LB

(curs 11 - SII)

10.05.2024

RTM (Register Transfer Methodology)

- descriere foarte similară cu cea utilizată pentru algoritmi, şi anume descrie fluxul de date prin prisma operaţiilor care au loc la nivel de registre.
- se poate realiza o analogie între operaţii la nivel de registru şi variabilele dintr-un algoritm.
- Nivelul RT (register transfer) de abstractizare este situat între nivelul poartă logică şi nivelul procesor.

- situat între nivelul poartă logică şi nivelul procesor.
- este descris în termeni de operaţie RT de bază.
- o unitate digitală este descrisă ca o succesiune de operaţii RT de bază, succesiune dictată de o logică (modul) de control.

Operaţia RT elementară

- este în esenţă o funcţie simplă care calculează valoarea a unui registru destinaţie pe baza conţinutului registrelor sursă și a semnalelor de intrare.

$$R_{DEST} \leftarrow f\left(R_{SRC1}, R_{SRC2}, \ldots, R_{SRC\_M}\right)$$

- Pe frontul crescător al tactului, informaţia disponibilă la intrare este încărcată in registre. După o întârziere (timp de propagare) ea este disponibilă la ieşiri.
- O logică combinaţională calculează funcţia *f* funcţie de intrări şi valorile încărcate în registruu.
- Rezultatul calc. este trimis printr-o logică combinaţională la intrările registrului destinaţie (ex. MUX).
- La proximul front crescător al semnalului de tact, rezultatul de la intrarea registrului destinaţie este încărcat în registru.

RT exemplu: suma

$R_1 \leftarrow R_1 + R_2$

Considerăm următoarele sufixe pentru registre:

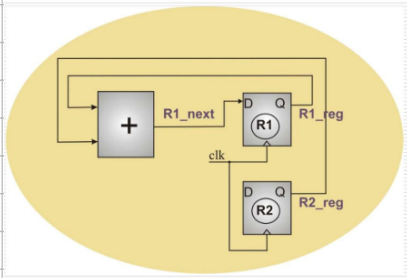
Sufix _reg – face referire la ieşirea curentă (din timpul impulsului de tact curent) a registrelor;
Sufix _next – face referire la datele care sunt la intrarea registrelor, calculaţii complete sau rezultate intermediare (din timpul impulsului de tact curent);

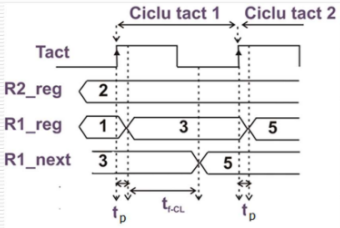
Atragem atenţia asupra comportamentului registrelor de a încărca datele primite la intrare numai la momente bine stabilite de timp (în discuţia noastră pe frontul crescător al impulsului de tact).

Aşadar operaţia RT se poate exprima astfel:

$R_1\_next \leftarrow R_1\_reg + R_2\_reg$
$R_1\_reg \leftarrow R_1\_next$ la intalnirea primului front crescator al tactului

- Rezultat sinteza & diagrama de timp

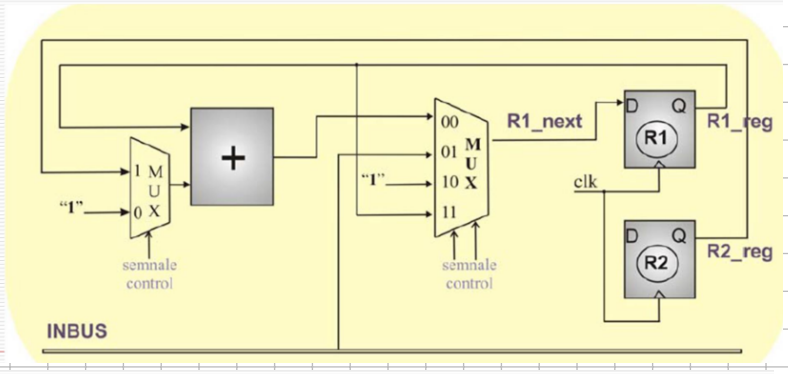




Mai multe operatii RT

Ex.:

$R_1 \leftarrow R_1 + R_2$
$R1 \leftarrow R1 + 1$
$R1 \leftarrow InBus$
$R1 \leftarrow 1$
$R1 \leftarrow R1$ - nop

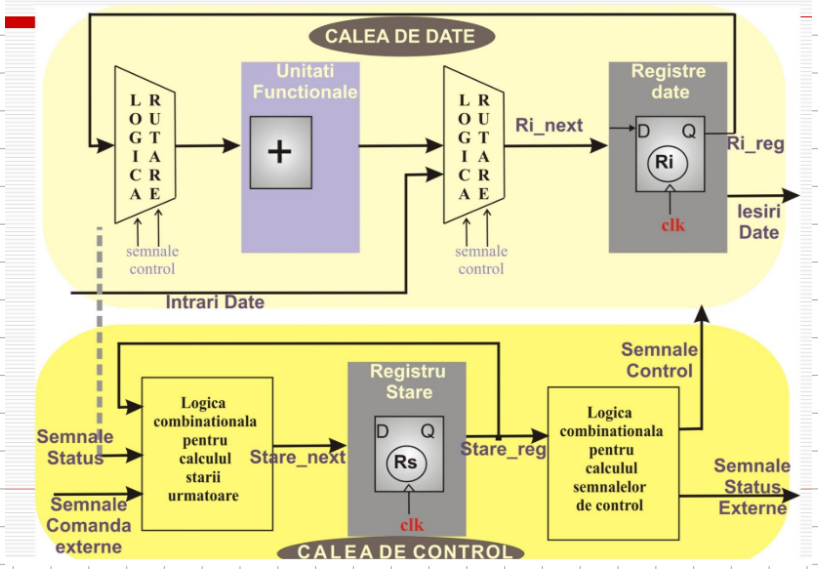


Automat cu stări finite şi cale de prelucrare a datelor (FSMD)

O implementare RTL necesită:

- Cale date:
  - prelucrarea şi rutarea datelor de către elemente secventiale de memorare.
    - unităţi funcţionale de prelucrare a datelor
    - logică de rutare (de regulă reprezentată de multiplexoare)
  - registre pentru stocarea datelor
- cale de control:dictează când şi ce operaţie RT se execută (FSM).

FSMD

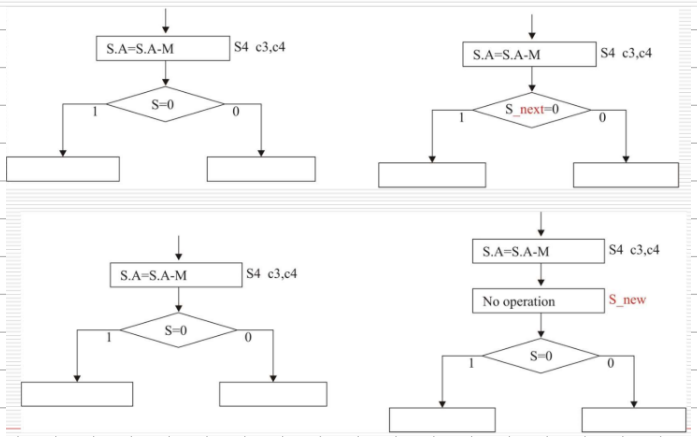


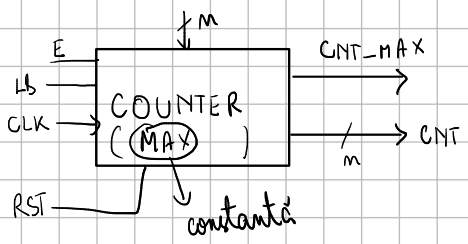
Algoritmic state machine

- constă din blocuri de decizie, blocuri in care se efectueaza secvente de RT ops, blocuri de start/stop.
- uneori blocurile de test au nevoie sa verifice conditia pe valoarea de intrare a unui registru.

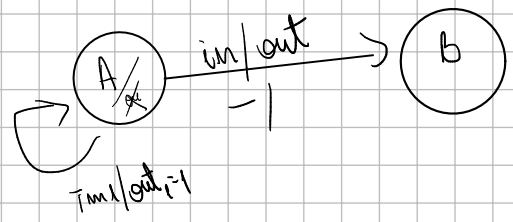
ASM charts

- testează informaţia din registru care a fost modificată în *pasul curent* (aşadar nu a beneficiat de un front crescător care să ducă la încărcarea ei în registru corespunzător). Acest scenariu prezintă două soluţii:
  - Introducerea unei stări echivalente unui nop (no operation) care să ofere timpul necesar update-ului;
  - Folosirea pentru testare a valorii de *_next* (valoarea de la intrare care încă nu este încărcată în registru);





E	LB	Operations
0	0	NOP
*	1	LOAD COUNT
1	0	COUNT UP



Ex.: Fibonacci

$$fib(n) = \begin{cases} 0, & n=0 \\ 1, & n=1 \\ fib(n-1) + fib(n-2), & n>1 \end{cases}$$

```
INPUT:
        An[31:0]=1;
        An-1[31:0]=1;
        An-2[31:0]=0;                          {S1, c0}
        N[7:0]=INBUS;
BEGIN:
        If (N=0) then goto OUTPUT;
        An=An-1+An-2; .... .... .... .... ....  {S2,c1}
        An-2=An-1;    .... .... .... .... ....  {S3,c2}
        An-1=An;  }
        N=N-1;    }   .... .... .... .... ....  {S4,c3}
        goto BEGIN
OUTPUT:                                          {S5,c4}
         OUTBUS=An;                              c5=END
```