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 14-

SINTEZĂ

- 2021-

# Reprezentarea numerelor în sistemele de calcul

- Sisteme de numerație poziționale (binar, octal, hexazecimal);
- Reprezentarea numerelor în virgulă fixă (SM, C1, C2);
- Reprezentarea numerelor de virgulă flotantă;
- Coduri binare pentru numere zecimale;

# Reprezentarea numerelor în virgulă fixă – **C2**

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| + | 2 | = | 0 | 0 | 1 | 0 | + | | | - | 2 | = | 1 | 1 | 1 | 0 | + | | |
| + | 3 | = | 0 | 0 | 1 | 1 | | | | + | 3 | = | 0 | 0 | 1 | 1 | | | |
| | | | 0 | 1 | 0 | 1 | = | + | 5 | | | | 0 | 0 | 0 | 1 | = | + | 1 |
| | | | | **a.** | | | | | | | | | | **c.** | | | | | |
| - | 2 | = | 1 | 1 | 1 | 0 | + | | | + | 2 | = | 0 | 0 | 1 | 0 | + | | |
| - | 3 | = | 1 | 1 | 0 | 1 | | | | - | 3 | = | 1 | 1 | 0 | 1 | | | |
| | | | 1 | 0 | 1 | 1 | = | - | 5 | | | | 1 | 1 | 1 | 1 | = | - | 1 |
| | | | | **b.** | | | | | | | | | | **d.** | | | | | |

- Domeniul valoric pentru numere întregi:
  - $-2^{n-1}$ și $2^{n-1}-1$

# Reprezentarea numerelor în virgulă flotantă (mobilă)

- reprezentate folosind notaţia ştiinţifică (care nu este poziţională) → un domeniu valoric foarte mare.
- Pentru a reprezenta un număr in virgulă flotantă folosim trei numere conform relaţiei:

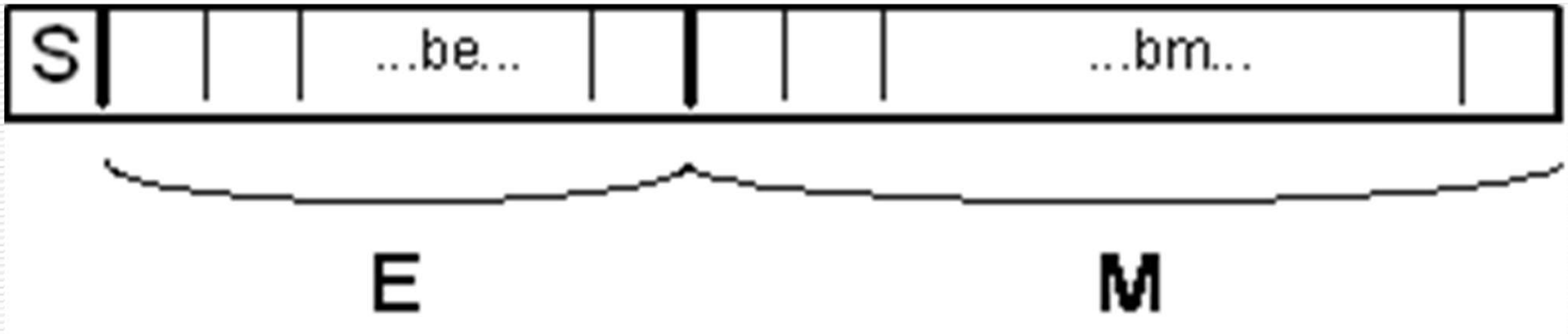
$$N = M * B^E$$

*M* - mantisa. (*M* poate fi reprezentată în SM sau C2)

*B* - baza (de obicei e 2 sau o putere a lui 2)

*E* - exponent. (E este reprezentat în SM sau cod exces)

# Reprezentarea numerelor în virgulă flotantă




*M* - mantisa. (*M* poate fi reprezentată în SM sau C2)

*B* - baza (de obicei e 2 sau o putere a lui 2)

*E* - exponent. (E este reprezentat în SM sau cod exces)

# Reprezentarea numerelor în virgulă flotantă

- Pentru a minimiza eroarea→ exponentul aferent lui 0 să fie cel mai mic posibil.
- valoarea min. a **oricărui** exponent să fie 0.
- **Toate** valorile negative reprezentabile pe N biţi sunt deplasate (devin pozitive) prin adunarea unui **bias** (unui surplus) = valoarea absolută a celui mai mic număr reprezentabil pe N numărul de biţi exponent.
- Pentru exponent reprezentat în:
  - SM pe 8 biți → valoarea bias-ului este 127
  - C2 pe 8 biți → valoarea bias-ului este 128

# Reprezentarea numerelor în virgulă flotantă

- Standardul IEEE 754/2008- formate:
  - Half precision
  - Simple precision
  - Double precision
  - Duble-extended

# Reprezentarea numerelor în virgulă flotantă: IEEE 754

- Caracteristici:
  - E şi M – format SM
  - Exponentul este reprezentat în exces de:
    - 127 pentru simplă precizie
    - 1023 pentru dublă precizie.
  - *Hidden bit*.
    - Mantisa are un bit de 1 ascuns.
    - bitul la dreapta virgulei care trebuie să fie 1 (din condiţia de normalizare).
    - (S.**1**M) (unde S este semnul iar M este mantisa) → virgula a fost mutată la dreapta bit-ului de 1 cel mai semnificativ: S**1**.M

# Reprezentarea numerelor în virgulă flotantă: IEEE 754 valori speciale

| Nr. | Exponent (E) | Mantisa (M) | Valoare speciala |
|---|---|---|---|
| 1. | 0 | 0 | $\pm 0$ |
| 2. | 0 | $\neq 0$ | Denormalized numbers |
| 3. | 255 | 0 | $\pm\infty$ |
| 4. | 255 | $\neq 0$ | NaN |

- Nr. denormalizate: rezultat care este mai mic decât valoarea minimă reprezentabilă
- Infinit: situaţia în care rezultatul intermediar este infinit sau avem overflow
- 0/0 sau radical din nr. negativ

# Coduri binare pentru numere zecimale - BCD

- există situaţii cand se doreşte afişarea rezultatelor de către interfeţele externe ale dispozitivului de calcul într-un format uşor de înţeles (decodificat) de către utilizator – şi anume mult întrebuinţatul format zecimal;
- cel mai la îndemână cod zecimal este BCD (binary-code decimal):
  - reprezentarea unei cifre BCD → înlocuirea cu reprezentarea în binar care îi corespunde→ cu un nr. pe 4 biţi

# Coduri binare pentru numere zecimale - BCD

- conversia unui număr zecimal în BCD prin înlocuirea succesivă a cifrelor zecimale cu tetradele corespunzătoare

$$N_1 = 4\ 5\ 9_{10} = \quad 0100\ \ 0101\ \ 1001_{BCD}$$

- operaţia inversă de transformare a unui număr reprezentat în BCD în omologul zecimal

$$N_2 = 1000\ \ 0111\ \ 0000\ \ 0010_{\mathrm{BCD}} = \quad 8\ 7\ \ 0\ 2_{10}$$

# Coduri binare pentru numere zecimale – Exces de 3

- Exces de 3 se obţine din codul BCD astfel:
  - la fiecare cifră reprezentată în cod BCD se adună valoarea 3 (0011 în binar).
  - fiecare cifră zecimale se reprezintă cu ajutorul unei combinaţii de 4 biţi (o tetradă de biţi)

# Algebra booleană și logica digitală

- Forma canonică;
- Forma standard;
- Aspecte legate de implementarea funcțiilor booleene cu porților logice;
- Hărți Karnaugh

**NAND**

| X | Y | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$$Z = \overline{X \bullet Y}$$

**AND**

| X | Y | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$$Z = X \bullet Y$$

**NOR**

| X | Y | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

$$Z = \overline{X + Y}$$

**OR**

| X | Y | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

$$Z = X + Y$$

# Porți logice (cont.)

**XOR**
$(X \oplus Y)$

| X | Y | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$Z = X\overline{Y} + \overline{X}Y$
X or Y but not both
("inequality", "difference")

**XNOR**
$\overline{(X \oplus Y)}$

| X | Y | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$Z = \overline{X}\,\overline{Y} + XY$
X and Y the same
("equality")

# Porți logice

- Fiecare poartă logică realizează una sau mai multe funcții logice;
- Colecția de porți logice folosită în realizarea unui circuit se numește **bibliotecă de porți**, iar porțile din cadrul ei **porți standard**;
- Bibliotecile moderne conțin zeci de porți a.î. să scadă costul cu întreținerea și să simplifice realizarea tool-urilor CAD

# Minimizarea funcțiilor logice

- se înţelege simplificarea/rescrierea ecuaţiilor logice booleene în vederea:
  - Unui cost mai mic și/sau;
  - Performanță mai ridicată;
- Cheia simplificării este: $y\left(x+\overline{x}\right)=y$
  - distributivitatea - *x(y+z)=xy+xz*
  - Proprietatea complementului $x+\overline{x}=1$

# Minimizarea funcțiilor logice

- Găsirea a doi termeni (suma sau produs funcție de reprezentarea dorită SOP/POS) pentru care:
  - funcția ia valoare 1
  - numai o variabilă își modifică valoarea

| A | B | F |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

B are aceși valoare → B este păstrat

A are valori diferite → A este eliminat

$$F = \bar{A}\,\bar{B} + A\bar{B} = (\bar{A} + A)\bar{B} = \bar{B}$$

# Construcție diagrame Karnaugh

- construcţia diagramelor Karnaugh pentru o funcţie logică cu 4 variabile de intrare

| ab \ cd | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0000 | 0001 | 0011 | 0010 |
| 01 | 0100 | 0101 | 0111 | 0110 |
| 11 | 1100 | 1101 | 1111 | 1110 |
| 10 | 1000 | 1001 | 1011 | 1010 |

c

b

a

d

# Pași:

- 1. Introducerea mintermilor în diagramă conform tabelului de adevăr.
- 2. se formează grupe de mintermi bazate pe reguli de adiacenţă:
  - O grupare are forma unor dreptunghiuri/pătrate și conţine 2^n mintermi!
  - Din totalul de *m* variabile booleene a funcţiei, termenul asociat grupării formate va conţine *m-n* variabile
- 3. Ecuaţia minimizată va conţine toţi implicanţii primi esenţiali, si uneori si implicanţi primi neesenţiali, astfel încât toate celule marcate cu 1 logic să fie acoperite.

# Minimizarea folosind diagrame Karnaugh

- Dacă la o astfel de grupare nu mai pot fi adăugaţi mintermi înseamnă că s-a obţinut un **implicant prim**.
- Dacă un anumit implicant prim conţine cel puţin un minterm care nu poate apare în alt implicanţi primi atunci acesta este un **implicant prim esenţial**

## PORȚI LOGICE CU MAI MULTE INTRĂRI

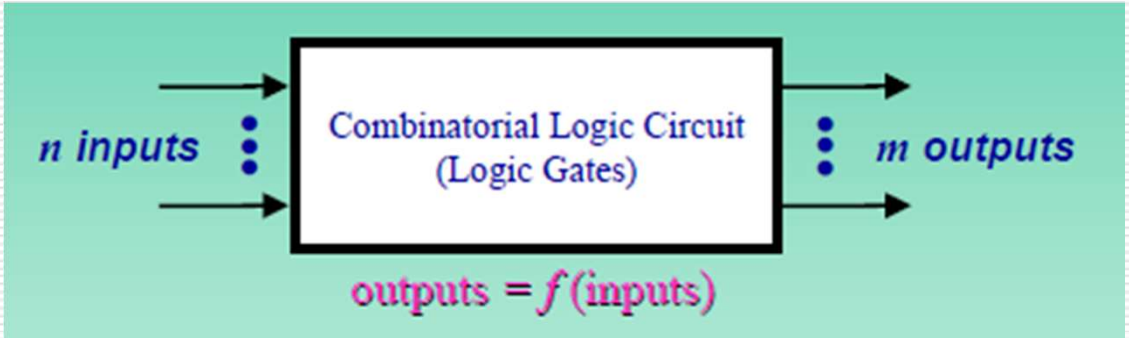
| Name | Graphic Symbol | Functional Expression | Number of transistors | Delay in *ns* |
|---|---|---|---|---|
| 3–input AND | | $F = xyz$ | 8 | 2.8 |
| 4–input AND | | $F = xyzw$ | 10 | 3.2 |
| 3–input OR | | $F = x + y + z$ | 8 | 2.8 |
| 4–input OR | | $F = x + y + z + w$ | 10 | 3.2 |
| 3–input NAND | | $F = (xyz)'$ | 6 | 1.8 |
| 4–input NAND | | $F = (xyzw)'$ | 8 | 2.2 |
| 3–input NOR | | $F = (x + y + z)'$ | 6 | 1.8 |
| 4–input NOR | | $F = (x + y + z + w)'$ | 8 | 2.2 |

# Circuite logice combinaționale

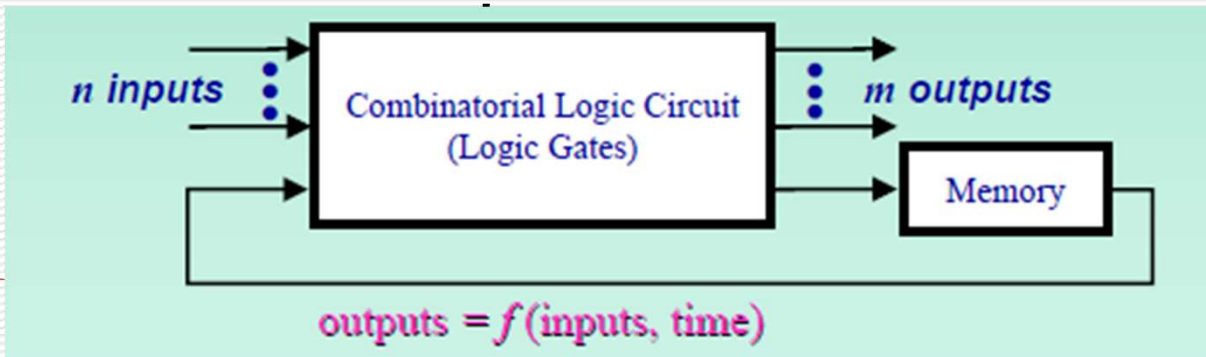
- Circuite de procesare
- Circuite de conversie
- Circuite de interconectare
- Componente universale

# Clasificare componente digitale

- Componente combinaționale
  - Ușor de analizat, partiționat, verificat




- Componente secvențiale




Copyright © 2004-2005 by Daniel D. Gajski

# Circuite reprezentative

- Sumator
- Decodificator
- Codificator (ex. Codificator de prioritate)
- Multiplexor

Caracterizarea acestora: definiție, ce se întâmplă dacă aplic o serie de intrări, care ar fi ieșirile, simbol.

# Circuite combinaționale

- Cunoașterea porților logice elementare;
- Desenarea unei scheme logice;
- Identificarea ecuației booleene aferente unei scheme logice;
- Determinarea ieșirilor unei scheme logica la care i s-au aplicat o serie de intrări;

# 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

# Multiplexor(Selector)

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

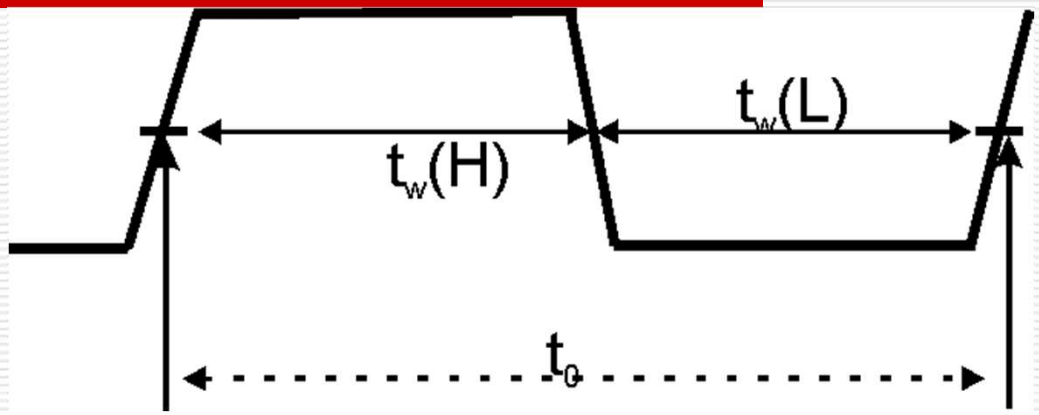
# Circuite secvențiale

- **Circuitele secvențiale se clasifică:**
  - **Asincrone**
  - **Sincrone**

- Componenentele secvențiale asincrone își modifică starea și valorile de ieșire funcție de modificările semanlelor de la intrare (oricând!) se modifică acestea.

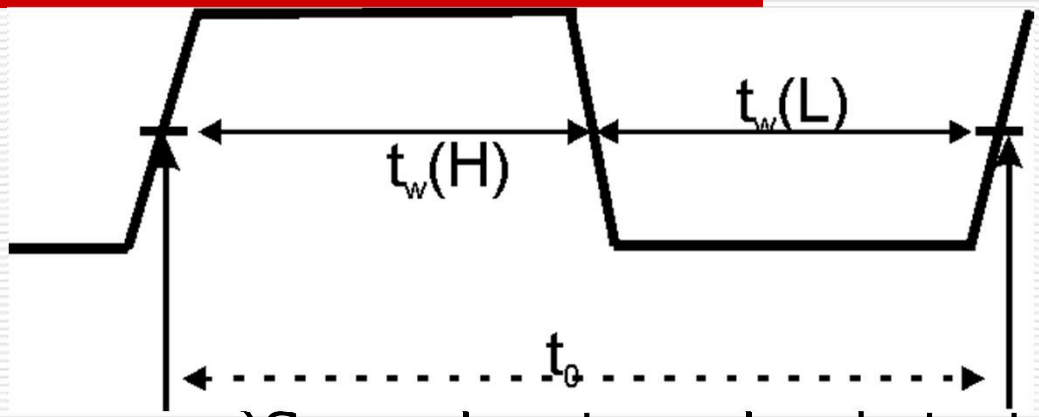
- Componentele secvențiale sincrone își modifcă valoarea funcție de valoarea semnalelor de intrare la momente bine definite de timp, dictate de un semnal (de intrare) care se numește tact (*clock*)

# Semnalul de tact

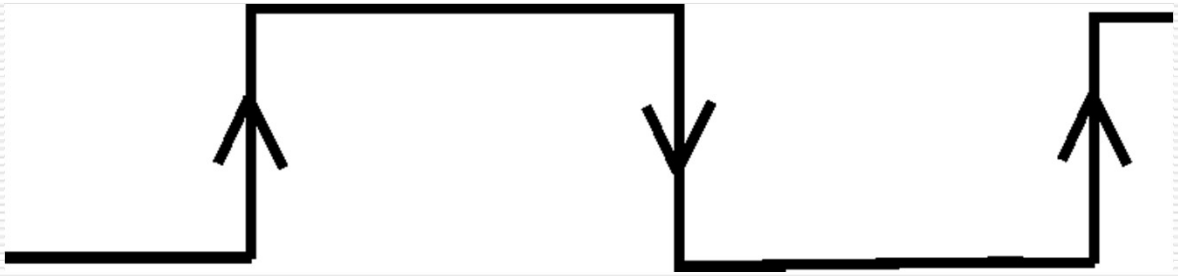



- **Palierul** uni semnal reprezintă porţiunea unde acesta rămâne constant o logic (palier negativ) şi 1 logic (palier pozitiv).
- **Frontul crescător** se referă la porţiunea unde senalul îşi modifică valoarea de la 0 logic la 1 logic (mai exact de la 10% din nivelul corespunzător lui 1 logic la 90% din nivelul corespunzător lui 1 logic
- t0 - perioada semnalului de tact,
- tw(H) şi tw(L) reprezintă durata unui impuls de 1 respectiv 0 logic

# Semnalul de tact

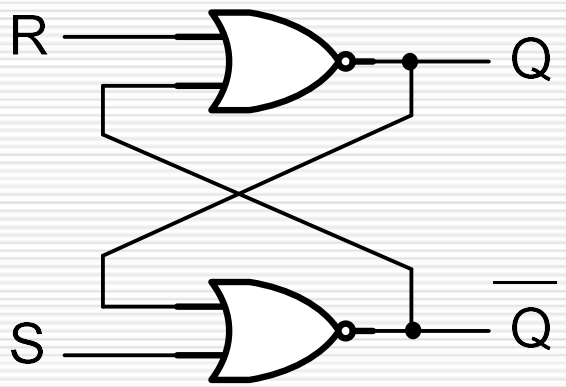



a)Semnal rectangular de tact

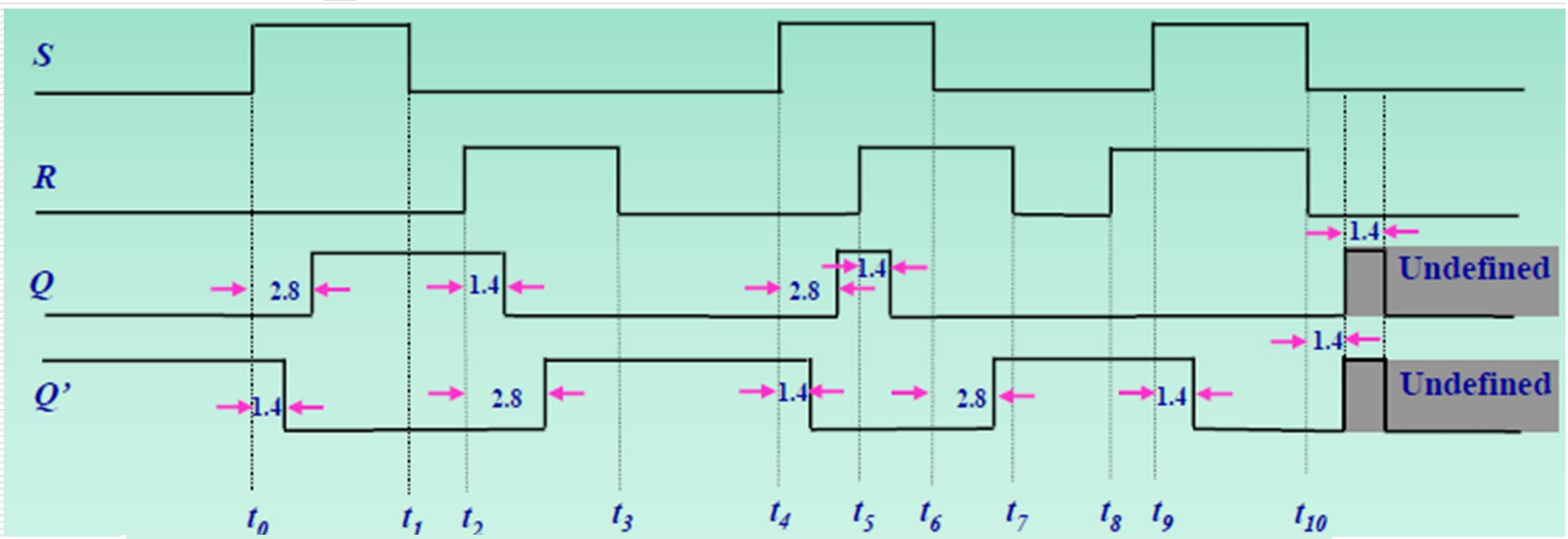


b)Formă idealizată a semnalului

# S-R Latch (SAU-NU) - asincron



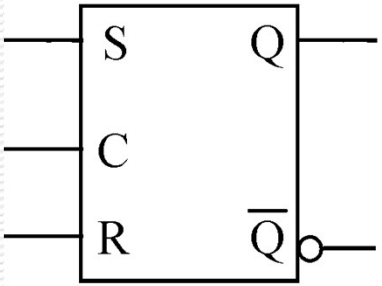

| R | S | $Q_{t+1}$ |
|---|---|---|
| 0 | 0 | $Q_t$ |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | interzis |




Copyright © 2004-2005 by Daniel D. Gajski

Slides by Philip Pham, University of California, Irvine

# Gated SR-latch



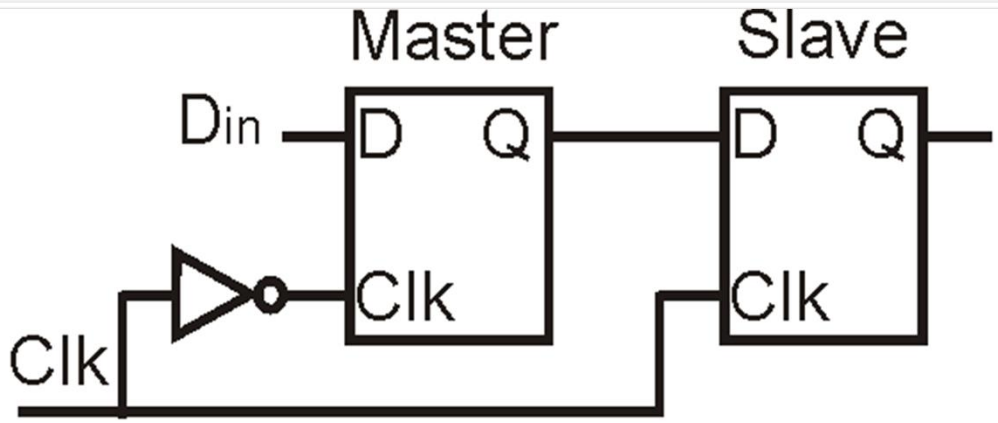

| S | R | C | Q_next | $\overline{Q}$_next |
|---|---|---|---|---|
| 0 | 0 | 1 | Q | $\overline{Q}$ |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | - | - |
| * | * | 0 | Q | $\overline{Q}$ |

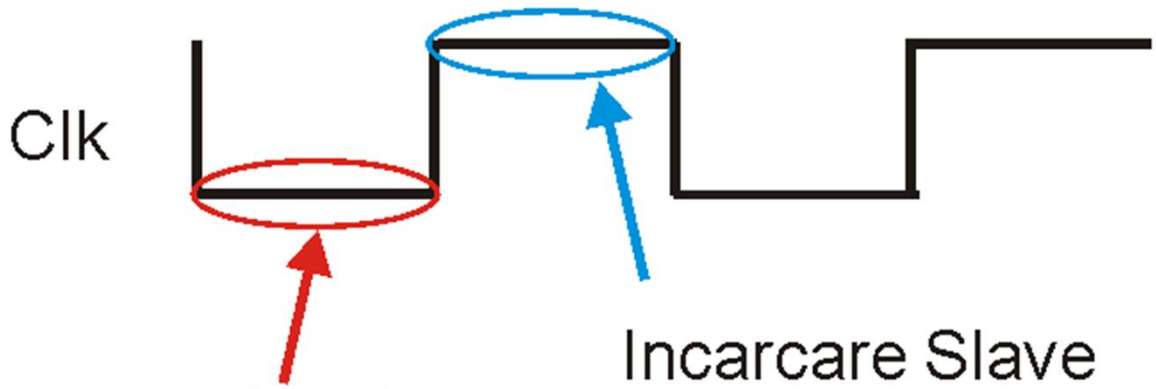
- Când semnalul C este activ valorile de la intrare sunt propagate prin latch
- Semnalele de intrare nu trebuie să se modifice în intervalul $t_{setup}$ și $t_{hold}$ al frontului descrescător

# Flip-flop-uri

- Se mai numesc și latch-uri sensibile pe frontul semnalului de tact;
- Basculare a se face pe frontul semnalului de tact (!nu pe palier – latch-uri)
- Două variante de arhitecturi:
  - Configurația master-slave
  - Edge-triggered FF

# FF-Master-Slave

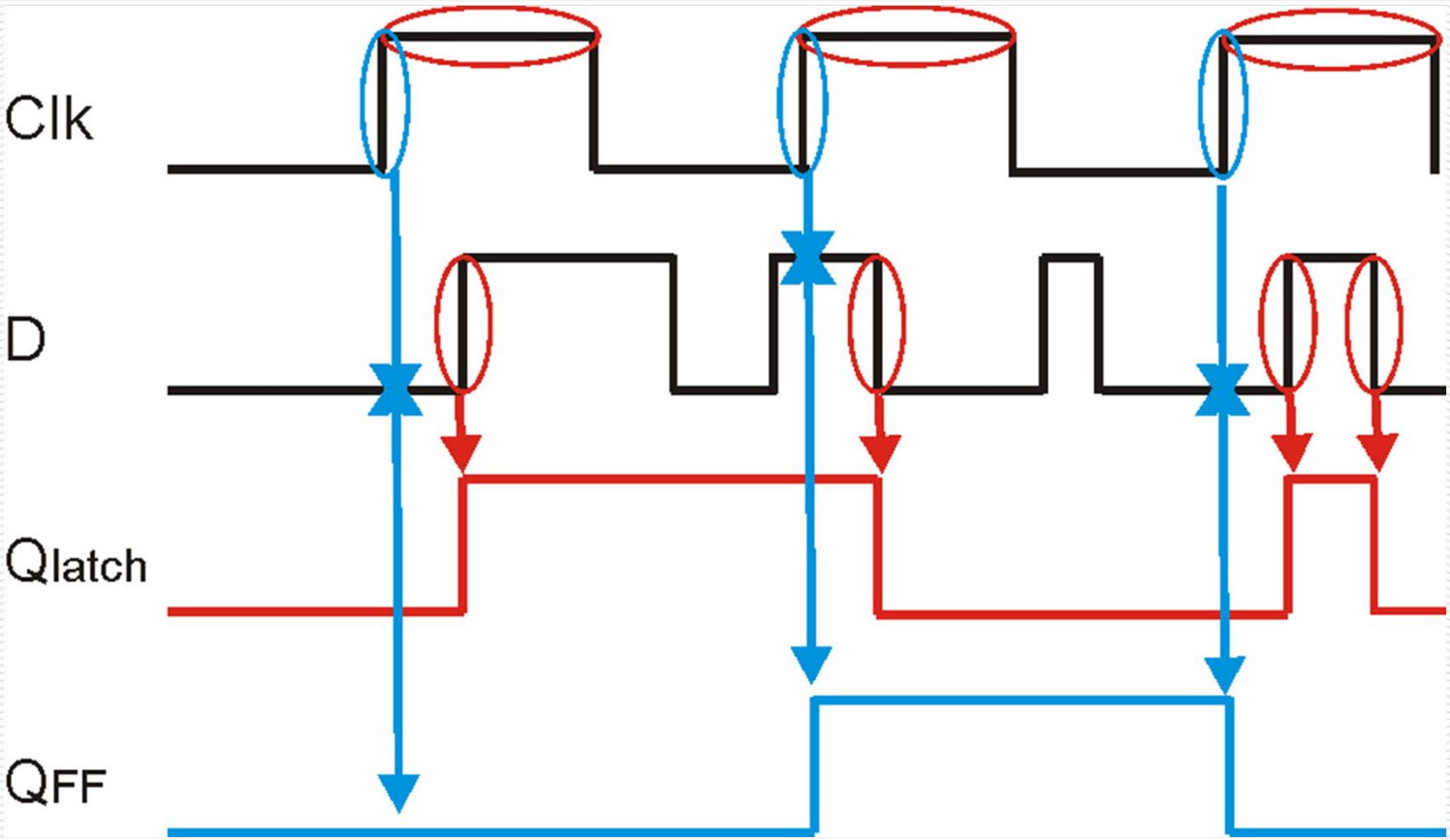



Incarcare Master Latch cu $D_{in}$
Slave Latch memoreaza data anteriora

Incarcare Slave Latch cu $Q_{master}$
Master Latch memoreaza valoarea $D_{in}$ dinaintea comutarii semnalului Clk

# Latch sincron vs. FF sincron

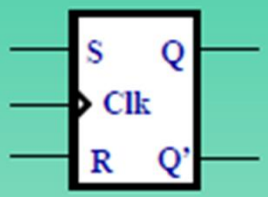
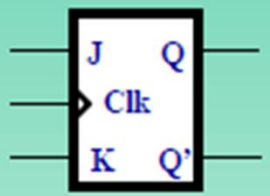
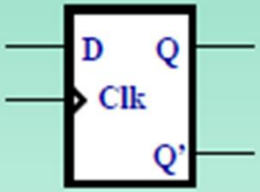


# Operatia de reset a elementelor secventiale

- Semnalul de reset (set)
  - Functionalitate – aducerea bistabilului intr-o stare “initiala” cunoscuta (de obicei starea 0)
  - Reset este un semnal global – este aplicat tuturor elementelor de memorie dintr-un sistem digital
  - Tipuri de reset
    - Reset sincron
    - Reset asincron

Copyright © 2004-2005 by Daniel D. Gajski

# Tipuri de FF-uri

| Flip-flop name | Flip-flop symbol | Characteristic table | Characteristic equation | Excitation table |
|---|---|---|---|---|
| SR |   | | $Q(next)=S+R'Q$<br>$SR=0$ | |
| JK |   | | $Q(next)=JQ'+K'Q$ | |
| D |   | | $Q(next)=D$ | |
| T | | | $Q(next)=TQ'+T'Q$ | |

| S | R | Q(next) |
|---|---|---|
| 0 | 0 | Q |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | NA |

| Q | Q(next) | S | R |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | X | 0 |

| J | K | Q(next) |
|---|---|---|
| 0 | 0 | Q |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | Q' |

| Q | Q(next) | J | K |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

| D | Q(next) |
|---|---|
| 0 | 0 |
| 1 | 1 |

| Q | Q(next) | D |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| T | Q(next) |
|---|---|
| 0 | Q |
| 1 | Q' |

| Q | Q(next) | T |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Copyright © 2004-2005 by Daniel D. Gajski

# Circuite secvențiale reprezentare

- **Circuitele secvenţiale:**
  - **MEALY** sunt caracterizate prin faptul că starea următoare şi ieşirea la un moment dat depind de starea ***prezentă*** si de ***intrarea prezentă***;
  - **MOORE** sunt caracterizate prin faptul că ieşirea depinde **numai** de **starea circuitului**. Starea următoare depinde de intrarea prezentă;
- Modelele matematice ale circuitelor secvenţiale se numesc in teoria comutaţiilor **automate finite**.

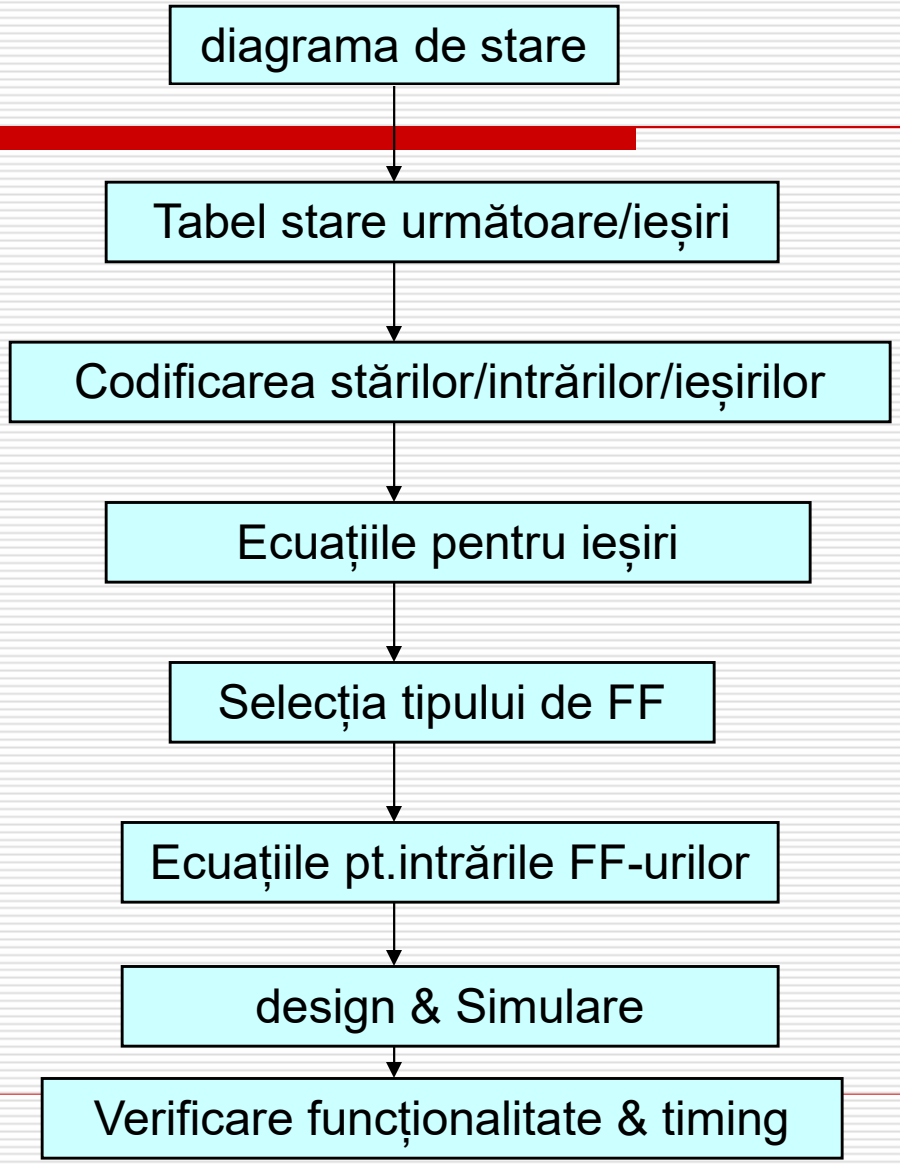
# Registre

- Reprezinta o colectie/grupare de *n* bistabile
- Nr maxim de valori a unui registru pe *n* biti – $2^n$ valori binare
- Folosit pentru memorarea unui cuvant de date/unei stari curente a sistemului

# Numărătoare

- circuite **secvenţiale sincrone autonome** (mulţimea intrărilor vidă), care baleiază o secvenţă de stări impuse de proiectant.
- de regulă este inițializat cu starea ,,0” , după care la fiecare impuls de numărare, comuta într-o nouă stare.
- caracterul **asincron** al unui numărător este dat de faptul că impulsul de tact nu comandă simultan toate bistabilele numărătorului.
- Funcție de direcția de parcurgere a secvenței de stări:
  - numărător în sens crescător,
  - numărător în sens descrescător,
  - numărător reversibil (ambele sensuri).

# Etape de sinteză circuit secvențial



# Circuite secvențiale

- Citirea unei diagrame de stări și realizarea tabelului tranzițiilor;
- Citirea unei diagrame de simulare:
  - Identificarea tipului de RST
  - Extragerea caracteristicilor unui circuit simplu (secventa de stări prin care trece circuitul pe baza intrărilor și stării curente din diagrama de timp)
- Clasificarea memoriilor, număr biți de adrese vs. capacitate, DRAM vs. SRAM

# Verilog HDL

- Diferența dintre *reg* și *wire*;
- Aspecte legate de modelarea corecta a unui block combinațional;
- Aspecte legate de modelarea corecta a unui block secvențial;
- Diferența dintre cei 2 operatori de atribuire =, =>
- Identificarea din cod a tipului de RST;

# Laborator

- 3 note:
  - Nota1_TestComb: Test 1
  - Nota2_TestSecv: Test 2
  - Nota3_Activitate: Nota Activitate (assignment-uri notate, răspunsuri în timpul laboratorului, simulare prezentată în timpul laboratorului)
- VERIFICAȚI pe CV!
- Scrieți la TA până cel târziu 3 iunie pentru orice nelămurile legată de aceste note;
- Nota LAB: media aritmetica a celor 3 note

# Examen

- QUIZ
- Open book
- Poate conține întrebări de tip eseu
- 2 părți
- Nota minimă pt fiecare parte: 4.5
- Date examinare:
  - Data1: 03 iunie, ora 10
  - Data2: 28 iunie, ora 10

# Întrebări?

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

*Imagini preluate din ECE 448 – FPGA and ASIC Design with VHDL*