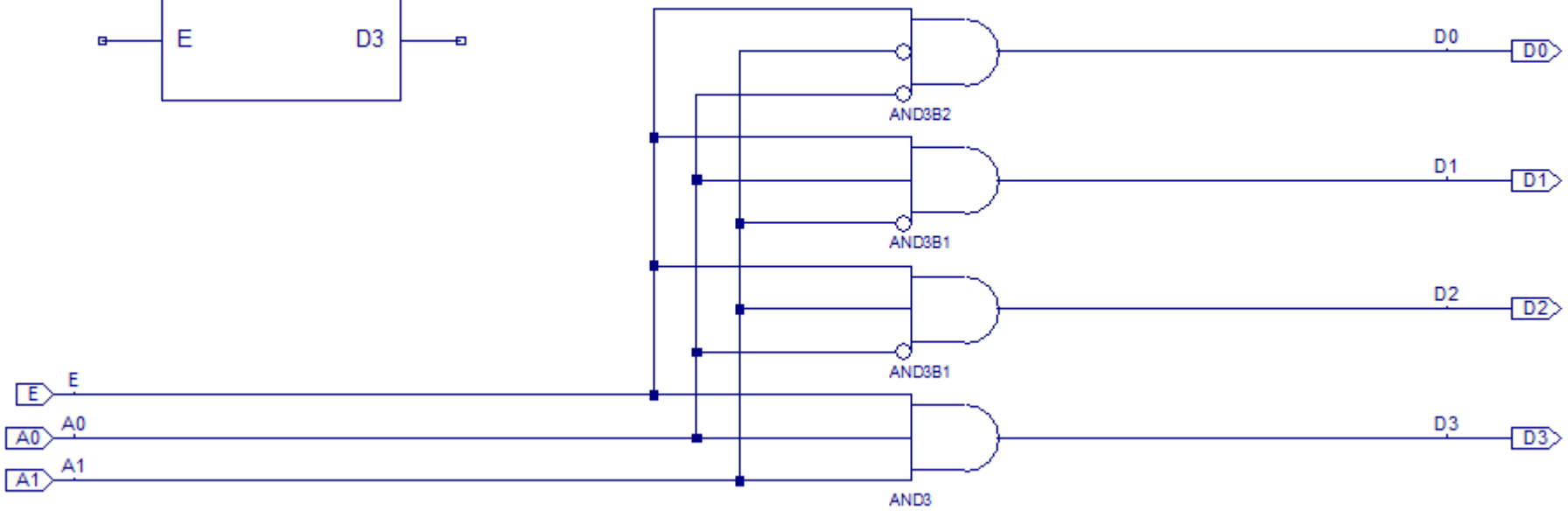
- Ieşirea y_i este activă, unde i este reprezentarea în binar a intrării x
  - DCD_3: este activă ieşirea y_5 dacă s-a primit la intrare x=(1,0,1)
  - EDCD (DCD_1): Ieşirea y_1 devine activă dacă s-a primit la intrare $x = 1$

Decodificatoare

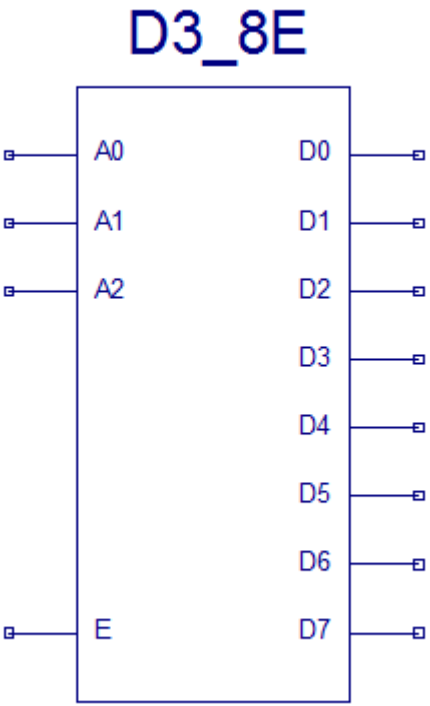


DCD_2

- ✓ 2 intrări + 1 (on/off)
- ✓ 4 ieşiri

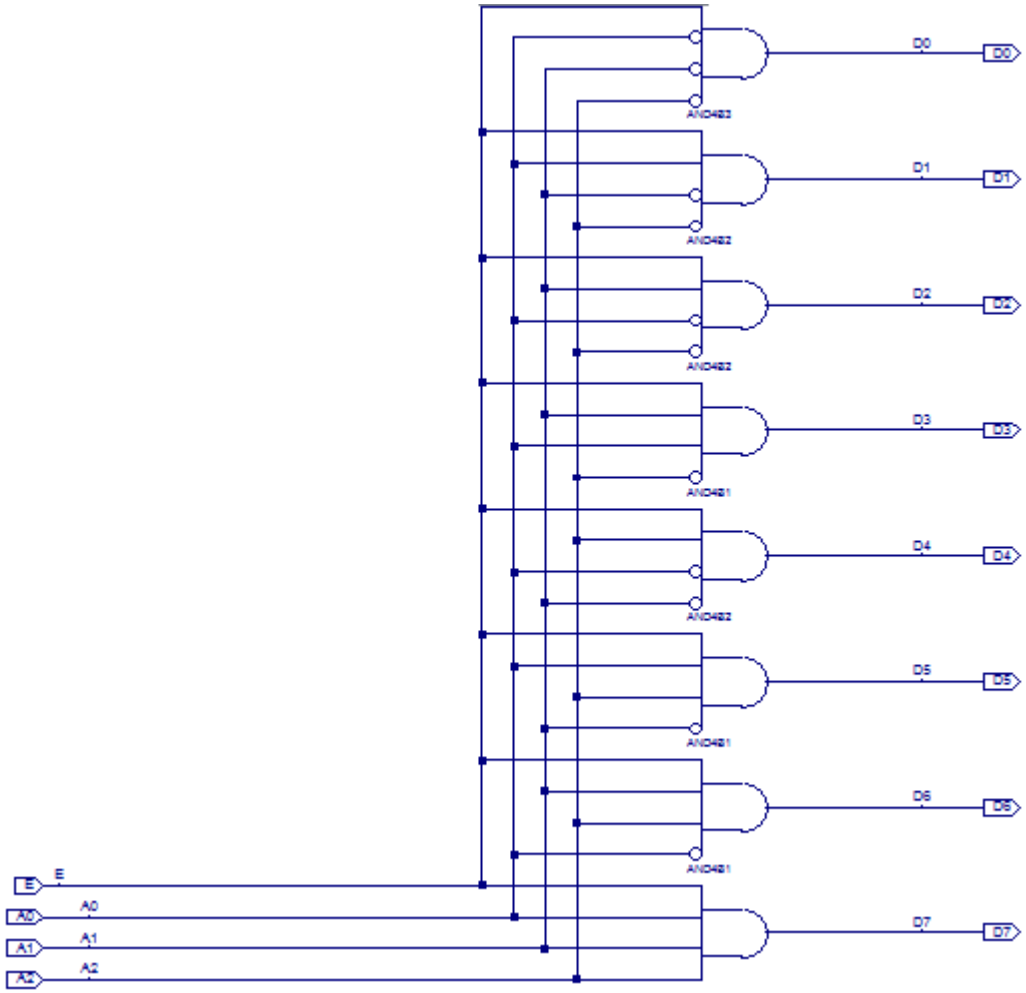


Decodificatoare



$\boldsymbol{DCD_3}$

- ✓ 3 intrări + 1 (on/off)
- ✓ 8 ieşiri



[Xilinx - ISE]

Decodificatoare

- DCD se pot defini recursiv:
  - ✓ DCD_1 este EDCD
  - ✓ DCD_n este compunerea dintre DCD_{n-1} şi DCD_1

- *Întrebare:* Care este construcţia?

- Construcţia anterioară introduce o complexitate exponenţială, aşa încât se preferă construcţia directă, cu n intrări în poarta AND

Codificatoare

- *Scop:* rezolvarea unei probleme; generarea unui rezultat

- *Mod de funcţionare*: intrările în codificator sunt ieşirile unui decodificator; apoi, folosind porţi **OR (SAU)** se implementează funcţia în **forma canonică** (**normal disjunctivă**)

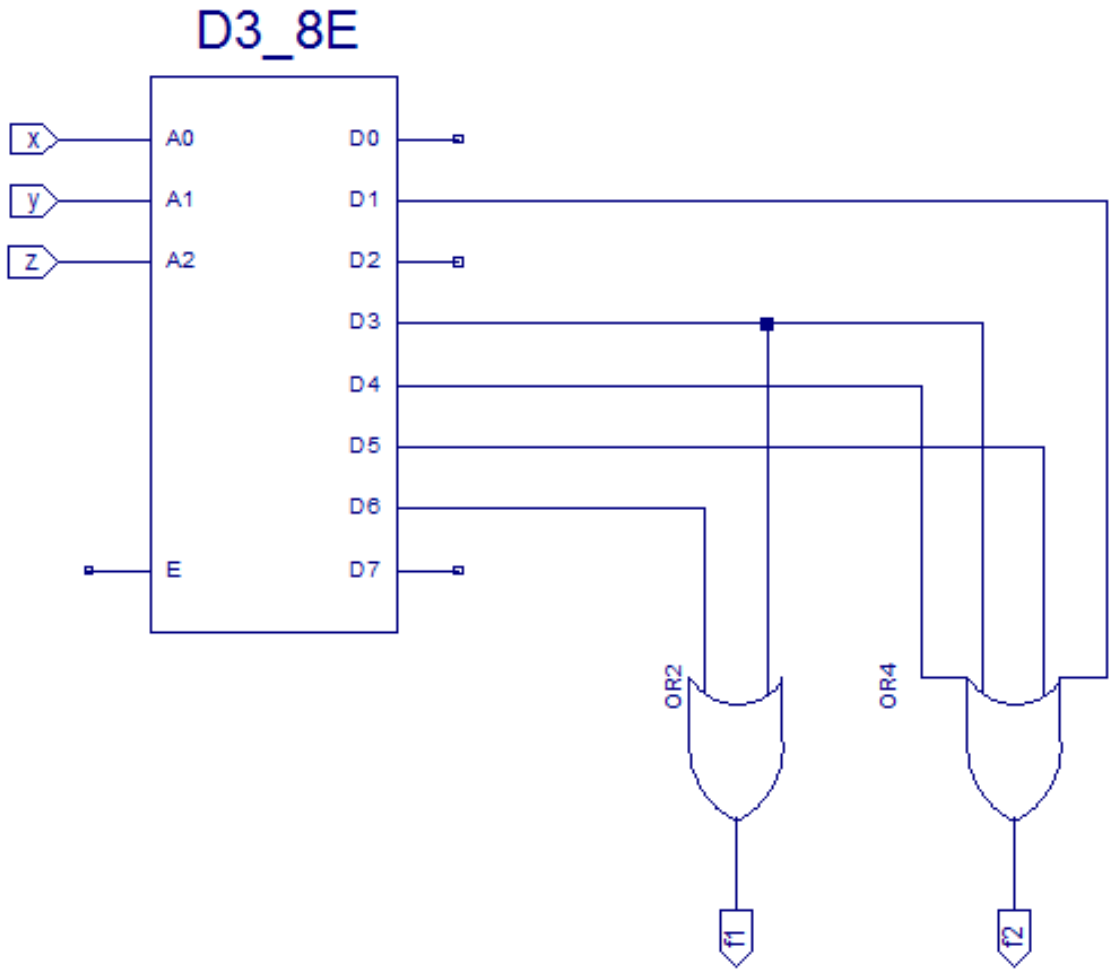
Codificatoare

- *Def.:* Un $(2^n, p)$-*codificator* este un circuit combinational cu 2^n intrări dintre care una singură este activă la un anumit moment şi p porţi OR, fiecare cu maxim 2^n intrări distincte.

- Construcţie:
  - Intrările în codificator sunt ieşirile unui decodificator DCD_n
  - Se implementează funcţionalitatea cu ajutorul porţilor OR (prin scriere în formă normal disjunctivă)

- Universalitate:
  - Orice funcţie logică poate fi implementată cu ajutorului unui codificator

Codificatoare



[Xilinx - ISE]

Codificatoare

- *Întrebare:* Care sunt formele canonice pentru funcţiile corespunzătoare circuitului reprezentat pe slide-ul anterior?

- *Răspuns:*

FND pentru cele 2 ieşiri sunt:

$$f_1(x, y, z) = xy\bar{z} + \bar{x}yz$$

$$f_2(x, y, z) = x\bar{y}\bar{z} + xy\bar{z} + \bar{x}\bar{y}z + x\bar{y}\text{z}$$

Unde:

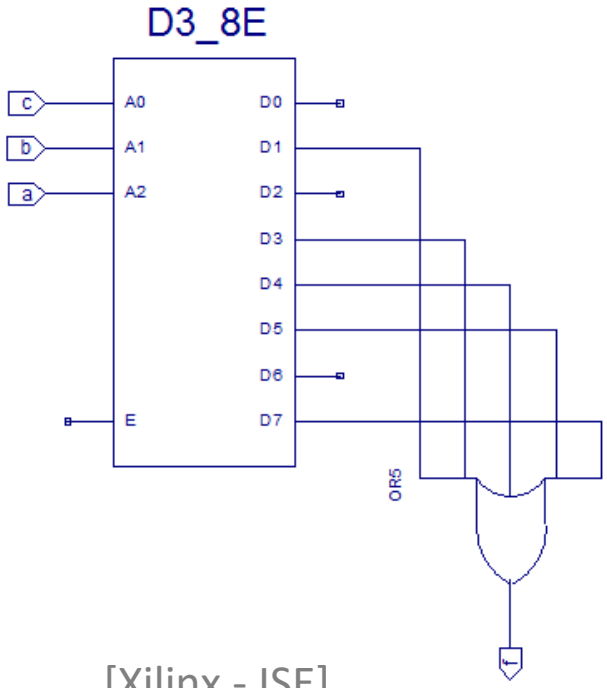
$$f(x, y, z) = (f_1(x, y, z), f_2(x, y, z))$$

Codificatoare

- *Întrebare:* Cum se reprezintă cu ajutorul unui codificator funcţia următoare?

$$f(a,b,c) = \bar{a}c + a\bar{b} + abc$$

- *Răspuns:*



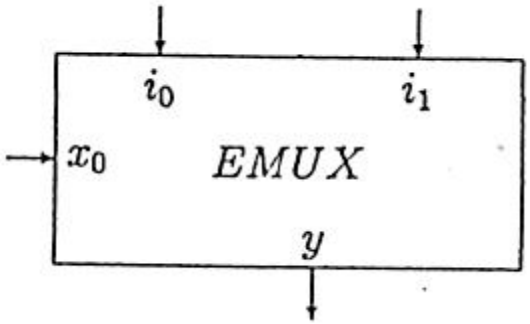
[Xilinx - ISE]

Se aduce funcţia la FND:

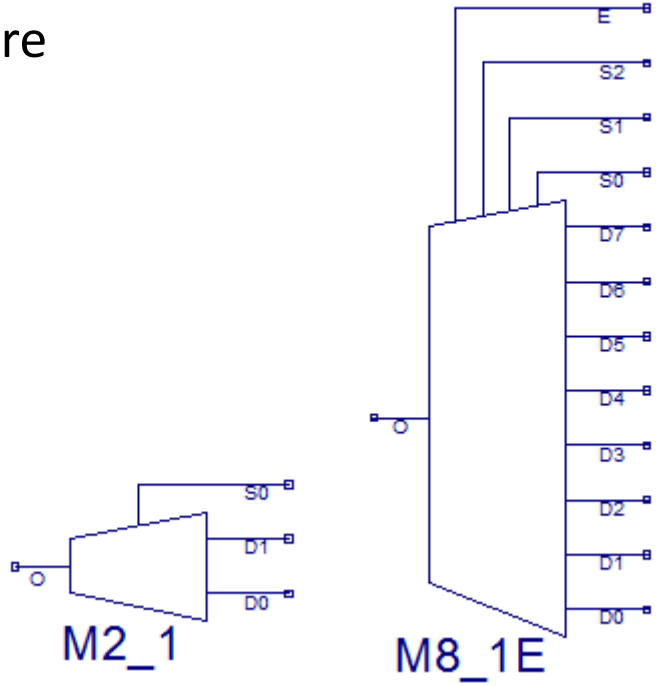
$$f(a,b,c) = \bar{a}\bar{b}c + \bar{a}bc + a\bar{b}\bar{c} + a\bar{b}c + abc$$

Multiplexoare

- *Scop:* selectează unul dintre semnalele primite la intrare în funcţie de un semnal de selecţie
- *Abreviere:* MUX (multiplexor); EMUX (multiplexor elementar)
- *Mod de funcţionare*: o singură ieşire este activă la un moment dat; aceasta determină semnalul de intrare
- *Reprezentare schematică:*



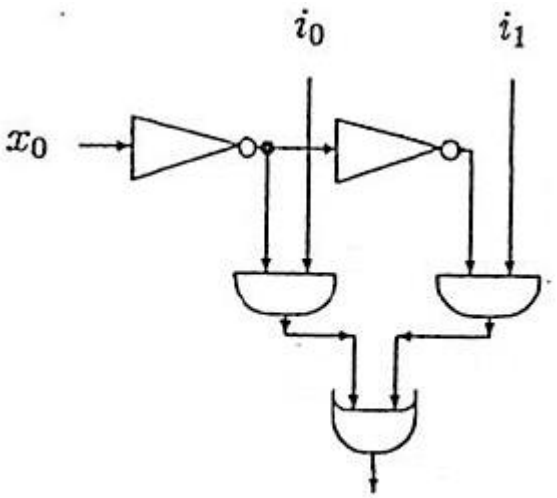
[AAT]



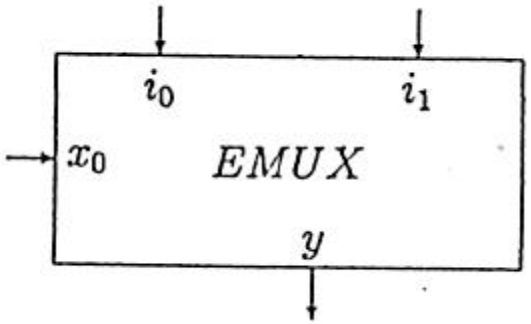
[Xilinx - ISE]

Multiplexoare

- *EMUX:* 2 intrări şi 1 bit de selecţie; 1 ieşire
  - ✓ Ieşirea copiază intrarea 0 dacă bitul de selecţie este 0;
  - ✓ Ieşirea copiază intrarea 1 dacă bitul de selecţie este 1;



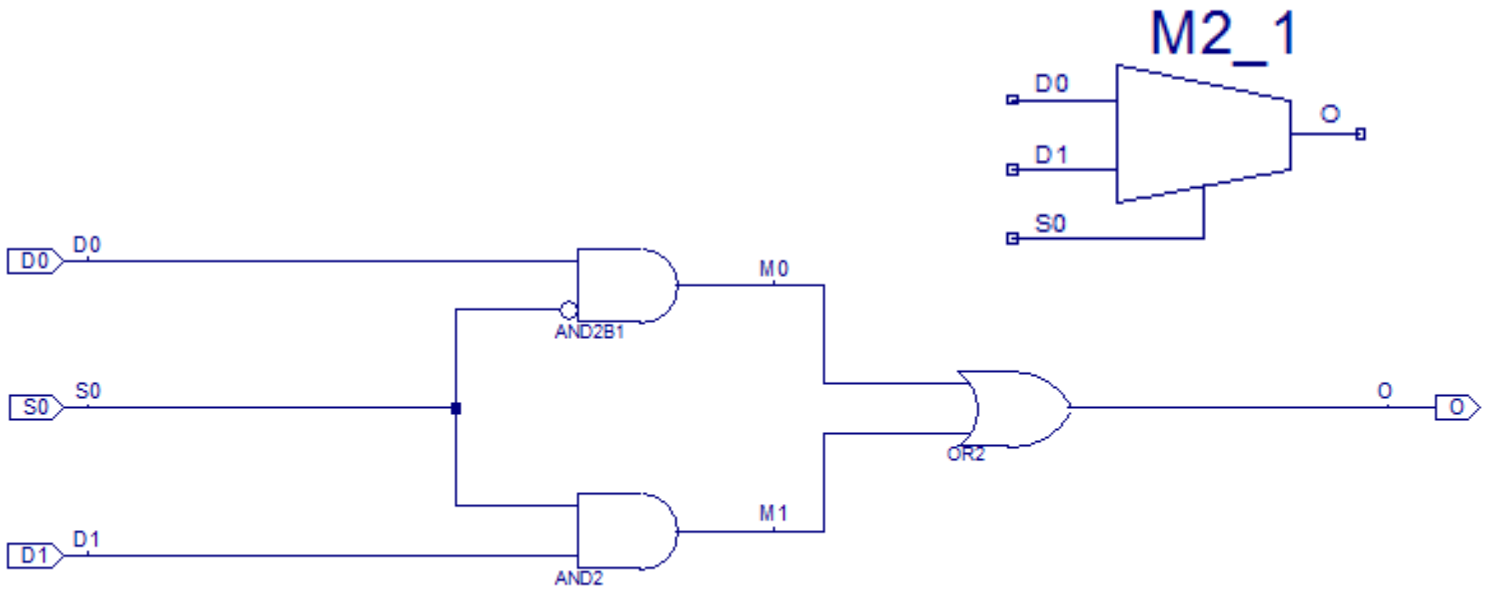
$$y = x_0 i_1 + \overline{x_0} i_0$$



[AAT]

Multiplexoare

- *EMUX:* 2 intrări şi 1 bit de selecţie; 1 ieşire
  - ✓ Ieşirea copiază intrarea 0 dacă bitul de selecţie este 0;
  - ✓ Ieşirea copiază intrarea 1 dacă bitul de selecţie este 1;



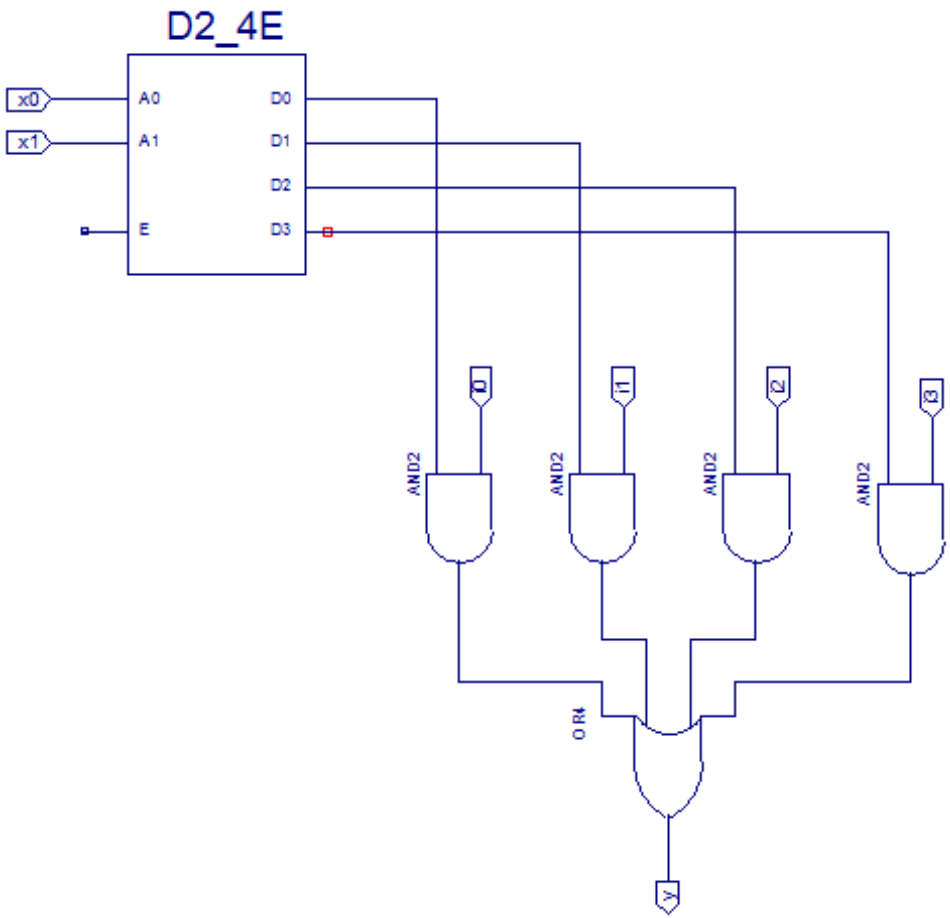
Multiplexoare

- *Def.:* Un MUX_n este un circuit combinational cu n biţi de control care selectează la ieşire o singură intrare din maxim 2^n posibile.

- Intrarea i_j este activă, unde j este reprezentarea în binar a semnalului de selecţie

Multiplexoare

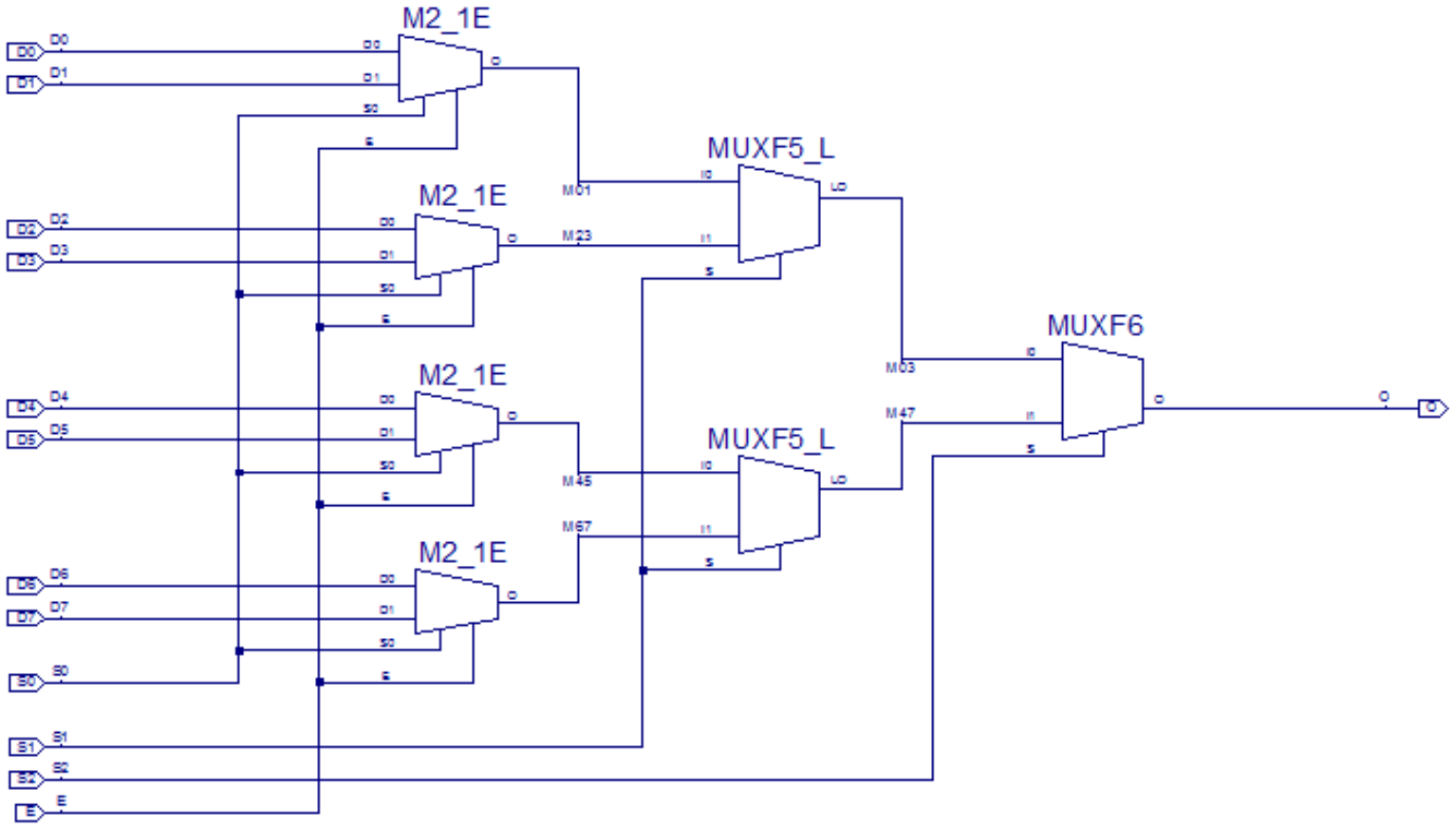
- Construcţia cu ajutorul decodificatorului:
  - Intrarea i_j este activă, unde j este reprezentarea în binar a semnalului de selecţie



[Xilinx - ISE]

Multiplexoare

- Construcţia recursivă (prin conectări seriale şi paralele):



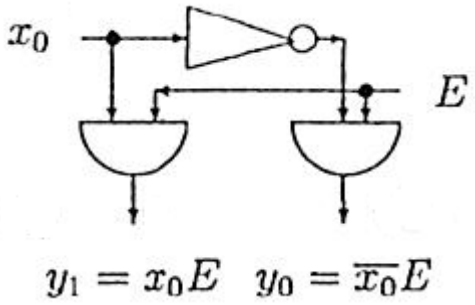
Demultiplexoare

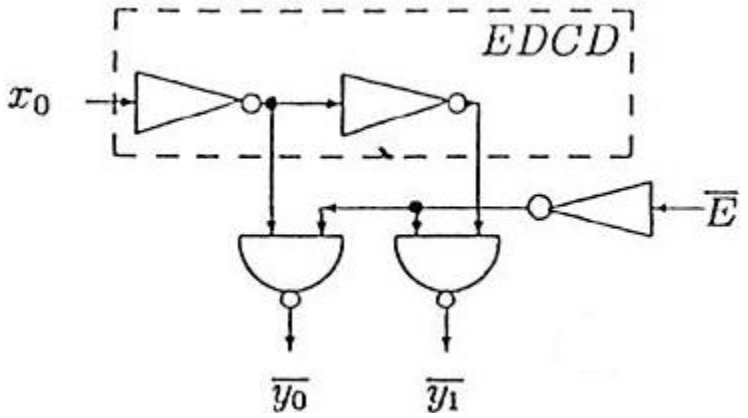
- *Scop:* transferă un semnal de intrare la o ieşire în funcţie de un cod de selecţie
- *Abreviere:* DMUX (demultiplexor); EDMUX (demultiplexor elementar)
- *Mod de funcţionare*: o singură ieşire a unui decodificator este activă la un moment dat; aceasta va fi cea care va prelua semnalul de intrare

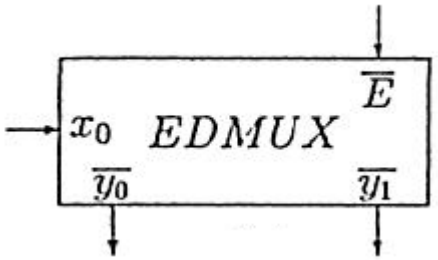
Demultiplexoare

- *EDMUX:* 1 intrare şi 1 bit de selecţie; 2 ieşiri

  - ✓ Ieşirea y_0 devine E dacă bitul de selecţie are valoarea 0
  - ✓ Ieşirea y_1 devine E dacă bitul de selecţie are valoarea 1





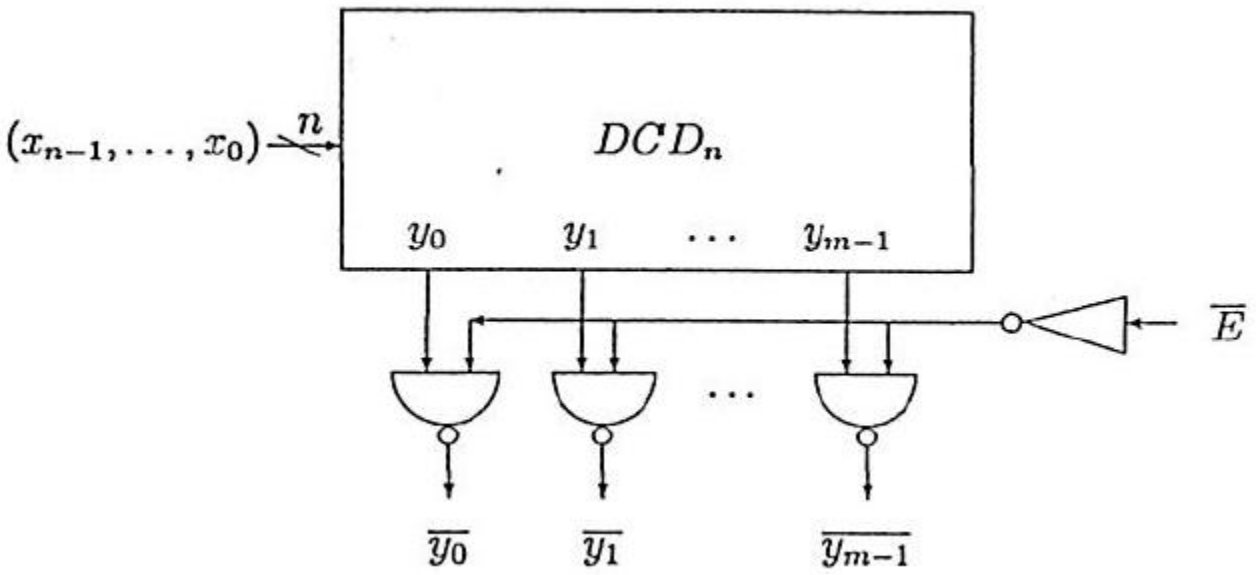


[AAT]

Demultiplexoare

- *Def.:* Un demultiplexor $DMUX_n$ cu n intrări transferă semnalul de la intrare la una dintre cele ieşiri conform cu codul binar de selecţie.

- Construcţia cu ajutorul decodificatorului:



[AAT]

PLA (Programable Logic Array)

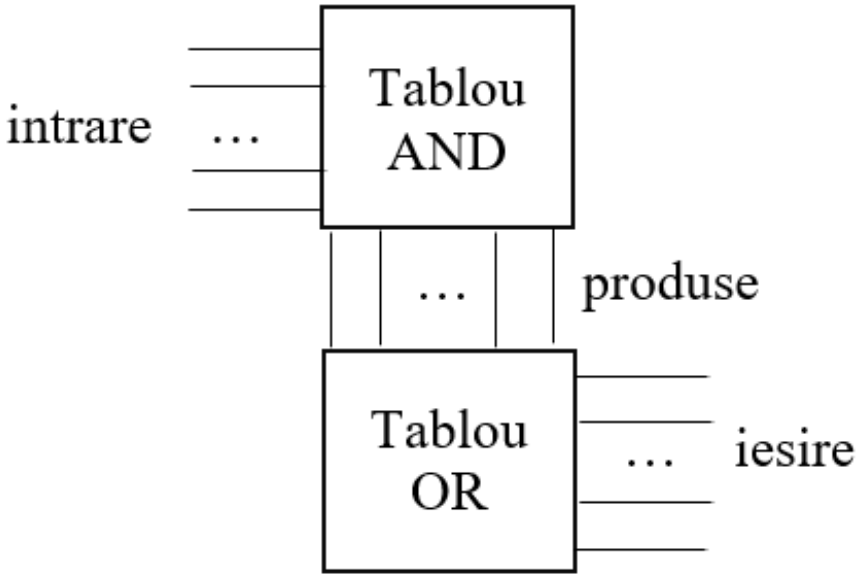
- Am arătat deja că orice funcţie logică poate fi codificată folosind ieşirile unui decodificator, prin aducere la forma normal disjunctivă

- Această construcţie nu este însă întotdeauna optimă pentru că se utilizează un decodificator complet, chiar dacă spre exemplu nu este necesară decât utilizarea câtorva produse de termeni

- *PLA (Programmable Logic Array)* implementează mai eficient o funcţie booleană, prin 2 nivele de porţi logice: un **tablou AND** şi un **tablou OR** (care conţin doar produsele utilizate)

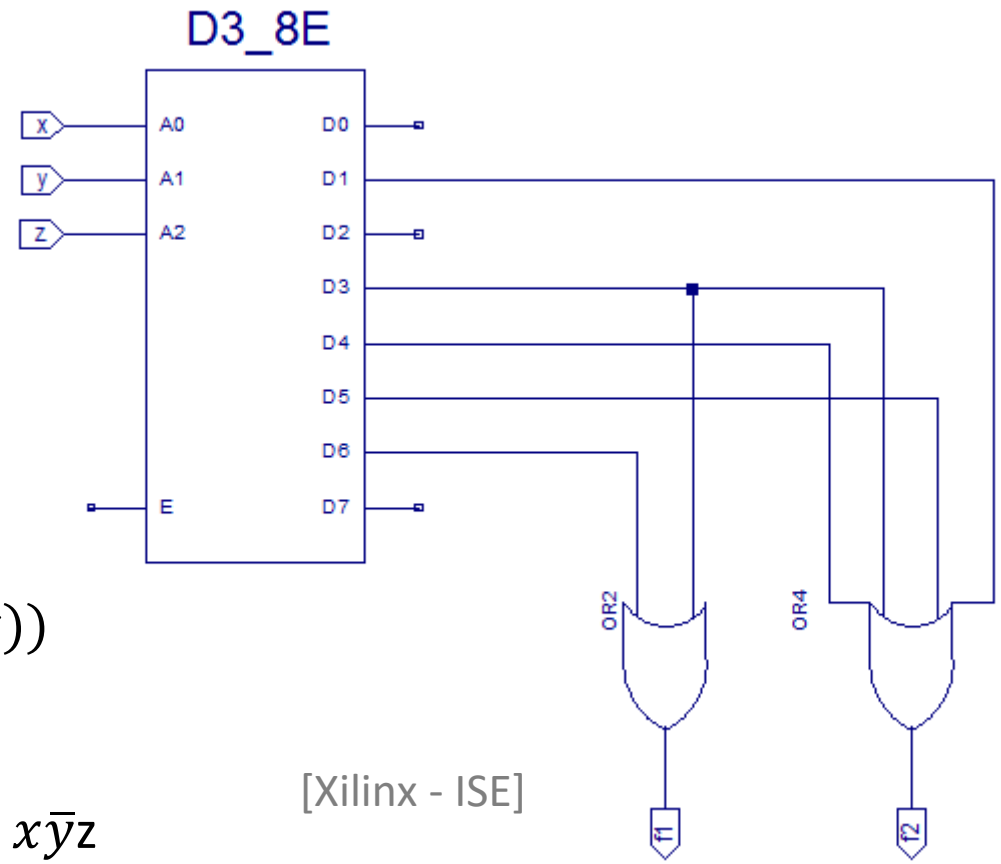
PLA (Programable Logic Array)

- Reprezentarea generală unui PLA este următoarea:



PLA (Programable Logic Array)

- Am codificat anterior cu ajutorul unui decodificator funcţia



$$f(x,y,z) = (f_1(x,y,z), f_2(x,y,z))$$

$$f_1(x,y,z) = xy\bar{z} + \bar{x}yz$$

$$f_2(x,y,z) = x\bar{y}\bar{z} + xy\bar{z} + \bar{x}\bar{y}z + x\bar{y}z$$

[Xilinx - ISE]

PLA (Programable Logic Array)

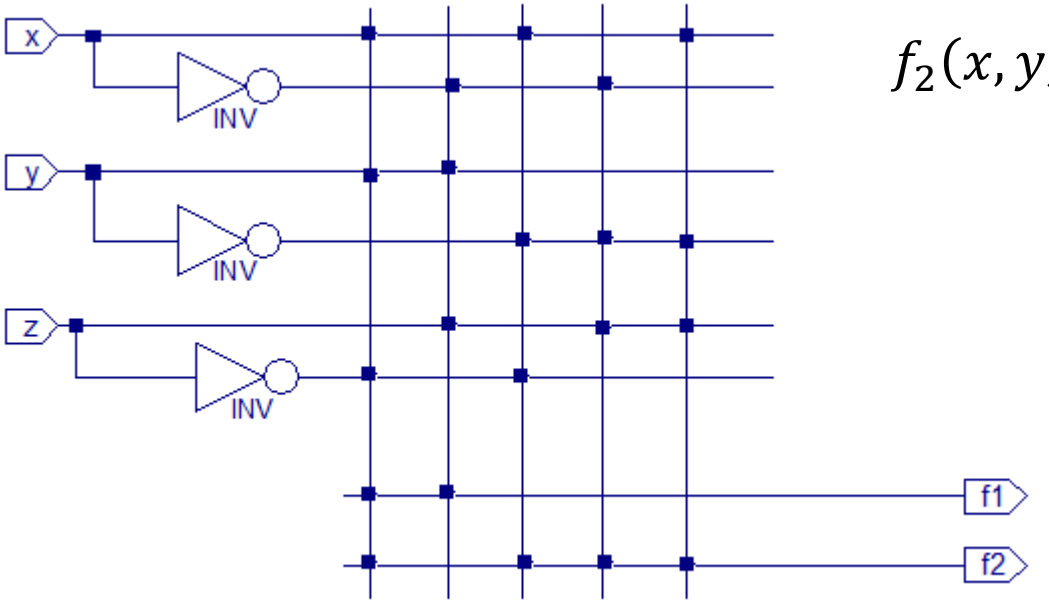
- *Întrebare:* Cum se reprezintă aceeaşi funcţie printr-un PLA?

- *Răspuns:*

$$f(x,y,z) = (f_1(x,y,z), f_2(x,y,z))$$

$$f_1(x,y,z) = xy\bar{z} + \bar{x}yz$$

$$f_2(x,y,z) = x\bar{y}\bar{z} + xy\bar{z} + \bar{x}\bar{y}z + x\bar{y}z$$



ROM (Read Only Memory)

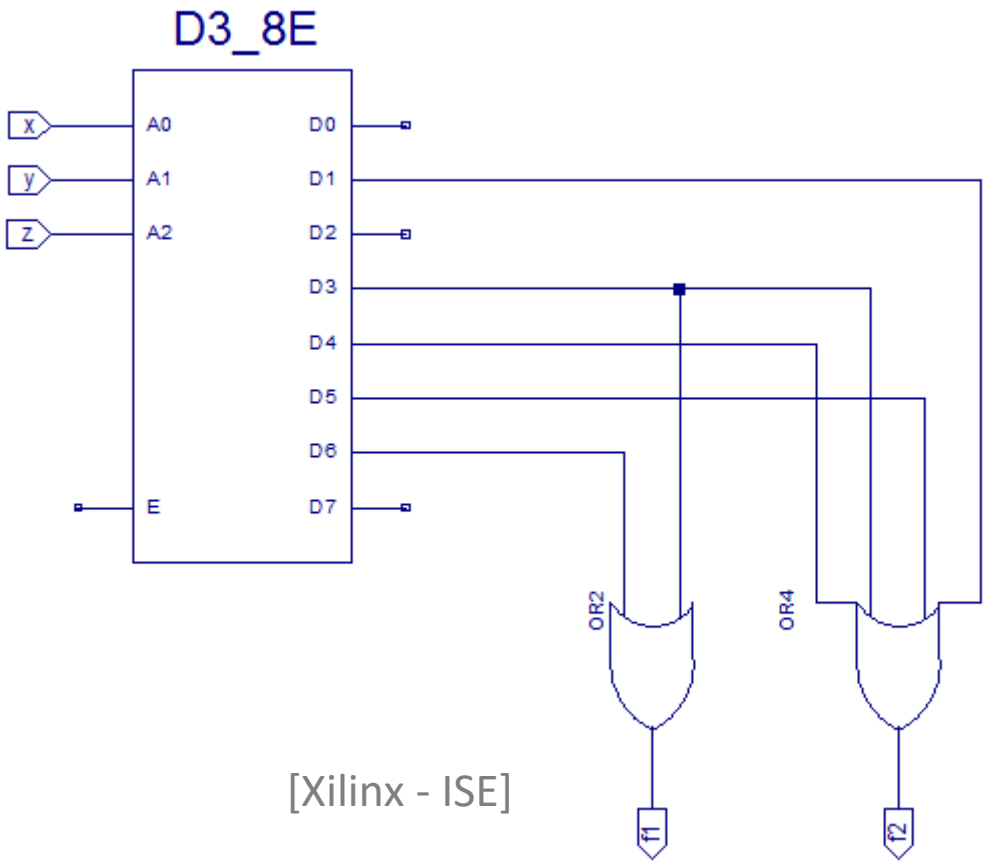
- Codificarea unei funcţii cu ajutorul unui decodificator poate fi interpretată şi ca o memorie fixă

- *Read Only Memory (ROM)* este un tip de memorie care se poate citi, dar nu se poate modifica

- Dacă se consideră intrările în codificator ca adrese, atunci ieşirea circuitului reprezintă valoarea stocată la adresa respectivă

ROM (Read Only Memory)

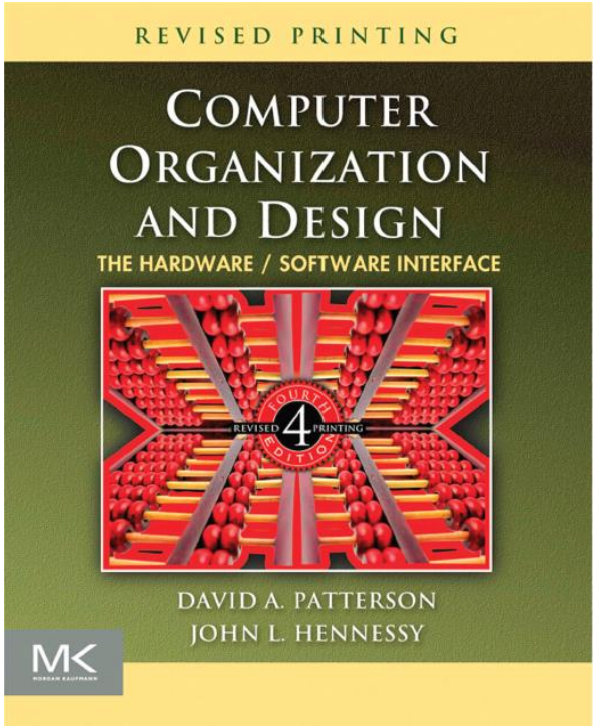
- *Întrebare:* Care este valoarea stocată la adresa 010? Dar la adresa 110?



- *Răspuns:* 00, respectiv 10

[Xilinx - ISE]

Referințe bibliografice



[COD] D. Patterson and J. Hennessy, Computer Organisation and Design

[AAT] A. Atanasiu, Arhitectura calculatorului

Schemele [Xilinx - ISE] au fost realizate folosind

http://www.xilinx.com/tools/projnav.htm