

PROIECTAREA DISPOZITIVELOR NUMERICE CU CIRCUITE LOGICE PROGRAMABILE



CIRCUITE PROGRAMMABILE

Grad de integrare

- **LSI** - Large Scale Integration
 - **PLA** - Programmable Logic Array
 - **PLD** - Programmable Logic Device
- **VLSI** - Very Large Scale Integration
 - **ASIC** - Application Specific Integrated Circuit
 - **CPLD** - Complex Programmable Logic Device
 - **FPGA** - Field Programmable Gate Array

PROIECTAREA CLASICĂ

Etapele fluxului de proiectare

- se începe cu o *specificație*
- se construiește o *diagramă bloc*
- se separă secțiunile organigramei, după care se *detaliază* fiecare până când se atinge nivelul corect al proiectului logic
- se *integrează piesele*
 - în caz că există un produs software dezvoltat anume pentru gestionarea sistemului, se va utiliza în această etapă

PROIECTAREA CLASICĂ

Etapele fluxului de proiectare

- se creează *prototipul*, care este depanat și corectat cu ajutorul software-ului
 - adeseori, prototipul nu funcționează la viteza proiectată și trebuie reexaminat pentru diverse corecții (de exemplu gâturi - *bottlenecks*)
- se realizează fizic sistemul pe placă (**PCB** - *Printed Circuit Board*)
 - și aici apar de multe ori corecții care trebuie efectuate, aspecte impuse de condițiile fizice de realizare a PCB-ului

PROIECTAREA CLASICĂ

- Revenirile multiple în procesul de proiectare clasică constituie regula, nu excepția
- Procesul de proiectare clasică - **costisitor din punct de vedere al timpului**

PROIECTAREA CU DISPOZITIVE PROGRAMABILE

Etapele fluxului de proiectare

- se pornește și aici tot de la *specificație*
- sistemul este apoi partiționat în *blocuri mari*
 - memorii
 - microprocesoare
 - PLD
 - CPLD
 - FPGA
 - + logică de interfațare

PROIECTAREA CU DISPOZITIVE PROGRAMABILE

Etapele fluxului de proiectare

- se formulează o *descriere de nivel înalt* a sistemului
 - cu un editor schematic
 - cu un limbaj de descriere hardware abstract (de exemplu VHDL sau Verilog)
- întregul sistem este *simulat* - se înlocuiește astfel vechea fază de prototip

PROIECTAREA CU DISPOZITIVE PROGRAMABILE

Etapele fluxului de proiectare

- se creează lista de componente și legăturile dintre ele (*netlist*) a sistemului
- netlista este folosită la *realizarea PCB*-ului
- în timpul realizării fizice a PCB-ului, *simularea* mai este *rafinată*

PROIECTAREA CU DISPOZITIVE PROGRAMABILE

- cele mai multe modificări apar în software, în PLD-uri sau în FPGA-uri, nu în interconexiuni sau în componente secundare
- **timpul foarte rapid** de la conceperea abstractă a unui proiect până la realizarea sa efectivă

COMPLETITUDINE FUNCȚIONALĂ

- **Concept de bază** în înțelegerea dispozitivelor numerice programabile
- *orice funcție booleană* poate fi realizată pornind de la o *sumă de produse*
- *completitudine funcțională*: proprietatea unui singur tip de poartă logică care este capabilă de a forma o sumă de produse
- *orice funcție booleană* poate fi realizată folosind **NUMAI** acel tip de poartă logică

COMPLETITUDINE FUNCȚIONALĂ

■ Calitățile esențiale pentru completitudinea funcțională

- cu ajutorul porții respective să poată fi construită funcția logică **ȘI**
- cu ajutorul porții respective să poată fi construită funcția logică **NU**

sau

- cu ajutorul porții respective să poată fi construită funcția logică **SAU**
- cu ajutorul porții respective să poată fi construită funcția logică **NU**

FUNCȚII UNIVERSALE

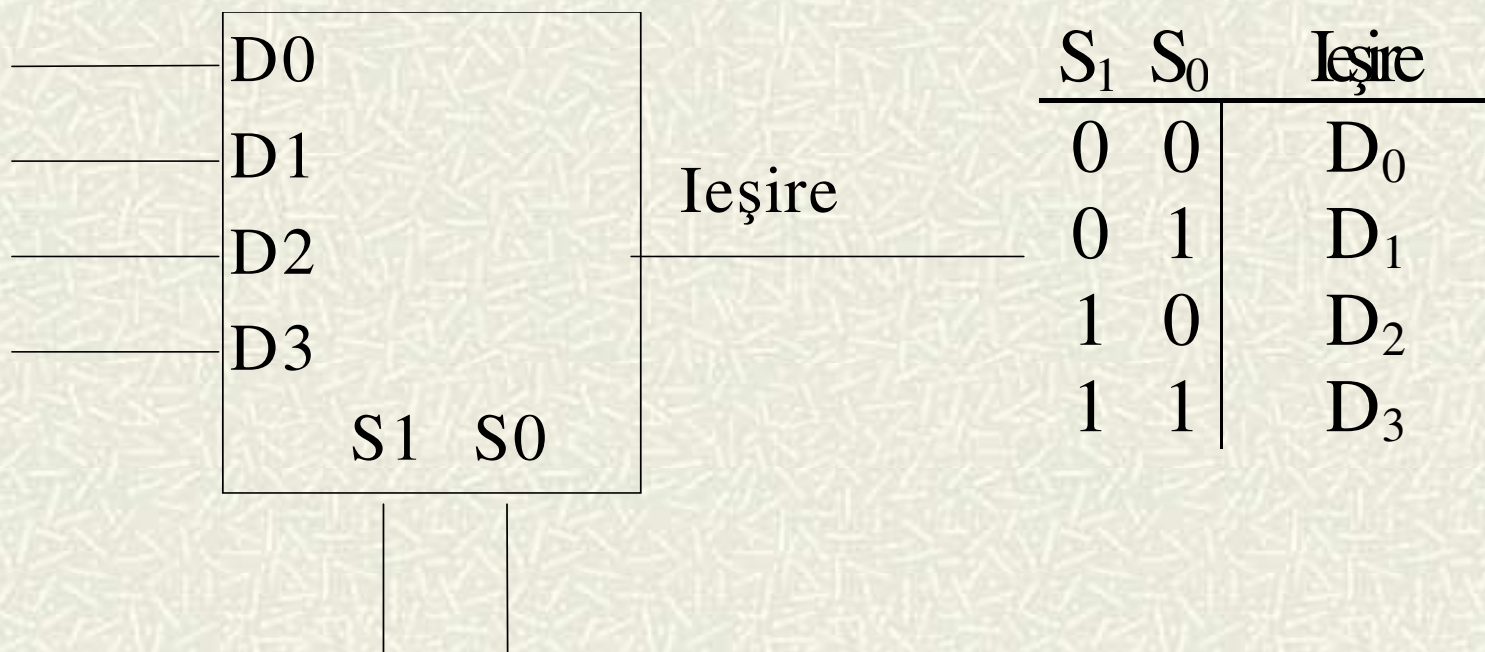
- *Funcțiile universale* sau generatoarele de funcții sunt *blocuri logice* care pot fi configurate astfel încât să realizeze orice funcție logică de intrările blocului

FUNCTII UNIVERSALE

- **tipuri** de astfel de blocuri logice:
 - memorii
 - multiplexoare
- toate aceste blocuri logice pot realiza funcții booleene formând tabelele lor de adevăr

FUNCTII UNIVERSALE

■ Exemplu: multiplexor 4:1



FUNCTII UNIVERSALE

- Ecuația ieșirii multiplexorului:

- $leșire = \bar{S}_1 \cdot \bar{S}_0 \cdot D_0 + \bar{S}_1 \cdot S_0 \cdot D_1 + S_1 \cdot \bar{S}_0 \cdot D_2 + S_1 \cdot S_0 \cdot D_3$

- dacă $D_0 = D_1 = D_2 = 0$ și $D_3 = 1$, atunci funcția $leșire = S_1 \cdot S_0 \rightarrow$ **ȘI logic**
- dacă $D_0 = 0$ și $D_1 = D_2 = D_3 = 1$, atunci funcția $leșire = S_1 + S_0 \rightarrow$ **SAU logic**
- dacă $D_0 = D_2 = 1$ și $D_1 = D_3 = 0$, atunci funcția $leșire = \bar{S}_0 \rightarrow$ **NU logic**

FUNCTII UNIVERSALE


■ Multiplexorul

- **universal** - poate forma orice funcție de cele două variabile de intrare S_0 și S_1 (selecțiile) prin setarea valorilor D din tabelul de adevăr la 0 sau la 1 logic
- **complet funcțional** - poate forma funcțiile:
 - **ȘI** și **NU** sau
 - **SAU** și **NU**

CELULE LOGICE

- O celulă cu proprietatea de completitudine funcțională

- O celulă cu funcție logică universală



pot realiza orice funcție logică combinațională folosind **una** sau mai **multe copii** ale lor

Exemplu - celulă logică

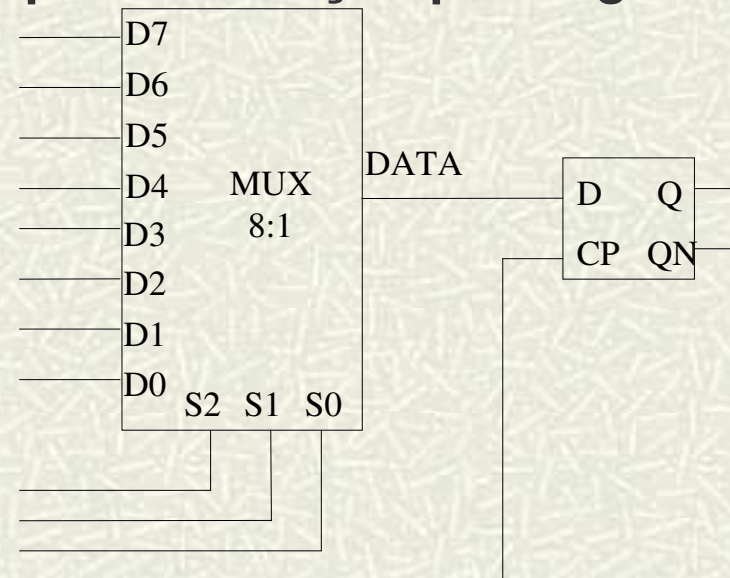
- **Exemplu:** celulă logică universală realizată cu multiplexoare
 - se utilizează MUX de tip 8:1



Exemplu - celulă logică

- celulă logică numai cu MUX 8:1 \Rightarrow bistabilele trebuie implementate tot cu MUX \Rightarrow prețul mult prea mare
- uzual, celula universală (sau funcțional completă) \rightarrow **celulă hibridă** \Rightarrow are ieșirea legată direct la intrarea unui bistabil D (care basculează pe front)

Multiplexor cu ieșire prin registru

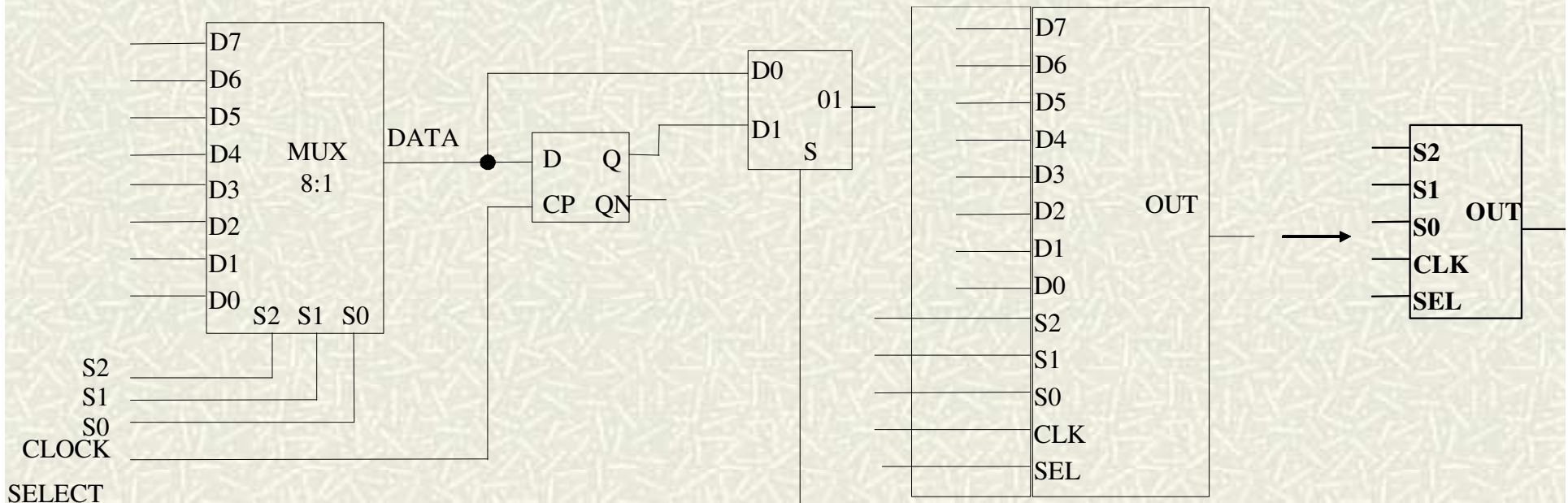


Exemplu - celulă logică

- celulă hibridă → **problemă**: toate funcțiile combinaționale sunt obligate să folosească și bistabile D
- **soluție** → se adaugă încă un MUX, de tipul 2:1, care va comanda direct intrarea D a bistabilului sau îl va ocoli pentru a "ieși" în exterior

Exemplu - celulă logică

- structura blocului constructiv poate forma:
 - orice funcție combinațională de trei variabile
 - funcții secvențiale realizate cu bistabile D
- intrările de date nu sunt scoase în afara celulei - număr de conexiuni externe redus



Celula logică constructivă de bază

20.12.2019

Simbolul celulei de bază

21

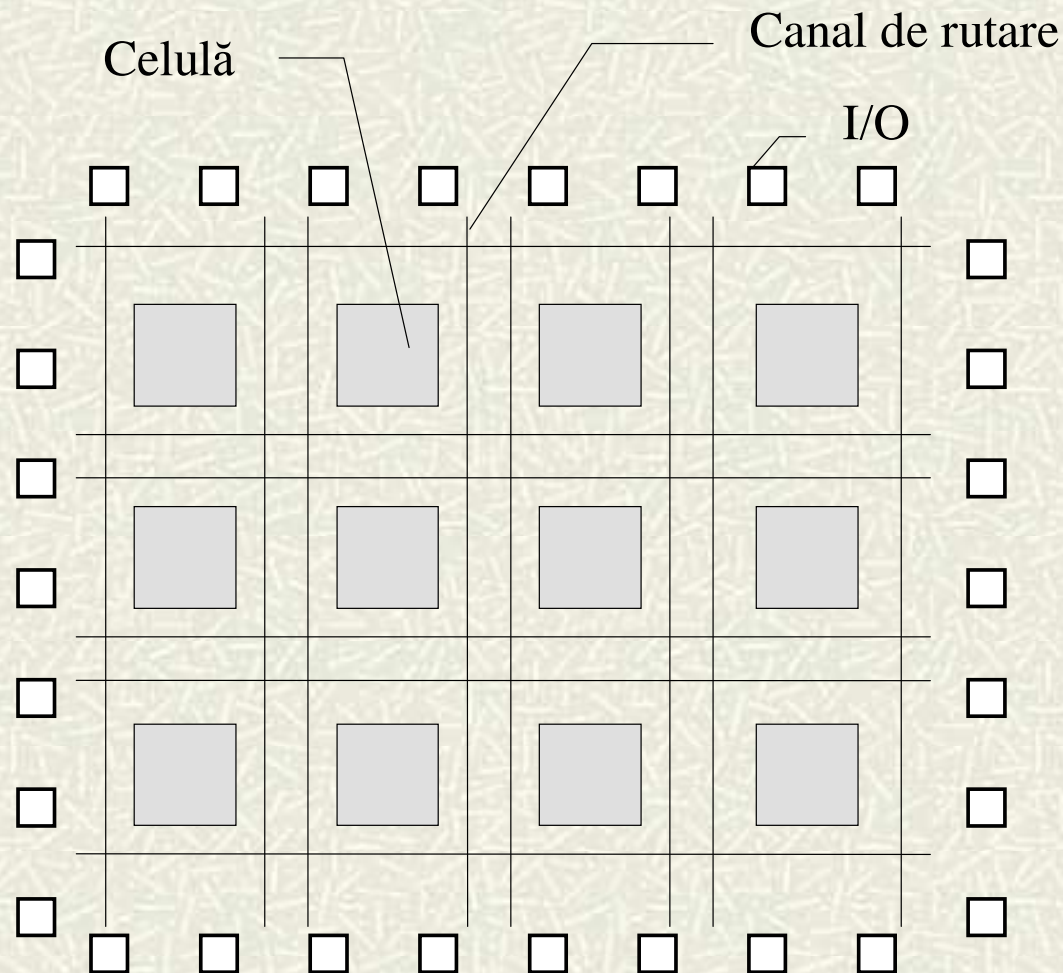
Exemplu - celulă logică

■ funcționare:

- pentru legături între celule → canale de rutare
- semnalele externe intră prin **buffer de intrare** și ajung la liniile verticale și orizontale
- în toate punctele de intersecție a unei linii orizontale cu una verticală se poate realiza o **conexiune**
- valorile logice de pe liniile de intrare de date ale multiplexoarelor sunt setate prin intermediul unor circuite de programare → **funcția** fiecărei celule este **programată intern**
- ieșirea multiplexoarelor este rutată înspre mai multe linii verticale și orizontale, la intrările altor celule sau la ieșirile circuitului
- ieșirile circuitului sunt realizate prin **buffer de ieșire**

ARHITECTURI DE DISPOZITIVE NUMERICE PROGRAMABILE

■ Arhitectură cu canale de rutare



ARHITECTURI DE DISPOZITIVE NUMERICE PROGRAMABILE

■ Arhitectură **foldback** (cu reacție inversă)

