

1. ELEMENTE INTRODUCATIVE PRIVIND OPERAREA ȘI ORGANIZAREA UNUI SISTEM NUMERIC

2. CONVENȚII DE PROIECTARE

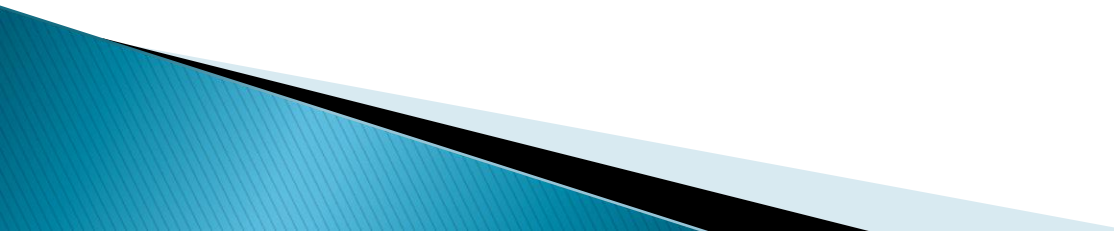
– *Curs4* –

Subiecte abordate:

1. ELEMENTE INTRODUCTIVE PRIVIND OPERAREA ȘI ORGANIZAREA UNUI SISTEM NUMERIC

- Elementele calculatorului după principiile lui von Neumann;
- Exemplu: algoritmul MAX

2. CONVENȚII DE PROIECTARE

- Transferurile între registre
 - Componente combinate
 - Componente secvențiale
- 

Un calculator numeric este constituit dintr-un ansamblu de *resurse fizice (hardware)* și de *programe de sistem (software de bază)*, care asigură prelucrarea automată a informațiilor, în conformitate cu algoritmi specificați de către utilizator, prin *programele de aplicații (software de aplicații - utilizator)*.

Conform principiilor stabilite de John von Neumann un calculator trebuie să posede următoarele elemente:

- ❖ un *mediu de intrare*, pentru instrucțiuni și date (operanzi);
- ❖ o *memorie* în care se stochează programul, datele inițiale, rezultatele parțiale și finale;
- ❖ un *ansamblu de prelucrare*, capabil să efectueze operații aritmetice și logice, în conformitate cu un algoritm dat, specificat prin program;
- ❖ un *mediu de ieșire*, pentru extragerea rezultatelor și prezentarea acestora într-o formă accesibilă utilizatorului;
- ❖ un *element de decizie* care, pe baza rezultatelor parțiale obținute pe parcursul prelucrării, va selecta una din opțiunile posibile de continuare a calculelor.

Funcționarea calculatorului are un *caracter secvențial*, constând în citiri și execuții succesive ale instrucțiunilor programului.

Într-un calculator pot fi evidențiate, pe parcursul execuției unui program, *două fluxuri de informații*:

fluxul datelor care se prelucrează

fluxul instrucțiunilor care controlează/ comandă procesul de calcul

Calculatoarele bazate pe principiile amintite mai sus se numesc *calculatoare von Neumann* sau convenționale, fiind comandate de fluxul de instrucțiuni.

Un algoritm reprezintă un set finit de reguli, care precizează o secvență de operații, pentru soluționarea unei clase date de probleme.

Un algoritm posedă cinci elemente mai importante:

- ★ *caracter finit*: trebuie să se termine după un număr finit de pași;
- ★ *caracter determinist*: fiecare pas al unui algoritm trebuie definit în mod precis, acțiunile care se execută trebuie să fie specificate riguros, fără ambiguități, pentru fiecare caz; execuția algoritmului cu același set de date de intrare trebuie să conducă la același rezultat;
- ★ *intrare*: un algoritm are una sau mai multe intrări, reprezentând datele inițiale;
- ★ *ieșire*: un algoritm are una sau mai multe ieșiri, care reprezintă rezultatele, aflate într-o anumită relație cu intrările;
- ★ *eficacitate*: un algoritm trebuie să se execute exact și într-un interval finit de timp.

Definirea câtorva noțiuni:

- *variabilele de stare* reprezintă mărimi primare, care presupun unele valori bine definite (ele pot reprezenta parametrii unui sistem fizic, de exemplu);
- un ansamblu de variabile de stare, în care fiecare poartă un nume, reprezintă *mulțimea variabilelor de stare*;
- o atribuire dată de valori pentru toate variabilele mulțimii variabilelor de stare poartă numele de *stare a mulțimii* sau o stare presupune o valoare dată fiecărei variabile de stare;
- ansamblul tuturor stărilor posibile, pentru o mulțime dată de variabile de stare, formează *spațiul stărilor* pentru acea mulțime;
- un calcul în spațiul stărilor reprezintă *o secvență de stări* în acel spațiu, primul element al secvenței reprezintă *starea inițială*, iar ultimul constituie *starea finală*.

Din punct de vedere formal, *metoda de calcul* reprezintă un quadruplu $\langle Q, I, E, f \rangle$ în care s-au făcut următoarele notații:

Q - mulțimea stărilor calculului,

I - mulțimea intrărilor,

E - mulțimea ieșirilor,

f - mulțimea funcțiilor de calcul, definite în **Q**.


Relații:

$$I \subset Q \quad ; \quad E \subset Q$$

fiecare intrare x în mulțimea **I** definește o secvență de calcul:

$$x_0, x_1, x_2, \dots, x_k, \dots,$$

$$x_0 = x \quad \text{si} \quad x_{k+1} = f(x_k) \quad \text{pentru } k > 0$$

dacă k este cel mai mic întreg pentru care x_k este în **E**  secvența de calcul se termină în k pași

Un algoritm reprezintă o metodă de calcul, care se termină după un număr finit (eventual foarte mare) de pași, pentru toate intrările $x \in I$.

Algoritmul MAX

- ☼ găsește elementul cu valoarea cea mai mare al mulțimii $\{A(i)\}$, unde $1 \leq i \leq n$, și o atribuie ieșirii **MAX**.

ALGORITM: MAX.

intrări: $\{A(i)\}$, $1 \leq i \leq n$,

ieșiri: MAX,

var.de stare: $\{x_c, x_m\}$.

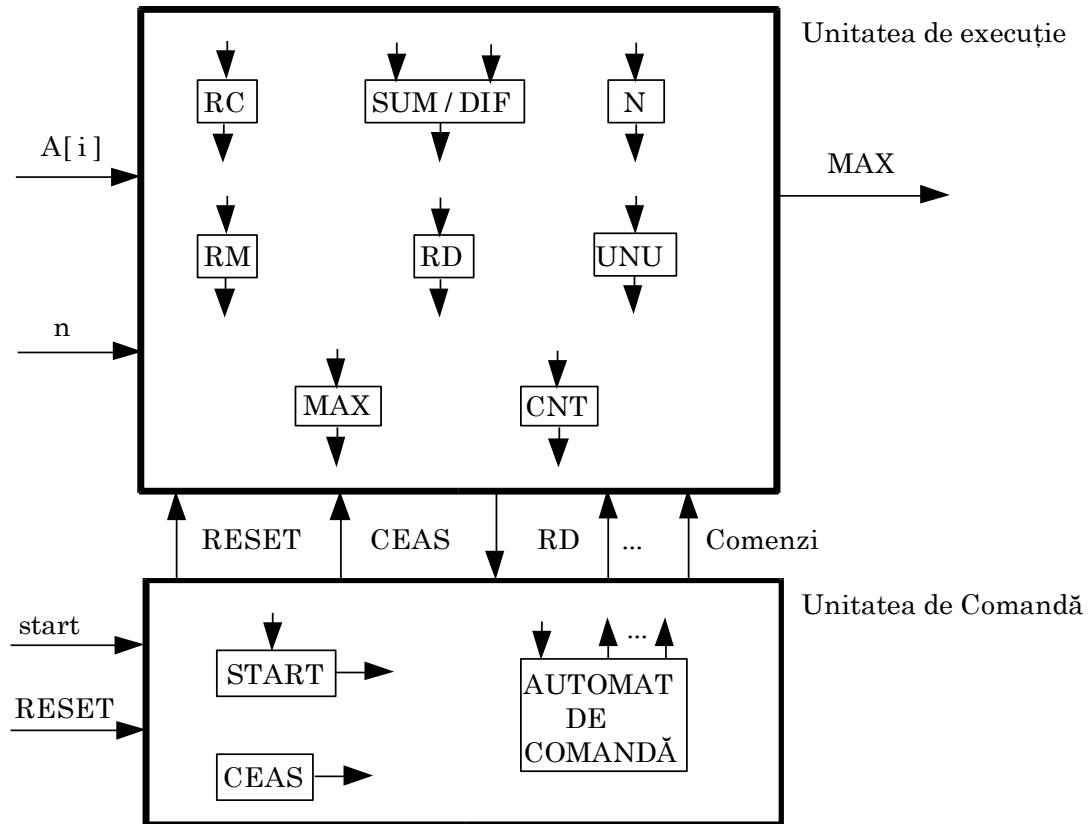
f: secvența de calcul:

- 1. if $n < 1$ go to STOP*
- 2. if $n = 1$ then $MAX \leftarrow A(1)$ and go to 9 (STOP)*
- 3. $x_m \leftarrow A(1)$; $x_c \leftarrow A(2)$; $i \leftarrow 3$*
- 4. if $x_m < x_c$ then $x_m \leftarrow x_c$*
- 5. $x_c \leftarrow A(i)$*
- 6. $i \leftarrow i + 1$*
- 7. if $i > n$ then $MAX \leftarrow x_m$ and go to STOP*
- 8. go to 4*
- 9. STOP*

Mecanizarea acestui algoritm presupune *existența unui modul*, care dispune de următoarele resurse hardware:

- RC**: registru în care se aduce *valoarea curenta $A(i)$* ;
- RM**: registru în care se plasează *valoarea curentă maximă $A(j)$* ;
- N** și **UNU**: registre în care se păstrează *constantele n și 1* ;
- CNT**: contor pentru *indexul i* ;
- RD**: registru în care se plasează *rezultatul scăderii*;
- MAX**: registru de *ieșire* (coincide ca nume cu *ieșirea MAX*);
- START**: bistabil în care se înregistrează comanda externă *start*;
- SUM/DIF**: unitate logică combinațională, care efectuează *adunarea/scăderea*;
- un automat cu 10 stări distincte.

Partiționarea modului MAX în unitate de execuție și unitate de comandă.

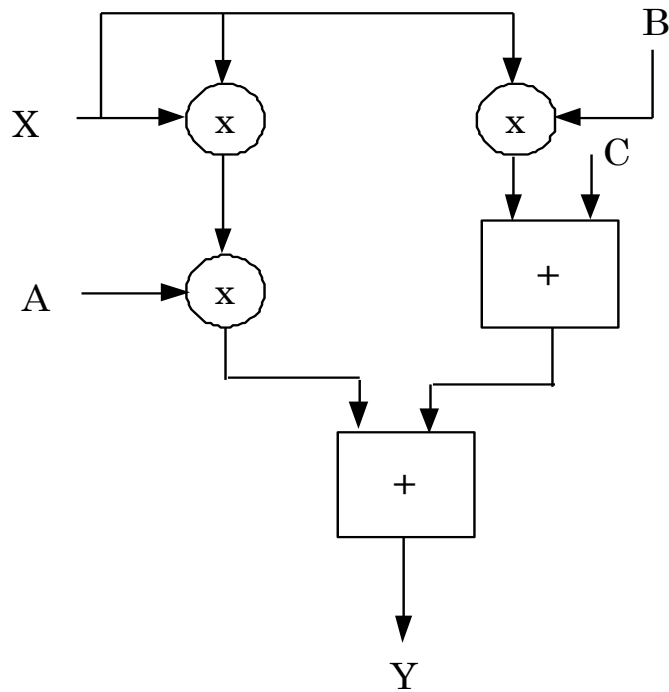


Algoritmii se pot caracteriza printr-un paralelism propriu (posibilitatea efectuării mai multor operații elementare în paralel), ceea ce permite creșterea vitezei de execuție, în condițiile existenței resurselor hardware necesare.

Exemplu de implementare a calculului valorii unui polinom de gradul 2:

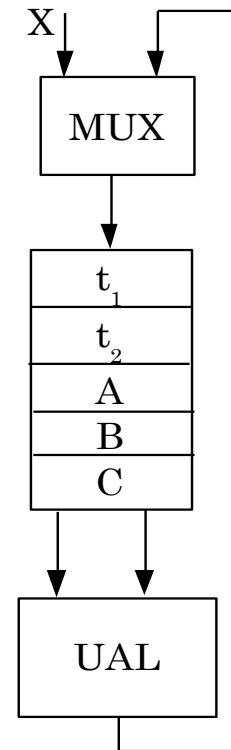
$$y = Ax^2 + Bx + C$$

Soluție paralelă (spațială)

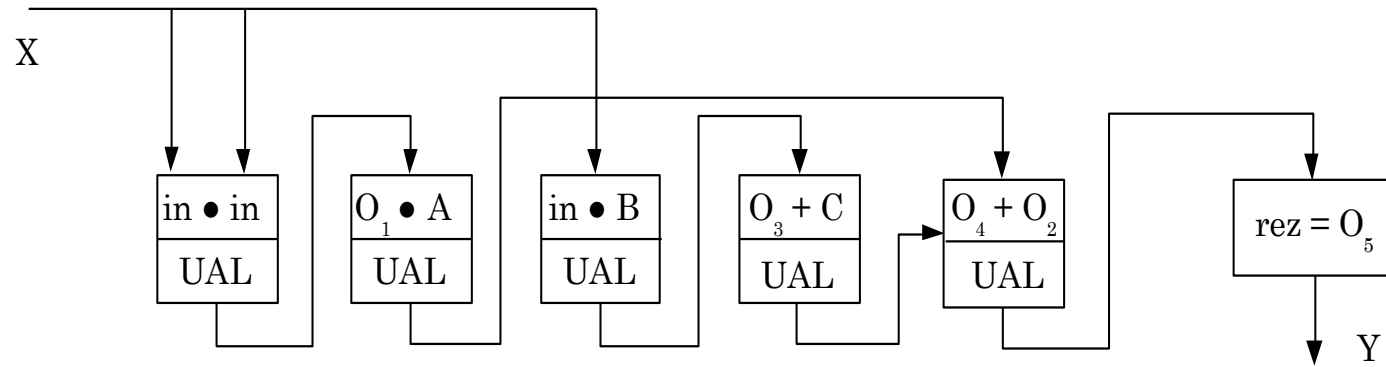


Soluție secvențială (temporală)

$t_1 \leftarrow X$
 $t_2 \leftarrow A \bullet t_1$
 $t_2 \leftarrow t_2 + B$
 $t_2 \leftarrow t_2 \bullet t_1$
 $y \leftarrow t_2 + C$



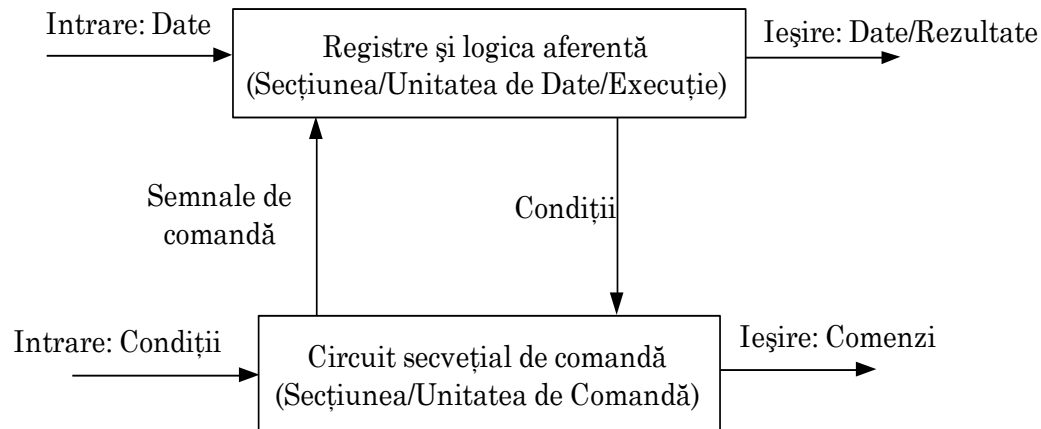
Implementarea spațială configurabilă a expresiei $y = Ax^2 + Bx + C$



Convenții de proiectare

Un sistem numeric poate fi partiționat în:

- unitatea de execuție ★
- unitatea de comandă ★ ★



★ asigură prelucrarea datelor , reprezentate sub forma unor vectori binari, în cadrul transferului acestora între registrele sursă și registrele destinație. Transferul se efectuează prin intermediul unor rețele/circuite logice combinaționale.

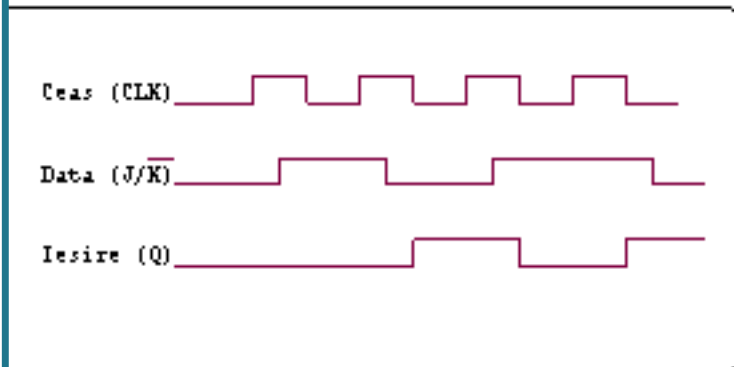
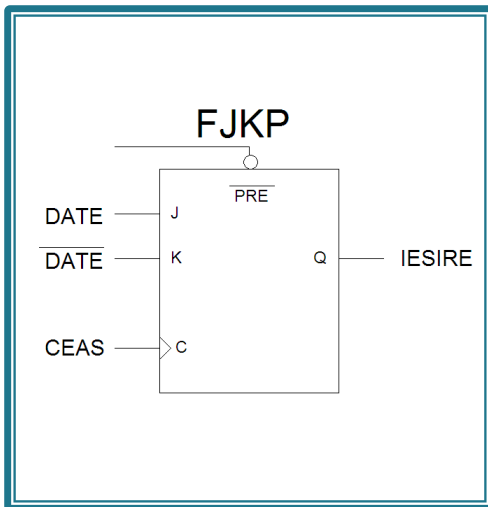
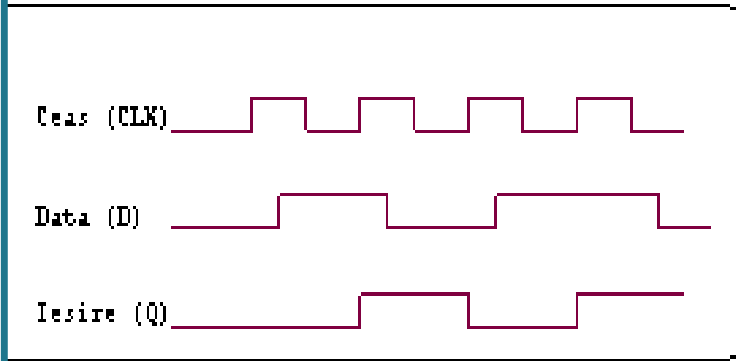
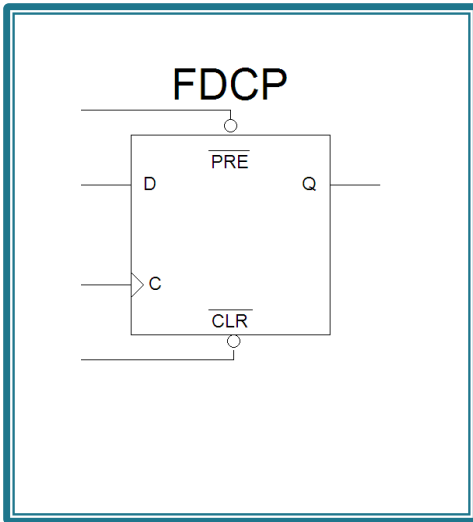
★ ★ furnizează, pentru secțiunea de date, diverse semnale de comandă, sincrone cu ceasul sistemului.

Transferuri între registre

Transferul conținutului unui registru sursă într-un registru destinație, fără a afecta conținutul sursei, se notează astfel: **AC = RD**

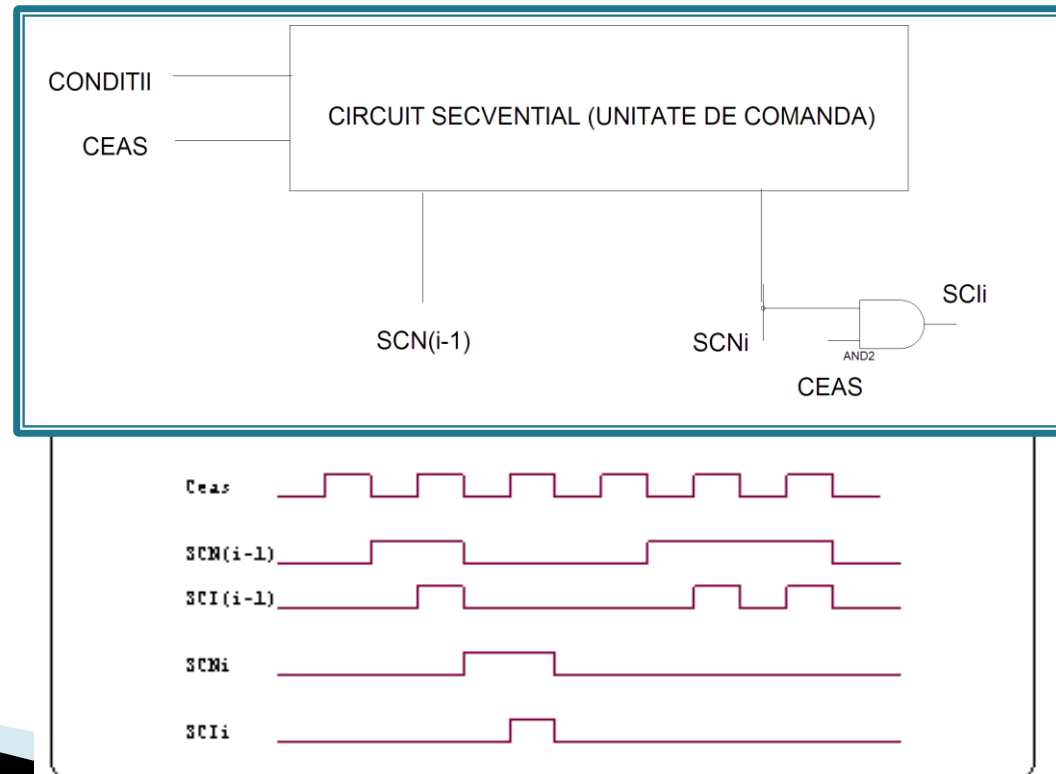
Dacă registrul AC are patru biți, anularea/forțarea în unu a tuturor bistabililor se poate nota după cum urmează: **AC = 4'b 0000; AC = 4'b 1111.**

- ➡ • Implementarea registrelor se bazează pe bistabile JK și D, cu intrări de sincronizare de ceas, *CLK*, și cu intrări de tip $\overline{PRESET} / \overline{CLR}$.
- ➡ • Transferurile sincrone au loc sub controlul semnalului de ceas pe fronturile anterior, pentru bistabilele de tip D, sau pe frontul posterior, pentru bistabilele master-slave de tip JK.

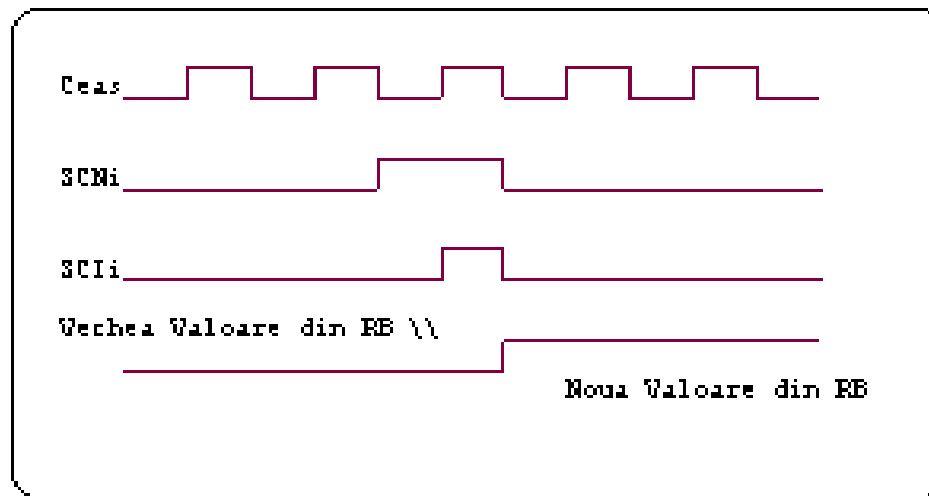


Un circuit secvențial de comandă furnizează, pentru secțiunea de date, diverse semnale de comandă, sincrone cu ceasul sistemului, cu perioade egale cu durata unei perioade de ceas sau cu multipli ai acesteia. ➡ **semnalele de comandă de tip nivel (SCN)**

Semnalele de comandă de tip nivel (SCN) vor avea un sufix numeric i ce va specifica numărul ieșirii de comandă a automatului ($SCNi$). Semnalele $SCNi$ pot fi strobate/eșantionate cu semnalul curent de ceas, pentru a forma **semnale de comandă de tip impuls ($SCIi$)**, cu durata activă corespunzătoare semnalului curent de ceas.



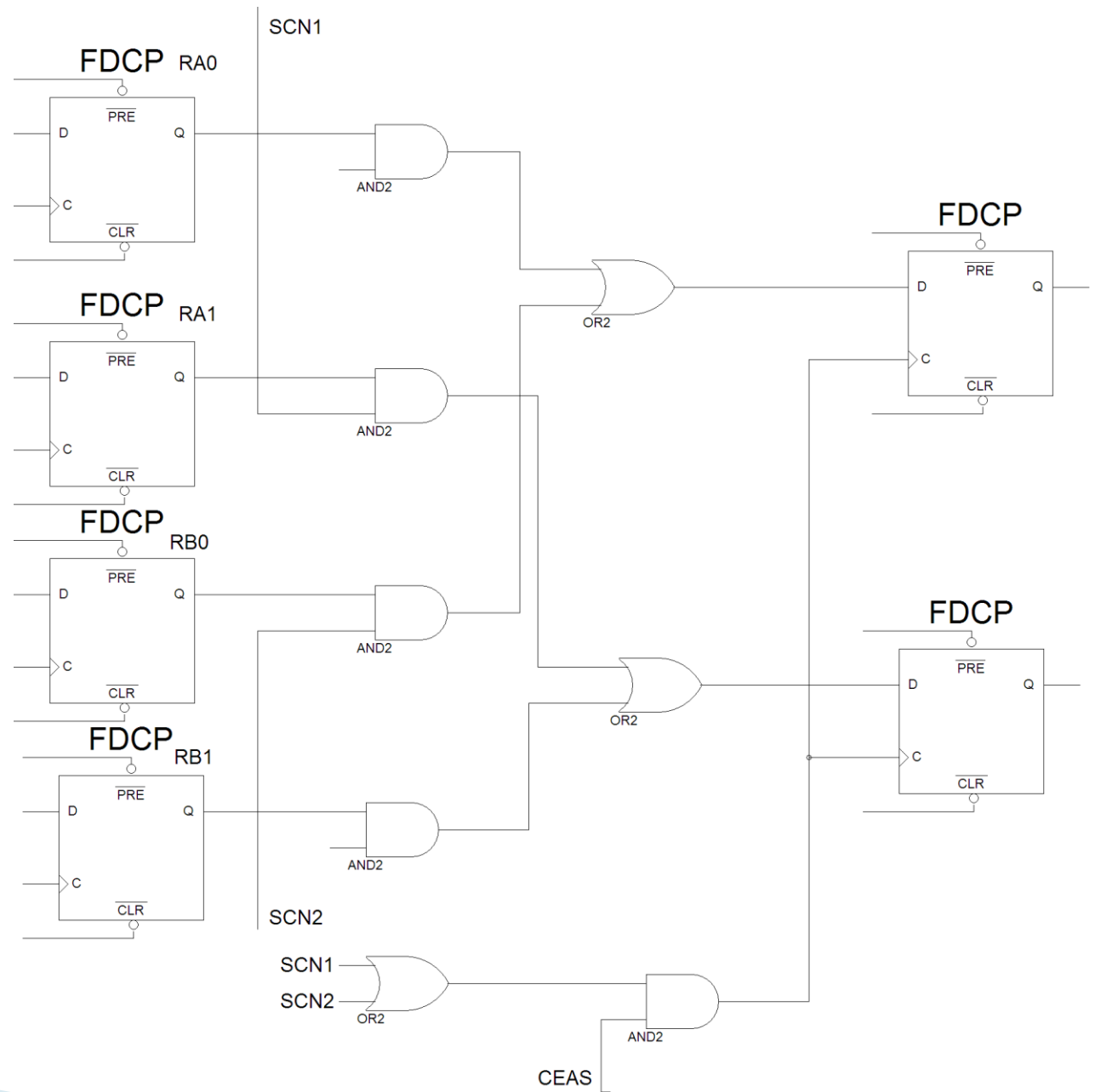
Diagramele de timp ale semnalelor implicate în transfer.



Exemplu:

Se dau următoarele transferuri pentru care se cere implementarea, la momentele 1, 2, sub controlul semnalelor SCN1, SCN2:

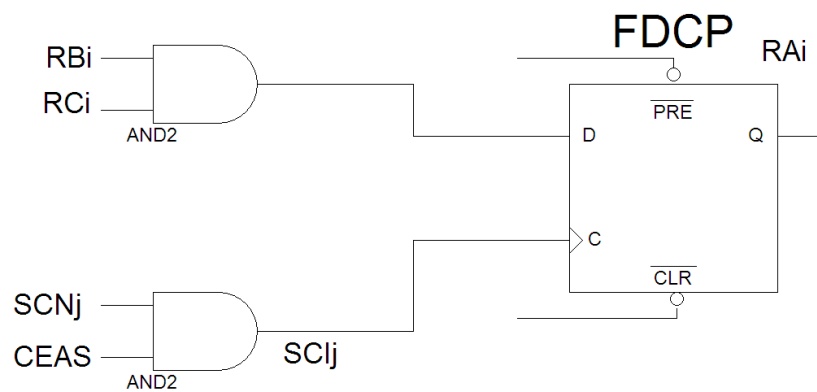
1. $RC = RA;$
2. $RC = RB;$



În cazul în care se urmărește implementarea operației elementare:

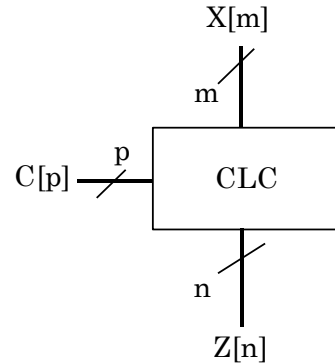
$$RA[i] = RB[i] \ \& \ RC[i] \ ;$$

se poate face transferul printr-o rețea logică combinațională ca mai jos:



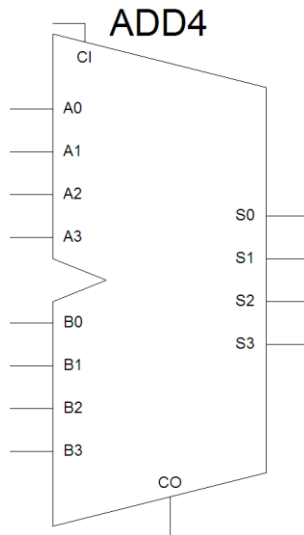
Componente combinaționale

Modelul general:



Operația Funcția	Descrierea	Comanda $C_{(p-1)} \dots C_0$
F_0	$Z = F_0(X)$	0 ... 0 0
F_1	$Z = F_1(X)$	0 ... 0 1
...
$F_{\binom{p}{2-1}}$	$Z = F_{\binom{p}{2-1}}(X)$	0 ... 0 0

Exemplu:



```

module sumator(a , b , ci , co , s);
output [3:0]s;
output co;
input [3:0] a, b;
input ci;

```

```

assign {co, s} = a+b+ci;

```

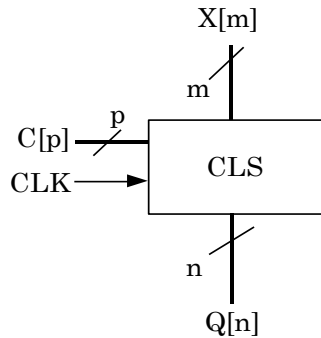
```

endmodule

```

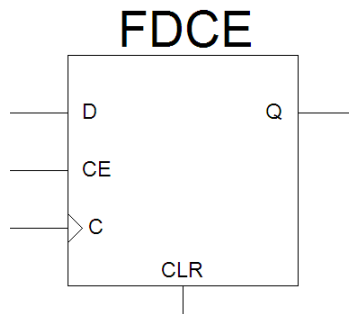
Componente secvențiale

Modelul general:



Operația Funcția	Descrierea	Comanda $C_{(p-1)} \dots C_0$
F_0	$Q = F_0(X)$	0 ... 0 0
F_1	$Q = F_1(X)$	0 ... 0 1
...
$F_{\binom{p}{2-1}}$	$Q = F_{\binom{p}{2-1}}(X)$	0 ... 0 0

Exemplu:



```

module bistabilD(d, ce, clk, clr, q);
output q;
reg q;
output co;
input d, ce, clk, clr;
always @(posedge clk)
    if (clr)
        q=0;
    else if (ce)
        q=d;
endmodule
    
```