# Arhitectura sistemelor de calcul

## - Prelegerea 9 -

## 1-DS (Memorii)

Ruxandra F. Olimid

Facultatea de Matematică şi Informatică
Universitatea din Bucureşti

# Cuprins

# 1-DS (Memorii)

- Introducem un *ciclu* şi obţinem un prim grad de autonomie al sistemelor digitale: starea sistemului depinde numai parţial de semnalul de intrare

- Sistemele *1-DS* (cu un ciclu) pot păstra informaţia o perioadă determinată de timp, motiv pentru care se numesc şi *circuite de memorie* (sau mai simplu, *memorii*)

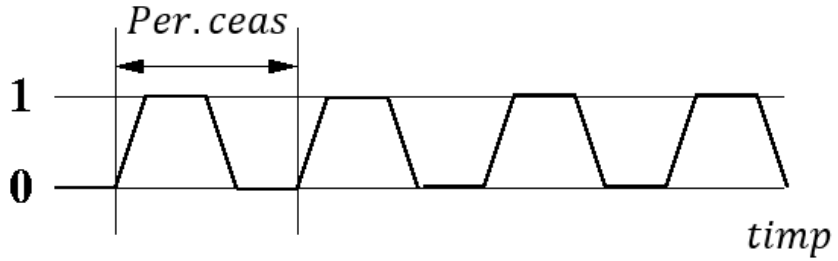
- Informaţia stocată este folosită pentru diferite calcule, operaţii, etc. în anumite faze de calcul bine determinate, fiind deci necesară o **sincronizare**

- Această sincronizare este posibilă cu ajutorul **ceasului**, un dispozitiv general de control al circuitelor

# Ceas

- Am discutat despre ceas când am studiat performanţa calculatoarelor
- *Ceasul* este caracterizat de:
  - ✓ *Perioada ceasului:* lungimea (durata) unui tact (a unui ciclu de ceas); unitate de măsură: *secunda*
  - ✓ *Frecvenţa ceasului:* câte cicluri de ceas se fac în unitatea de timp; unitate de măsură: *Hertz*

$$Per.ceas = \frac{1}{Frecv.ceas}$$




[Cazul real: trecerea de la 0 la 1 şi invers nu se realizează instantaneu]

# Ceas


[Cazul ideal: trecerea de la 0 la 1 şi invers se realizează instantaneu]

- Ceasul este un semnal oscilant între o valoare scăzută (*low* sau *0*) şi o valoare înaltă (*high* sau *1*)
- Ideal, trecerea de la un nivel la altul al ceasului se realizează instantaneu
- Practic acest lucru nu este posibil, fiind necesar un timp $t_{LH}$ pentru trecerea de la 0 (low) la 1 (high), respectiv un timp $t_{HL}$ pentru trecerea de la 1 (high) la 0 (low)
- Observaţi că timpul în care ceasul prezintă valoarea înaltă poate să difere de timpul în care ceasul prezintă o valoare scăzută

# Sisteme sincrone

- Sincronizarea circuitelor se face fie pe *frontul pozitiv* (trecerea de la 0 la 1), fie pe *frontul negativ* al ceasului (trecerea de la 1 la 0)

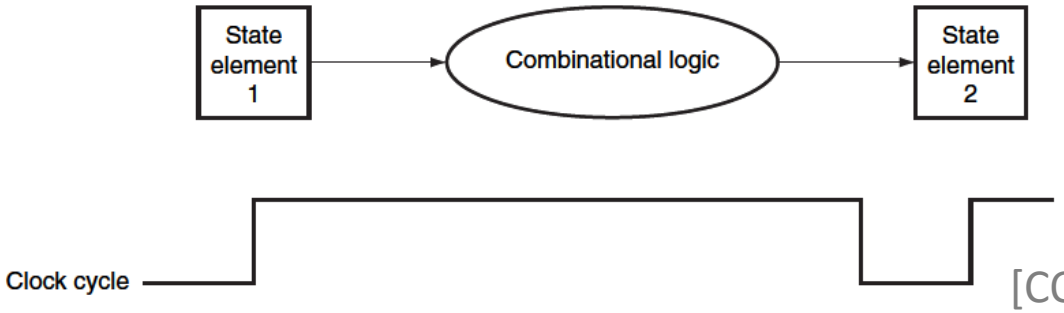
- Acesta se numeşte *front activ* şi poate conduce la schimbarea stării sistemului

- Un sistem care utilizează ceasul se numeşte *sistem sincron*; mai multe astfel de sisteme care rulează concomitent sunt *sincronizate*

- Pentru sistemele sincrone, semnalele trebuie să fie *valide* (i.e. **constante**) pe frontul activ

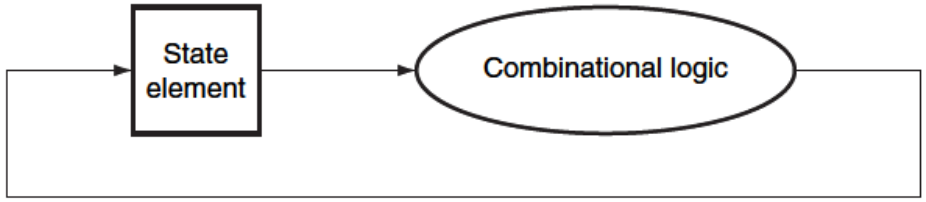
# Sisteme sincrone

- Pentru un sistem care conţine un element de memorie (1-DS) şi un circuit logic (0-DS), trebuie ca perioada ceasului să fie destul de lungă pentru a permite stabilizarea la introducerea semnalului în circuitul de memorie




[COD]

- Avantajul acestei construcţii este că semnalul rămâne activ şi dacă sistemul se închide printr-un ciclu




[COD]

# Cicluri stabile şi instabile

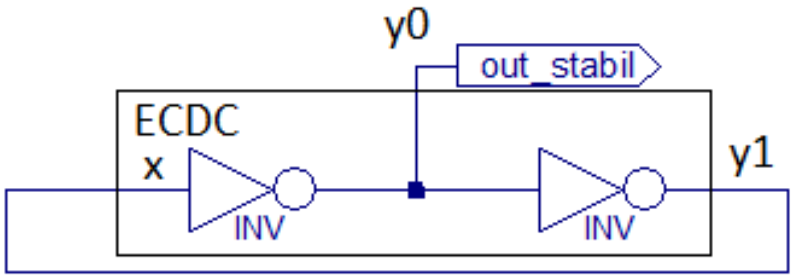
- Un *ciclu stabil*:
  - ✓ generează o **stare stabilă** (**constantă**)
  - ✓ se foloseşte pentru constructia **memoriilor**

- Un *ciclu instabil*:
  - ✓ generează o **stare instabilă** (**oscilantă**)
  - ✓ se foloseşte pentru construcţia **ceasului**

- Constructiv:
  - ✓ ciclul stabil prezintă un număr **par** de complementări ale intrării până la ieşire
  - ✓ ciclul instabil prezintă un număr **impar** de complementări ale intrării până la ieşire

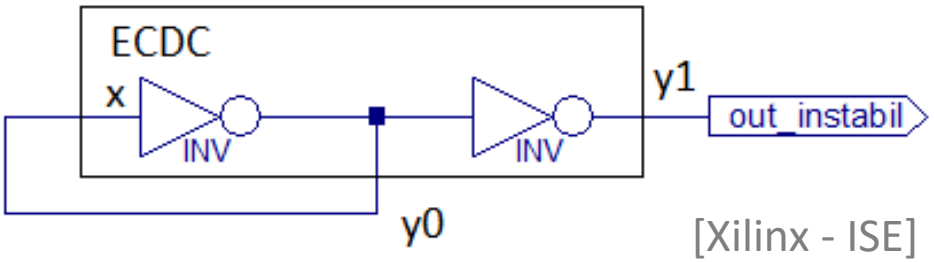
# Cicluri stabile şi instabile

- Un *ciclu stabil* se obţine imediat pornind de la un decodificator elementar (EDCD) şi formând un ciclu între intrare şi ieşirea 1




[Xilinx - ISE]

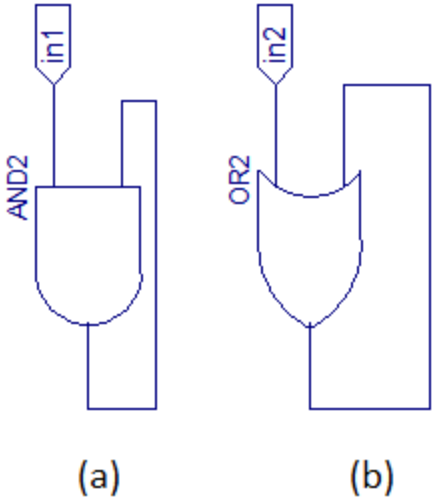
- Un *ciclu instabil* se obţine imediat pornind de la un decodificator elementar (EDCD) şi formând un ciclu între intrare şi ieşirea 0




[Xilinx - ISE]

# Zăvoare elementare

- Cele mai simple circuite stabile sunt următoarele, definite doar cu ajutorul unei singure porţi logice:




[Xilinx - ISE]

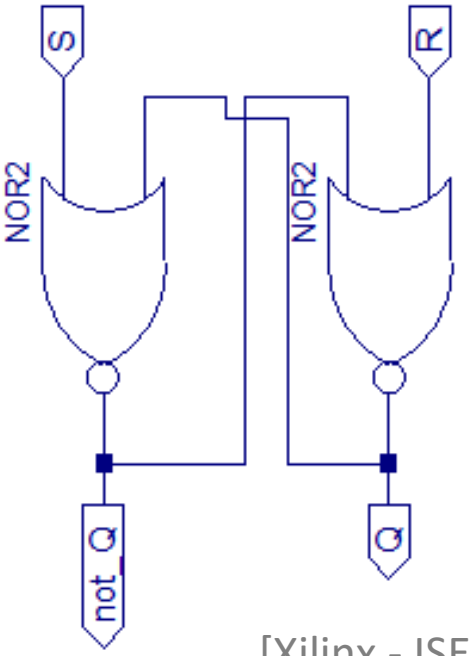
- *Întrebare:* În ce stare (0 sau 1) sunt fiecare dintre cele 2 circuite stabile?

- *Răspuns:*

  Circuitul (a) este stabil în starea 0 (se observă că dacă se introduce 0 la intrare o singură dată, circuitul va avea starea internă mereu 0, indiferent dacă se modifica intrarea în 1 sau nu).

  Analog, circuitul (b) este stabil în starea 1.

# Zăvoare elementare (S-R Latch)



[Xilinx - ISE]

- Combinând ambele funcţionalităţi într-un singur circuit, obţinem un *S-R Latch* (*Set-Reset Latch*) cu următoarea funcţionalitate:
  - ✓ *(Set)* dacă $S = 1, R = 0$ , atunci $Q = 1$
  - ✓ *(Reset)* dacă $S = 0, R = 1$ , atunci $Q = 0$

  [Am notat prin $\text{not } Q$ negarea lui $Q$]

- *Întrebare:* Ce se întâmplă când $S = 0, R = 0$ ? Dar când $S = 1, R = 1$ ?

- *Răspuns:*

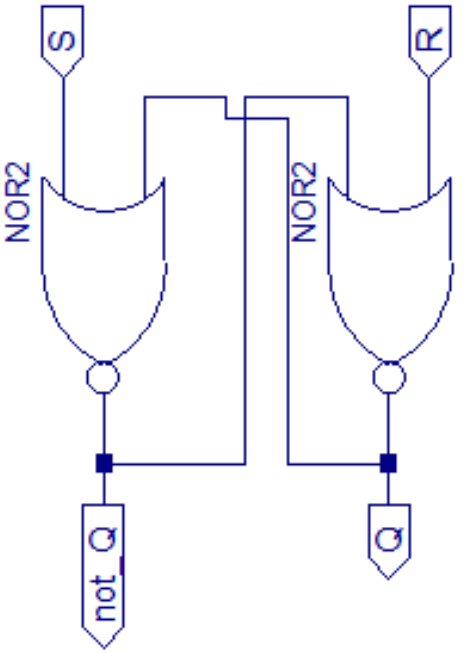
  Pentru $S = 0, R = 0$ se păstrează starea actuală a circuitului

  Pentru $S = 1, R = 1$ se ajunge într-o stare inconsistentă (de eroare)

  .

# Zăvoare elementare (S-R Latch)

- Limitări ale acestei construcţii:
  - ✓ Pentru $S = 1, R = 1$ , se ajunge într-o stare inconsistentă (de eroare)
  - ✓ Nu se poate determina intrarea care trebuie activată (set sau reset) pentru a schimba starea sistemului când aceasta nu se cunoaşte




[Xilinx - ISE]

- Pentru eliminarea acestor limitări introducem **o intrare de ceas** şi **o singură intrare de date D** ($S = D, R = \overline{D}$ )

# Latches. Flip-Flops

- Introducem sisteme cu intrare de ceas: *D-Latches* (*zăvoare de date*) şi *D-Flip-Flop*

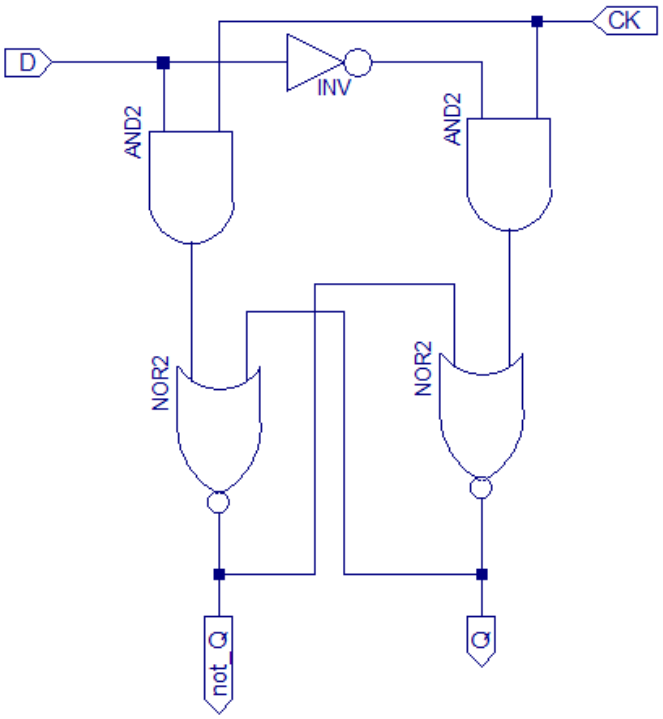
- Ambele scot la ieşire starea internă, modificare aceasteia depinzând de semnalul de ceas:

  - ✓ *D-Latches* (*zăvoare de date*): schimbarea are loc când **se schimbă intarea şi ceasul este activat**

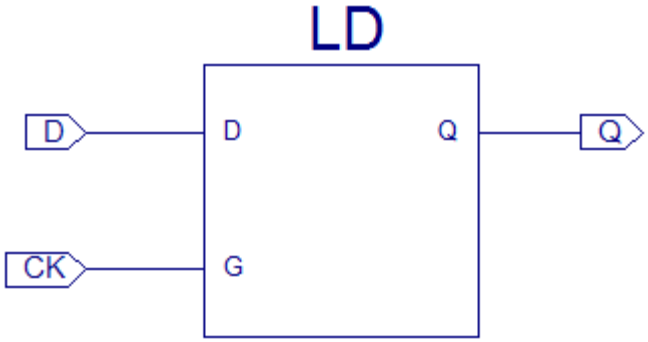
  - ✓ *D-Flip-Flops*: schimbarea are loc pe **frontul activ** al semnalului de ceas

# D-Latches

- *D-Latches* (*zăvoare de date*) prezintă*:*
  - ✓ 2 intrări: *CK (intrarea de ceas)* şi *D (intrarea de date)*
  - ✓ 2 ieşiri: *Q (starea sistemului)* şi $\bar{Q}$ *(starea negată)*



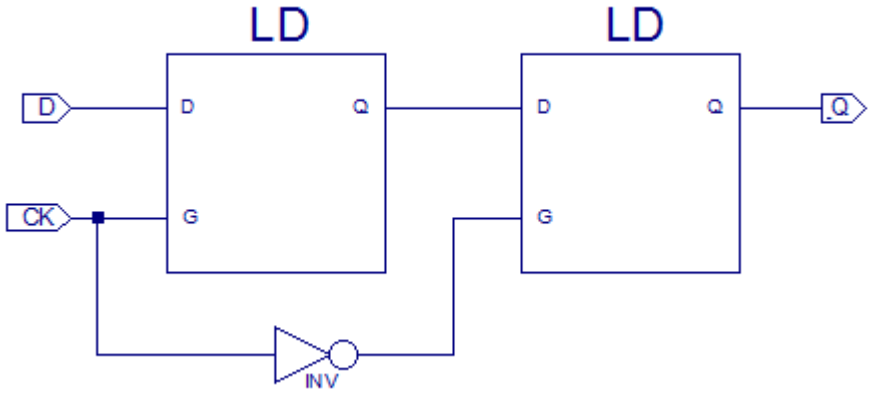

- Prezintă următoarea funcţionalitate:
  - ✓ Pentru $\mathrm{CK} = 1$, atunci D-Latch-ul este **deschis** şi Q depinde de D
  - ✓ Pentru $\mathrm{CK} = 0$, atunci sistemul **păstrează starea anterioară**




[Xilinx - ISE]

# D-Flip-Flops

- *D-Flip-Flops* prezintă*:*
  - ✓ 2 intrări: *CK (intrarea de ceas)* şi *D (intrarea de date)*
  - ✓ 2 ieşiri: *Q (starea sistemului)* şi $\bar{Q}$ *(starea negată)*




[Xilinx - ISE]

- Prezintă următoarea funcţionalitate:
  - ✓ Pentru $\mathrm{CK} = 1$, primul D-Latch este deschis şi permite setarea stării
  - ✓ Pentru $\mathrm{CK} = 0$, al doilea D-Latch se deschide şi copiază starea la ieşire.
- Modificarea memoriei se face pe frontul negativ al ceasului
- Circuitul scoate la ieşire intrarea D întârziată cu 1 tact

# Regiştri

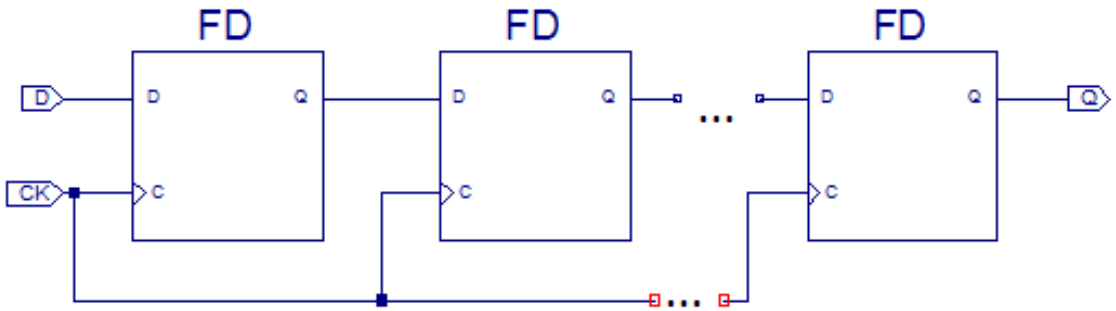
- *Regiştrul* pe *n* biţi se obţine imediat din compunerea (serială sau paralelă) a *n* D-Flip-Flop-uri cu semnal de ceas comun

- *Def.:* Un *registru serial* pe n biţi se defineşte recursiv astfel:
  - ✓ Pentru n = 1 este un D-Flip-Flop
  - ✓ Pentru n > 1 se obţine prin extensia serială a unui registru pe n-1 biţi cu un registru pe 1 bit

- *Def.:* Un *registru parelel* pe n biţi se defineşte recursiv astfel:
  - ✓ Pentru n = 1 este un D-Flip-Flop
  - ✓ Pentru n > 1 se obţine prin extensia paralelă a unui registru pe n-1 biţi cu un registru pe 1 bit

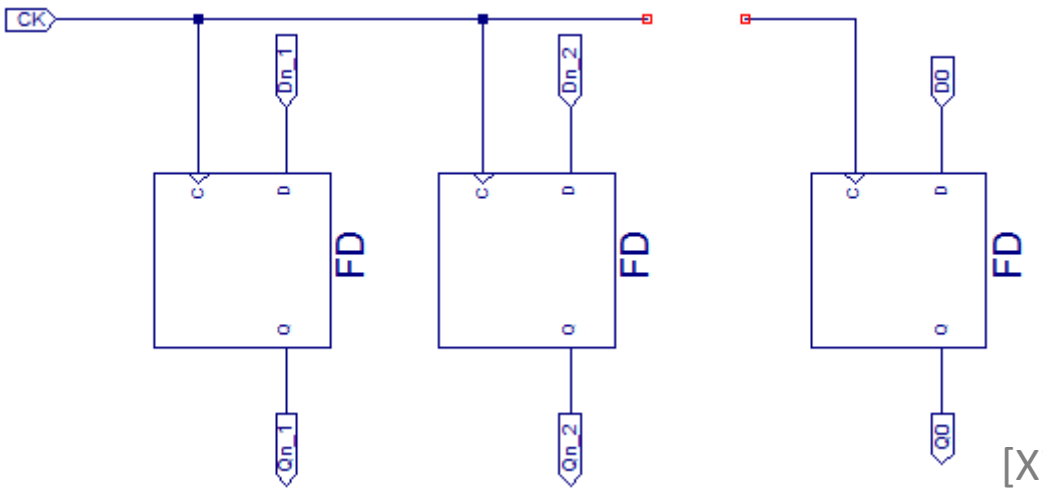
# Regiştri

- *Registrul serial* pe n biţi:




[Xilinx - ISE]

- *Registrul paralel* pe n biţi:




[Xilinx - ISE]

# Regiştri

- Regiştri seriali introduc o întânziere de n tacturi de ceas (se folosesc dacă se doreşte întârziere controlată)

- Mai mulţi regiştrii paraleli se pot lega în serie, formând un registru serial-paralel

# Register File

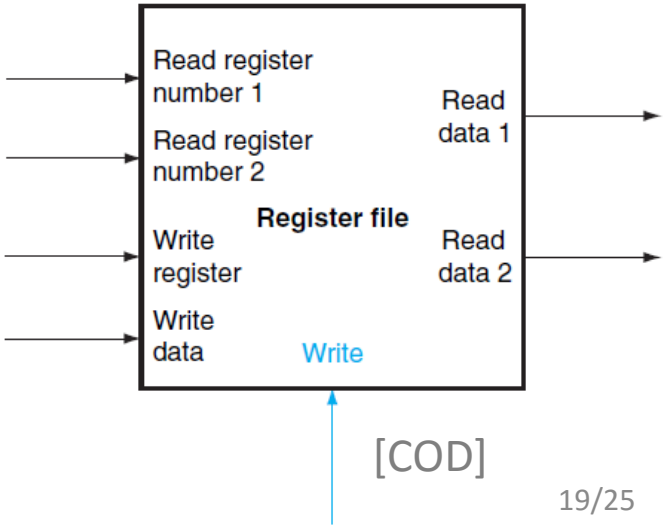
- *Fişier de Regiştri (Register File)* este un set de regiştri care pot fi accesaţi pentru citire / scriere prin numărul lor

**Intrare:**

- *Read Register number 1, Read Register number 2:* indică numărul celor 2 regiştrii din care se citeşte
- *Write Register:* indică numărul registriului în care se scrie
- *Write Data:* conţine informaţia care se scrie în registru

**Ieşire:**

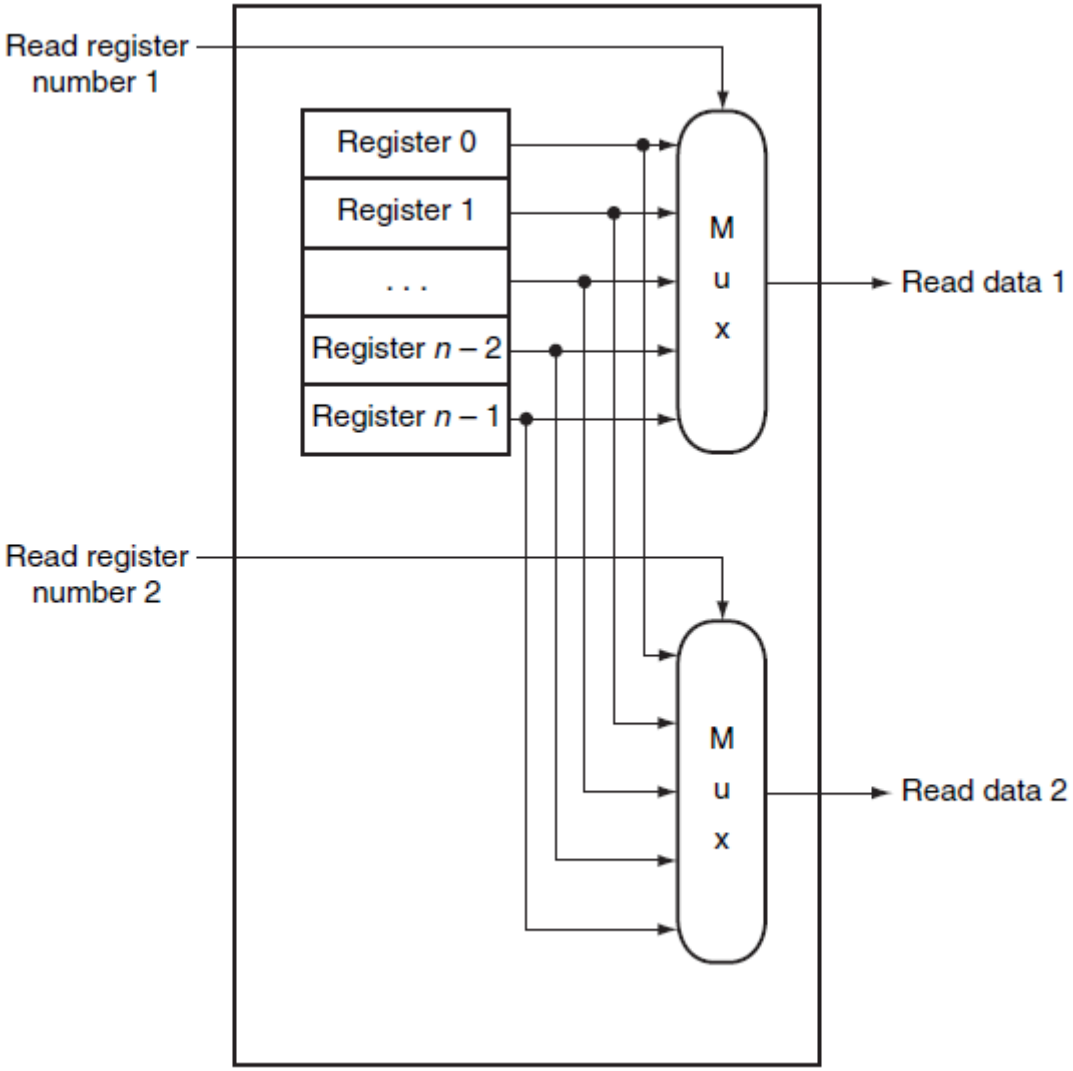
- *Read data 1, Read data 2:* conţine informaţia citită din regiştrii



# Register File

**Citire:**

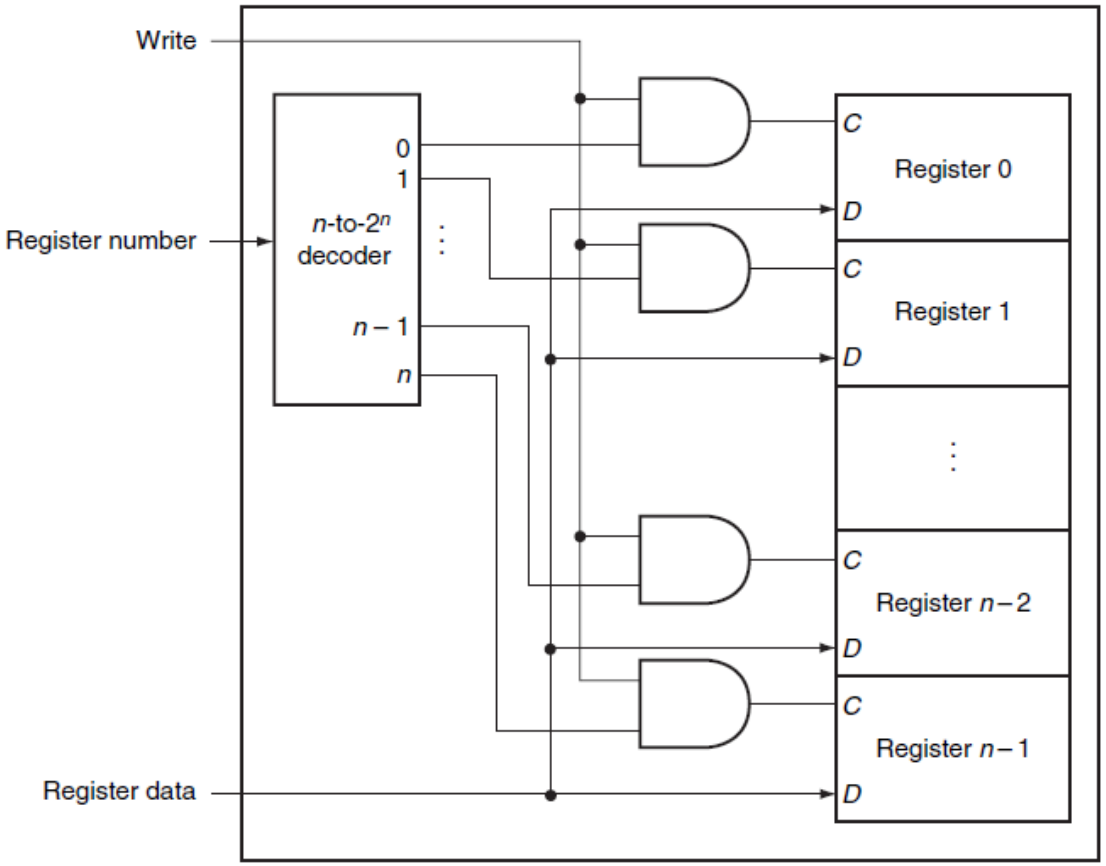
- Se foloseşte câte un **multiplexor** pentru identificarea fiecărui registru din care se citeşte



# Register File

**Scriere:**

- Se foloseşte un **decodificator** pentru identificarea registrului în care se scrie



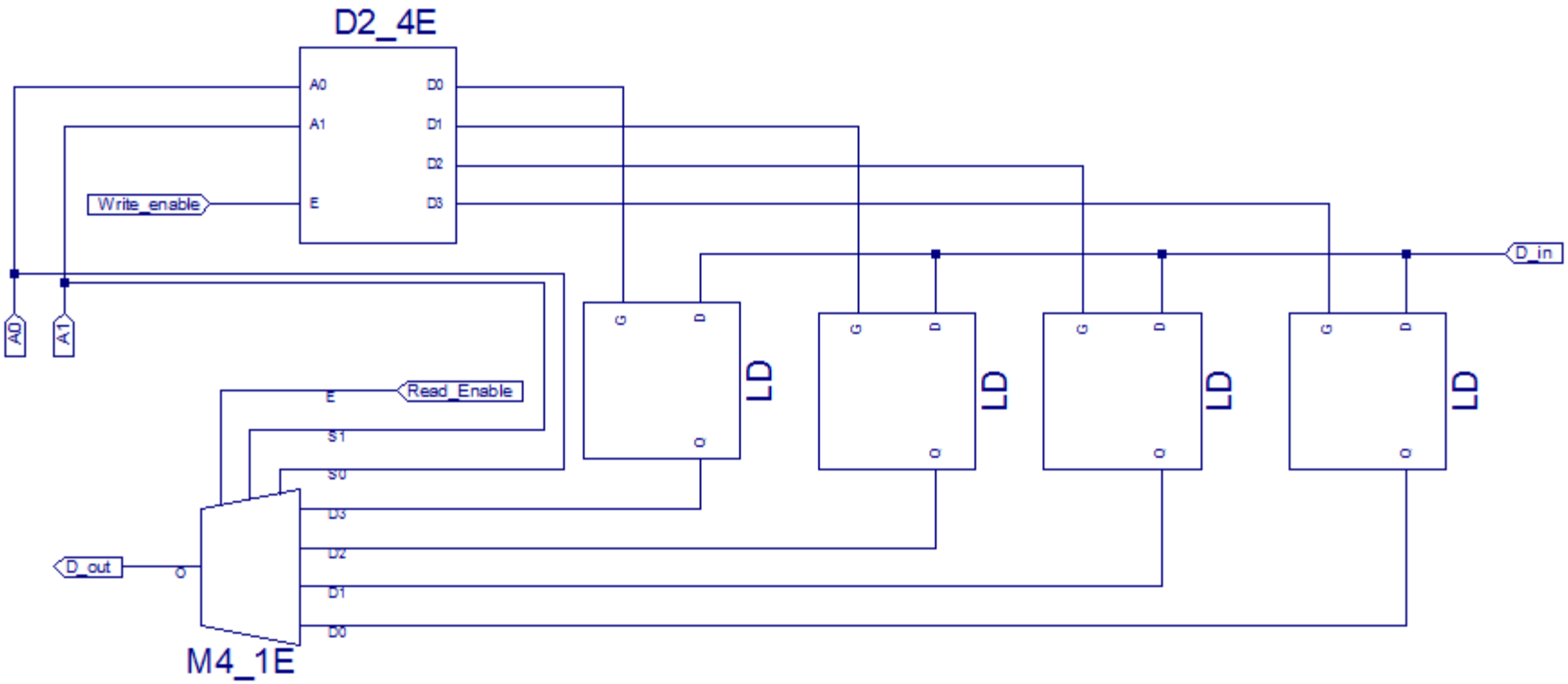

[COD]

# RAM (Random Access Memory)

- *RAM (Random Access Memory)* este un tip de memorie care permite atât citire cât şi scriere la adrese de memorie indicate:
- Este de 2 tipuri principale: *SRAM (Static RAM)* şi *DRAM (Dinamic RAM)*
  - ✓ *SRAM*: vectori de memorii, bazate pe latch-uri
  - ✓ *DRAM:* memorează datele folosind un condensator pentru fiecare bit
- Introducem succint doar structura RAM bazată pe D-Latches
- Mai multe informaţii la cursul următor (Challenge 2)

# RAM (Random Access Memory)

- Structura RAM pentru citirea / scrierea unei singure celule de memorie din 4 posibile este următoarea :



# RAM (Random Access Memory)

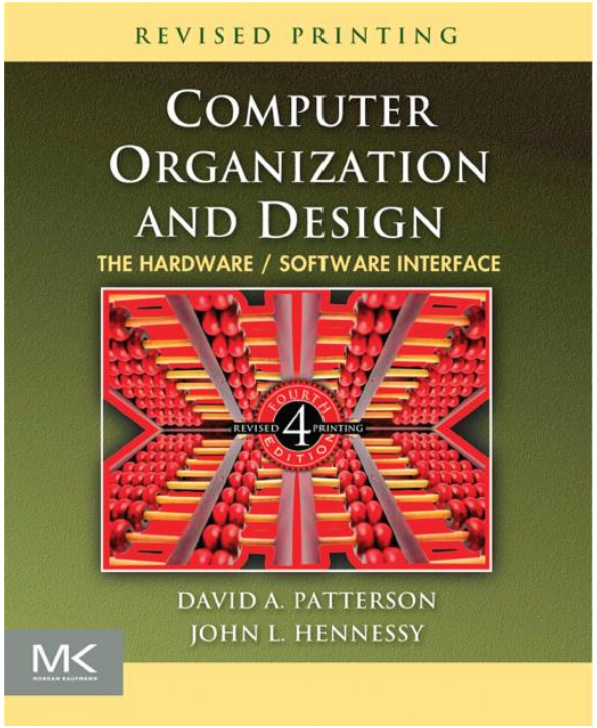
**Citire** (se utilizează un MUX) **:**

- *Read Enable*: activ la citire
- *A1,A0*: biţii adresei de unde se citeşte
- *D_out*: informaţia citită din memorie

**Scriere** (se utilizează un DCD) **:**

- *Write Enable*: activ la scriere
- *A1,A0*: biţii adresei de unde se scrie
- *D_in*: informaţia scrisă în memorie

# Referințe bibliografice




[AAT] A. Atanasiu, Arhitectura calculatorului


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

Schemele [Xilinx - ISE] au fost realizate folosind

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