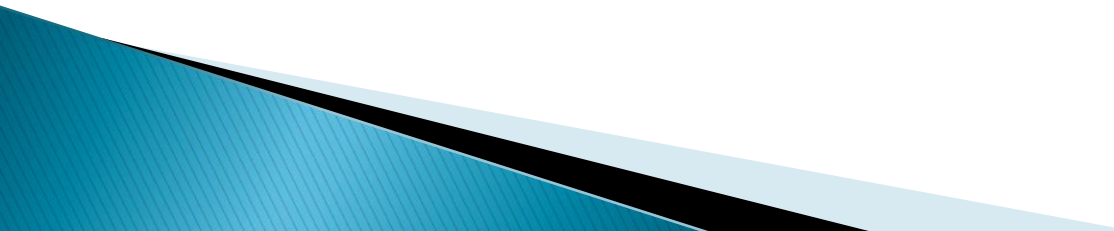


# Operații aritmetice

– *Curs9* –

## Subiecte abordate:

---

- Procesorul aritmetic
  - Operații aritmetice în virgulă fixă
  - Operații aritmetice în virgulă mobilă
- 

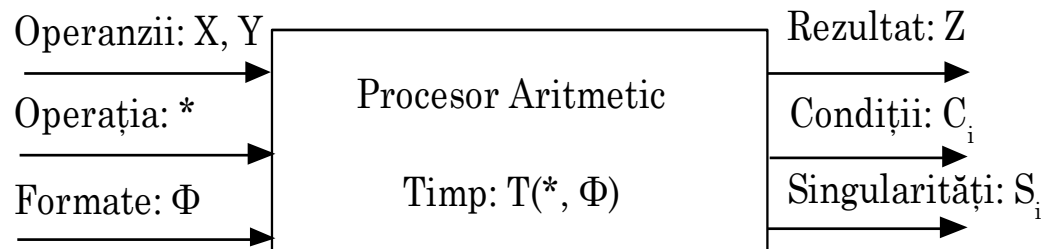
# Procesorul aritmetic

---

*Un procesor aritmetic reprezintă un dispozitiv capabil să efectueze operații simple sau complexe asupra unor operanzi furnizați în formate corespunzătoare.*

*Exemple:*

- Unitatea Aritmetică simplă;
- Incrementatorul;
- Dispozitivul pentru Transformata Fourier Rapidă



*Proiectarea aritmetică* pleacă de la specificațiile date de către utilizator și le transforma în specificații de operații aritmetice detaliate, la nivel de ranguri individuale/bit, în cadrul reprezentării concrete a datelor.



*Proiectarea logică* pleacă de la specificațiile furnizate de către proiectarea aritmetică și, în cadrul unei tehnologii date, selectează tipurile de circuite logice, pe care le interconectează, în mod corespunzător, în vederea implementării operațiilor aritmetice impuse de către algoritmi aritmetici.

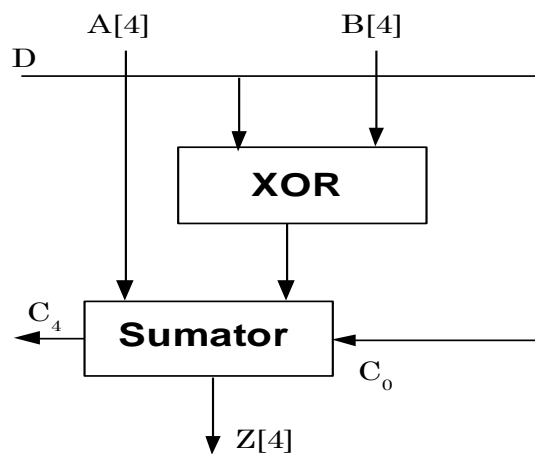


*proiectantul la nivel logic trebuie să elaboreze, atât proiectul unității de execuție, cât și proiectul unității de comandă.*

## Operații aritmetice în virgulă fixă

- *Operațiile de adunare și scădere* ale numerelor în virgulă fixă se implementează, în majoritatea covârșitoare a cazurilor, cu numere reprezentate în complementul față de doi.

Adunarea se efectuează rang cu rang, începând cu rangurile mai puțin semnificative, inclusiv rangurile de semn. Transportul, care apare la stânga rangului de semn, se neglijează.



Schema unui sumator/scăzător

Operația	Descrierea	Comanda $D = \Omega$
ADD	$Z = C_4, ADD(A, B)$	0
SUB	$Z = C_4, SUB(A, B)$	1

*Indicatorii de condiții* specifică o serie de proprietăți ale rezultatului, care apare în registrul acumulator al rezultatului AC.

→ sunt stocați în bistabile notate cu mnemonice, care formează un registru, încorporat în cuvântul de stare al programului/procesului.

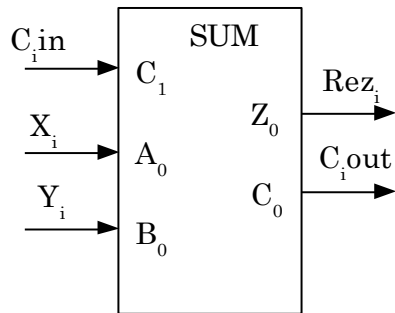
Indicatorii de condiții  
specifică diverse situații:

- rezultat = 0 - mnemonic Z;
- semnul rezultatului  $> 0$  sau  $< 0$  - mnemonica S
- apariția transportului, la stânga rangului de semn -  
mnemonica C
- rezultatul verificării parității - mnemonica P

## Implementarea unui sumator

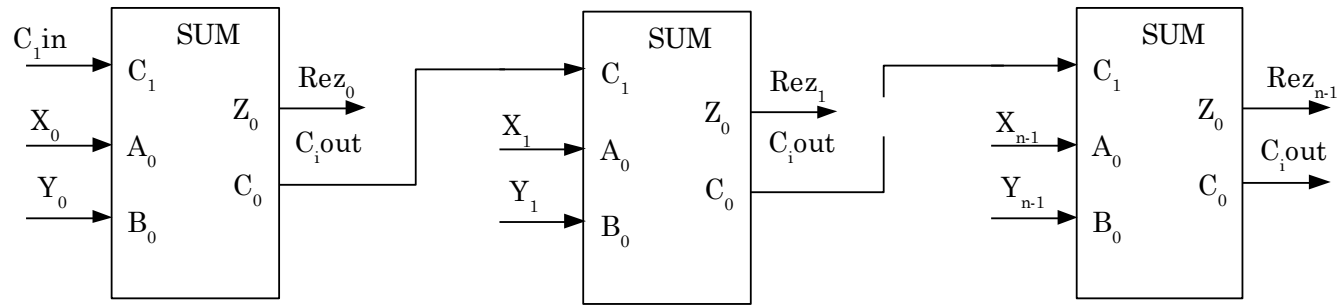
$$C_{out_i} = (x_i \cdot C_{in_i}) \cup (y_i \cdot C_{in_i}) \cup (x_i \cdot y_i)$$

$$sum_i = (x_i \cdot \bar{y}_i \cdot \bar{C}_{in_i}) \cup (\bar{x}_i \cdot y_i \cdot \bar{C}_{in_i}) \cup (\bar{x}_i \cdot \bar{y}_i \cdot C_{in_i}) \cup (x_i \cdot y_i \cdot C_{in_i})$$

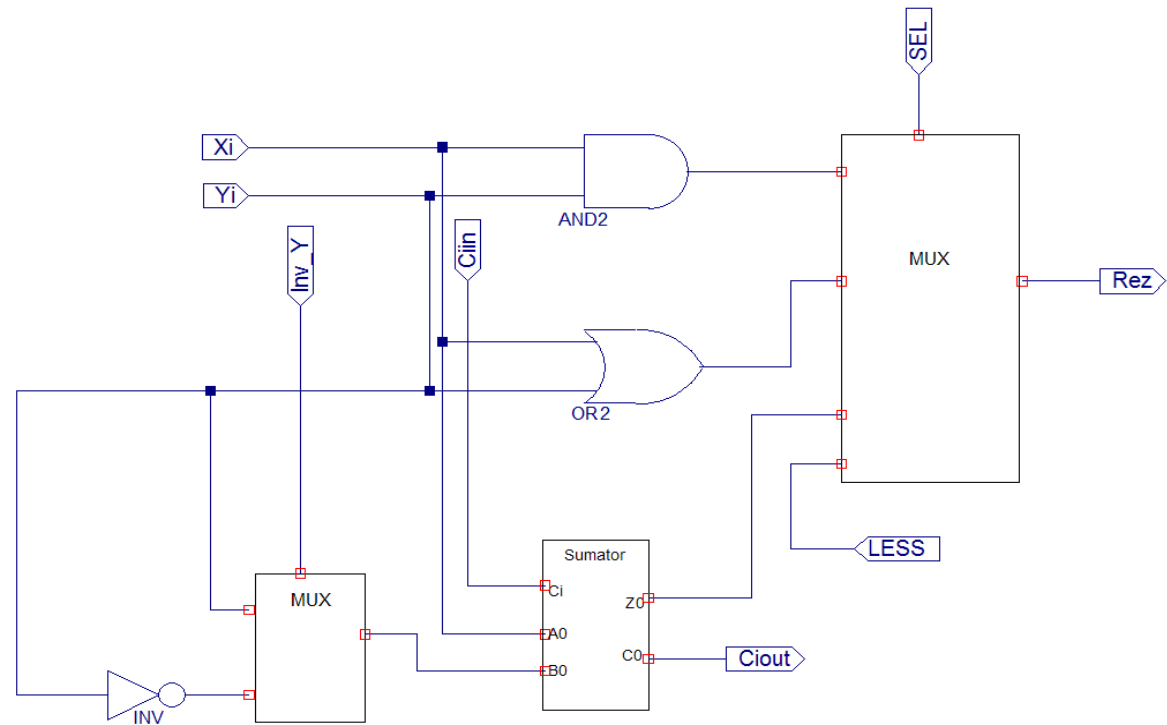


```
module fulladd(sum,carry,a,b,c);  
input a,b,c;  
output sum,carry;  
wire sum1;  
xor xor1(sum1,a,b);  
xor xor2(sum,sum1,c);  
and and1(c1,a,b);  
and and2(c2,b,c);  
and and3(c3,a,c);  
or or1(carry,c1,c2,c3);  
endmodule
```

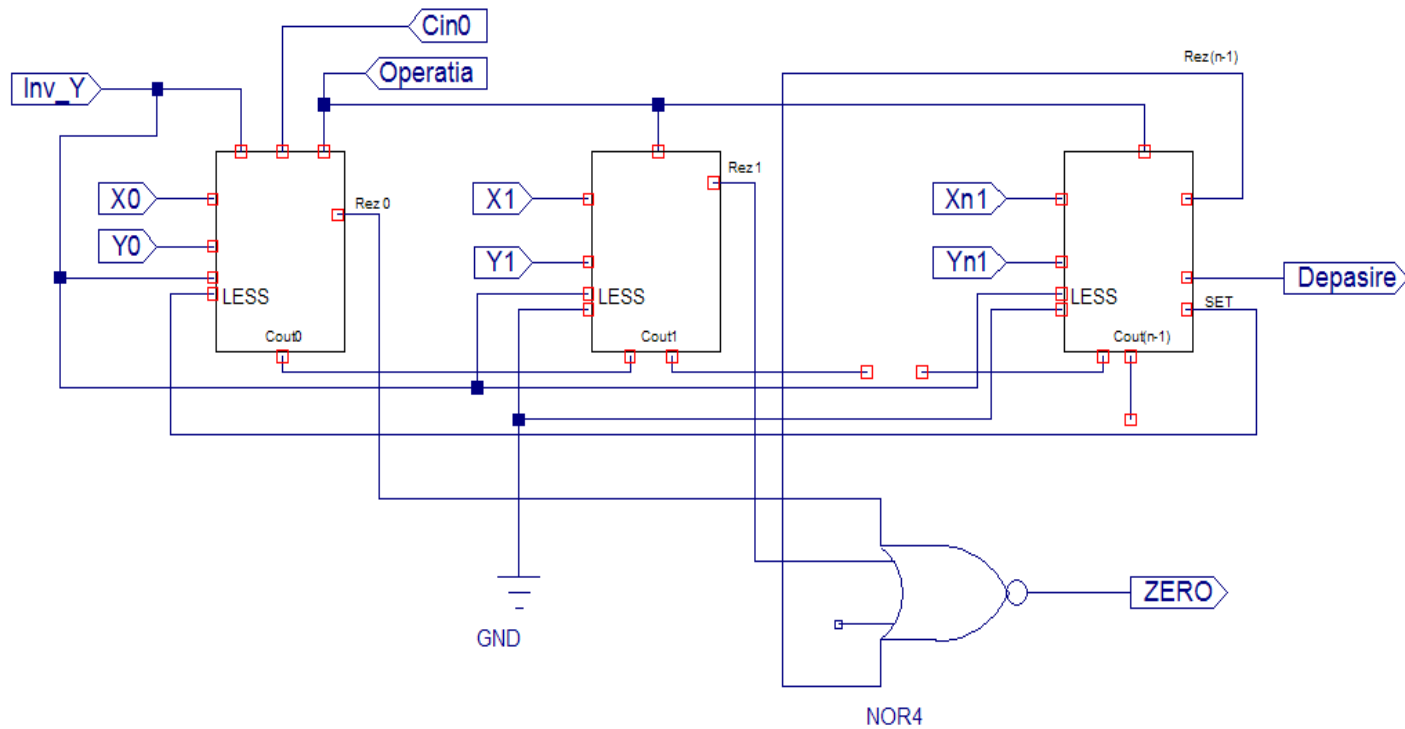
## Sumator pe $n$ biți



*Implementare completă, la nivel de bit, cu dăugarea operației de scădere*







*Implementarea operației “set-on-less-than” și poziționarea în “1”, a indicatorului de condiții, care specifică apariția unui rezultat egal cu “0”, la ieșirea unității aritmetice logice.*

## Sumatoare performante

---

### Sumatorul cu transport succesiv

tipul cel mai simplu de sumator paralel, obținut prin interconectarea unor sumatoare complete

$$C_{out_i} = (x_i \cdot C_{in_i}) \cup (y_i \cdot C_{in_i}) \cup (x_i \cdot y_i)$$

$$Sum_i = (x_i \cdot \bar{y}_i \cdot \overline{C_{in_i}}) \cup (\bar{x}_i \cdot y_i \cdot \overline{C_{in_i}}) \cup (\bar{x}_i \cdot \bar{y}_i \cdot C_{in_i}) \cup (x_i \cdot y_i \cdot C_{in_i})$$

### Sumatorul cu întârziere minimă

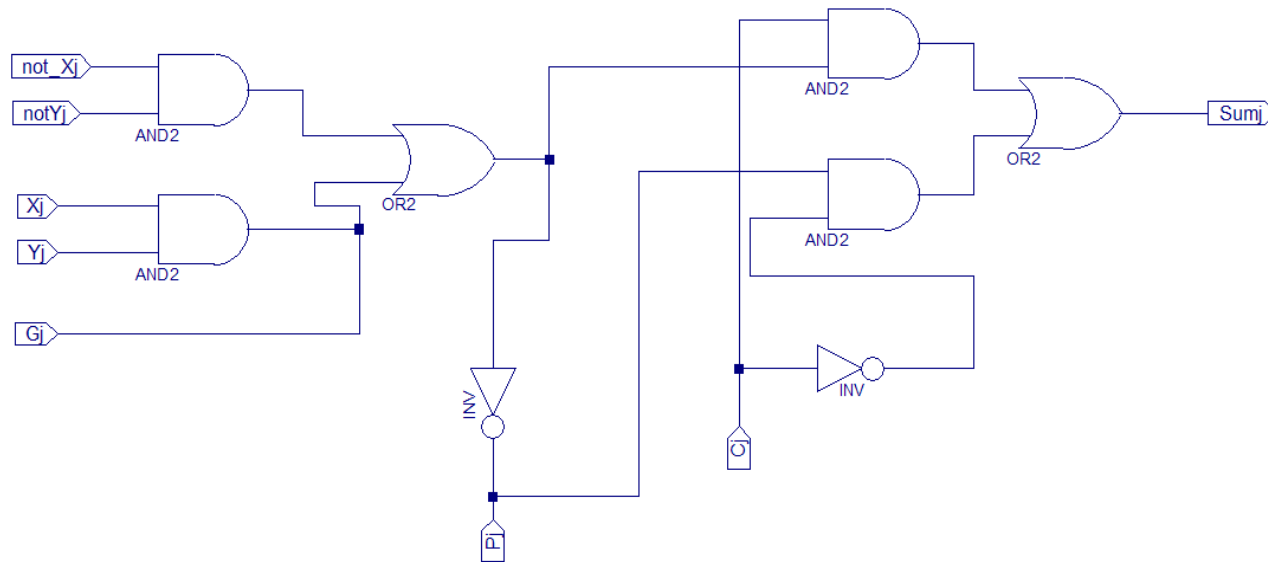
• considerând *ecuația sumei*, de la ieșirea celui mai puțin semnificativ rang, se încearcă stabilirea unei expresii a acesteia ca funcție de intrările pentru rangul curent, cât și de intrările pentru rangurile mai puțin semnificative. În acest mod se va obține o expresie logică implementabilă în două trepte.

• procedeul de implementare a sumei la primul etaj, Sum 1, conduce la utilizarea unei porți OR cu 12 intrări și a 12 porți AND cu câte 4 intrări. Extinzând la următoarele ranguri, se ajunge la utilizarea unui număr mult prea mare de porți, practic de nerealizat în prezent.

## Sumatorul cu anticiparea transportului

$$C_{j+1} = G_j \cup (P_j \cdot C_j)$$
$$Sum_j = (P_j \cdot \bar{C}_j) \cup (\bar{P}_j \cdot C_j)$$

unde:  $\left\{ \begin{array}{l} P - \text{propagarea transportului} \\ G - \text{generarea transportului} \end{array} \right.$

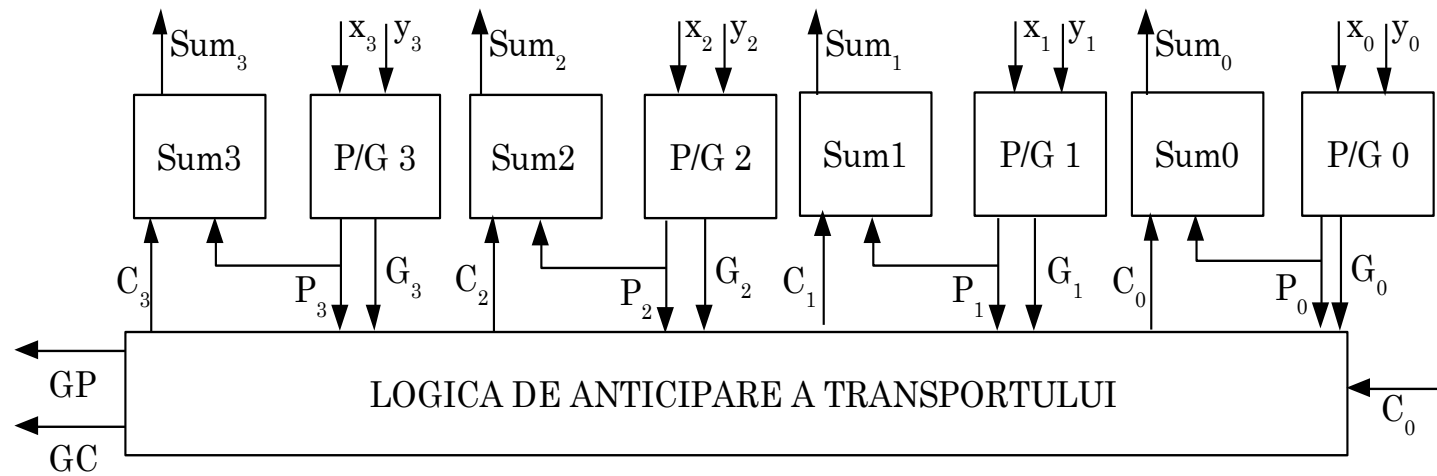


$$C_1 = G_0 \cup (P_0 \cdot C_0)$$

$$C_2 = G_1 \cup (P_1 \cdot C_1) = G_1 \cup (P_1 \cdot G_0) \cup (P_1 \cdot P_0 \cdot C_0)$$

$$C_3 = G_2 \cup (P_2 \cdot C_2) = G_2 \cup (P_2 \cdot G_1) \cup (P_2 \cdot P_1 \cdot G_0) \cup (P_2 \cdot P_1 \cdot P_0 \cdot C_0)$$

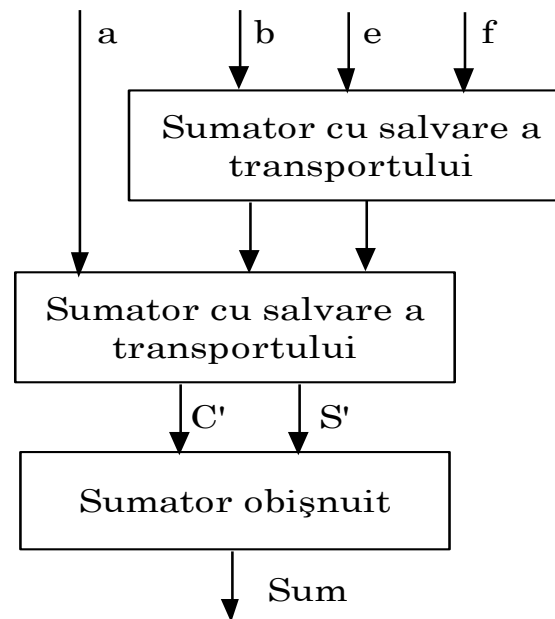
$$C_4 = G_3 \cup (P_3 \cdot C_3) = G_3 \cup (P_3 \cdot G_2) \cup (P_3 \cdot P_2 \cdot G_1) \cup (P_3 \cdot P_2 \cdot P_1 \cdot G_0) \cup (P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0)$$



*Secțiune pe 4 biți a unui sumator cu anticipare a transportului*

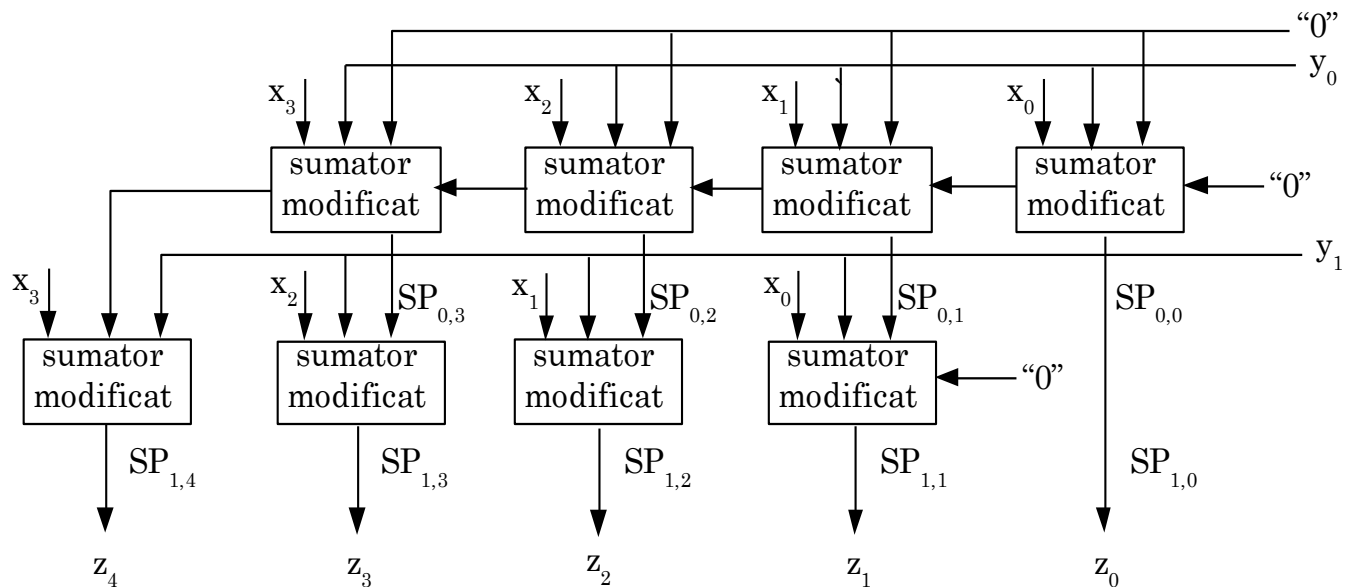
## Sumatorul cu salvare a transportului

în cazul adunării mai multor vectori binari simultan se poate recurge la o schemă mai rapidă, constând în utilizarea mai multor sumatoare în cascadă.



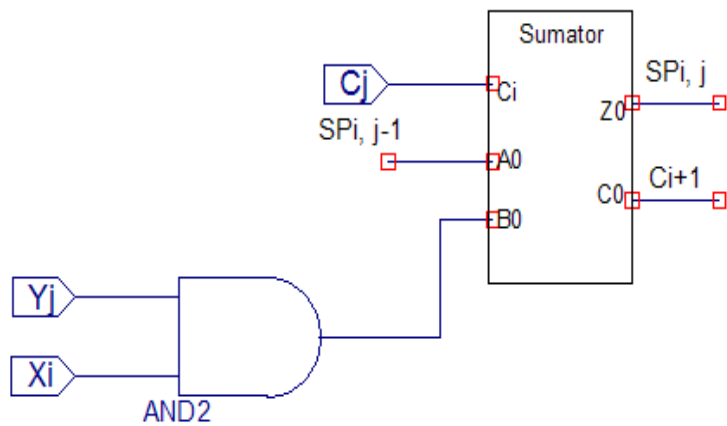
## •Înmulțirea

Constă în generarea și adunarea produselor parțiale obținute prin înmulțirea rangului curent al înmultitorului cu deînmulțitul.

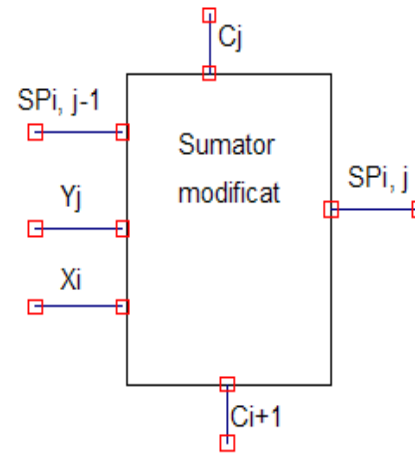


**Primele două etaje ale dispozitivului paralel de înmulțire.**

*S-a notat cu  $SP_{ij}$  bitul  $i$  al sumei produselor parțiale  $j$ .*



a) Sumator modificat



b) Schema bloc a sumatorului modificat

Sumatorul modificat poate fi utilizat în cadrul unei structuri de înmulțire paralelă, fiind vorba de o “programare spațială”, care conduce la viteza ridicată de operare.

## ALGORITMUL LUI BOOTH

$$Y = y_{n-1} \cdot 2^{n-1} + y_{n-1} \cdot 2^{n-2} + y_{n-3} \cdot 2^{n-3} + \dots + y_1 \cdot 2^1 + y_0 \cdot 2^0 = \sum_{i=0}^{n-1} (y_{i-1} - y_i) \cdot 2^i$$

unde:

- $y_{n-1}$  reprezintă rangul de semn, codificat cu 0/1 în cazul numerelor pozitive/negative;
- $y_{-1}$  constituie rangul aflat la dreapta rangului 0, având inițial valoarea 0.



$$X \times Y = \sum_{i=0}^{n-1} x \cdot (y_{i-1} - y_i) \cdot 2^i$$

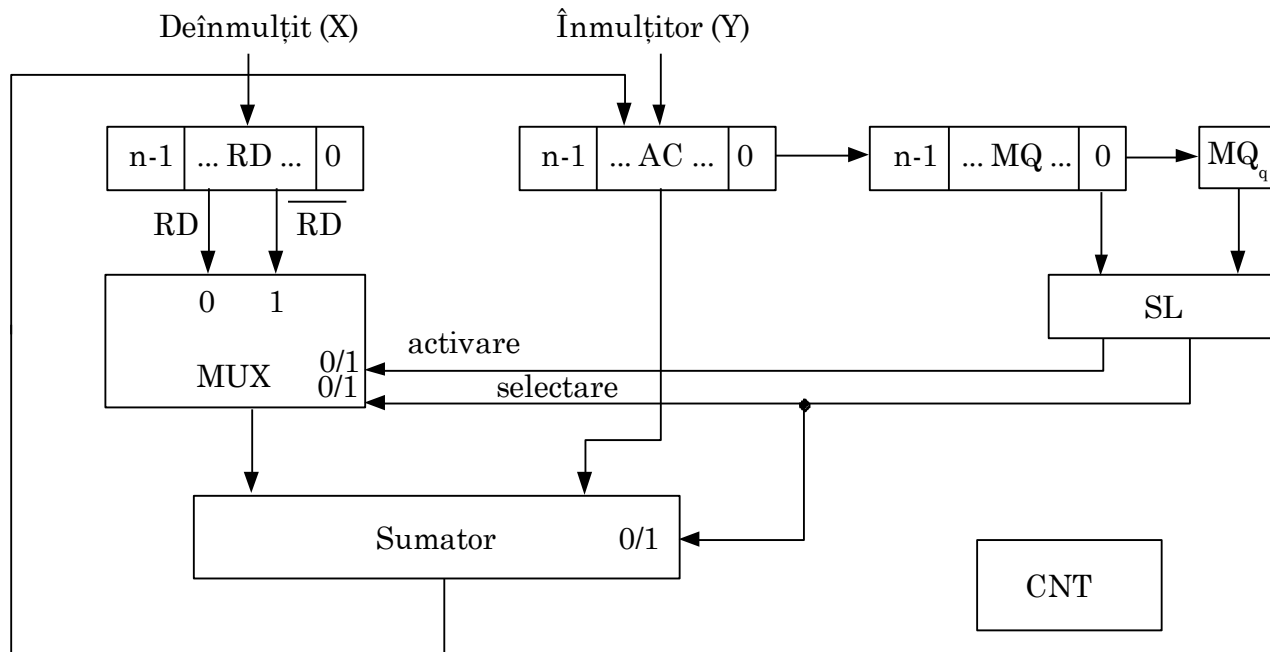


produsul parțial de rang  $i$  va fi:

$y_{i-1}$	$y_i$	produs parțial
0	0	0
0	1	$-x \cdot 2^i$
1	0	$x \cdot 2^i$
1	1	0



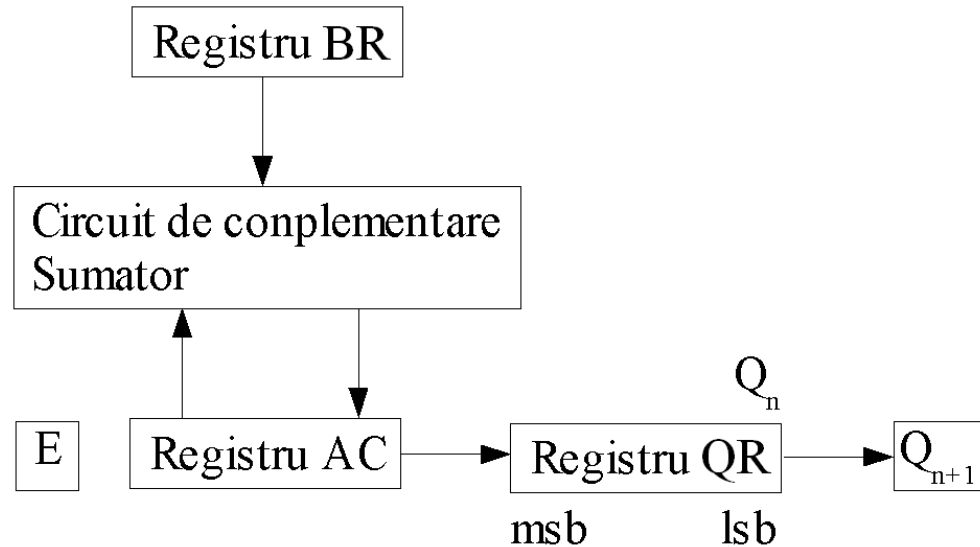
# Implementarea hardware a algoritmului lui Booth, structură orientată pe un singur acumulator



Resurse  
hardware

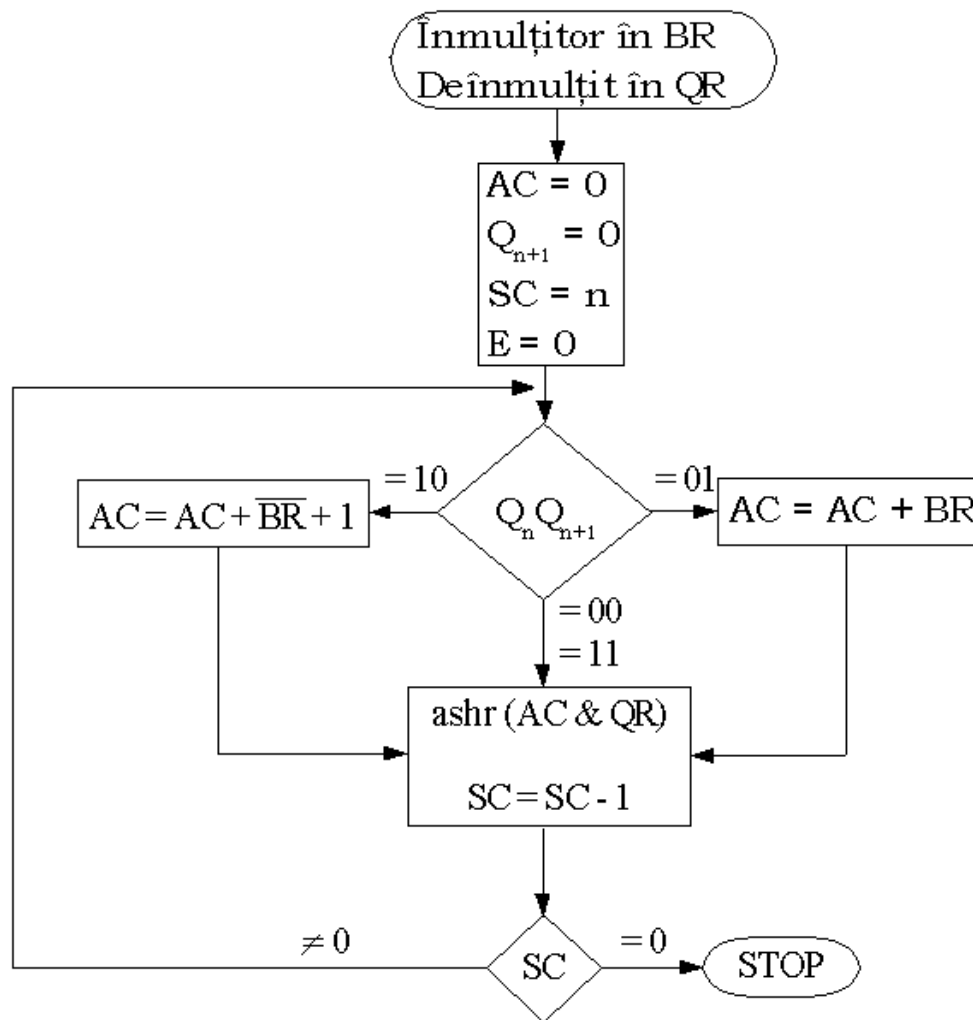
- AC – registru acumulator;
- RD – registru de date al memoriei;
- MQ – registru de extensie al acumulatorului;
- MQ<sub>q</sub> – registru de un bit, extensia lui MQ;
- CNT – contor de cicluri;
- SL – structura logică pentru activarea și selectarea intrărilor multiplexorului, cât și pentru controlul transportului la sumator;
- Sumator- sumator combinațional cu  $n$  ranguri binare.

Altă  
variantă:



- **Registru BR** – registru care menține valoarea înmulțitorului;
- **E** – registru de 1 bit folosit pentru memorarea valorii transportului rezultat în urma operației de adunare;
- **Registru AC** – registru auxiliar de lucru;
- **Registru QR** – registru care menține valoarea de înmulțitului

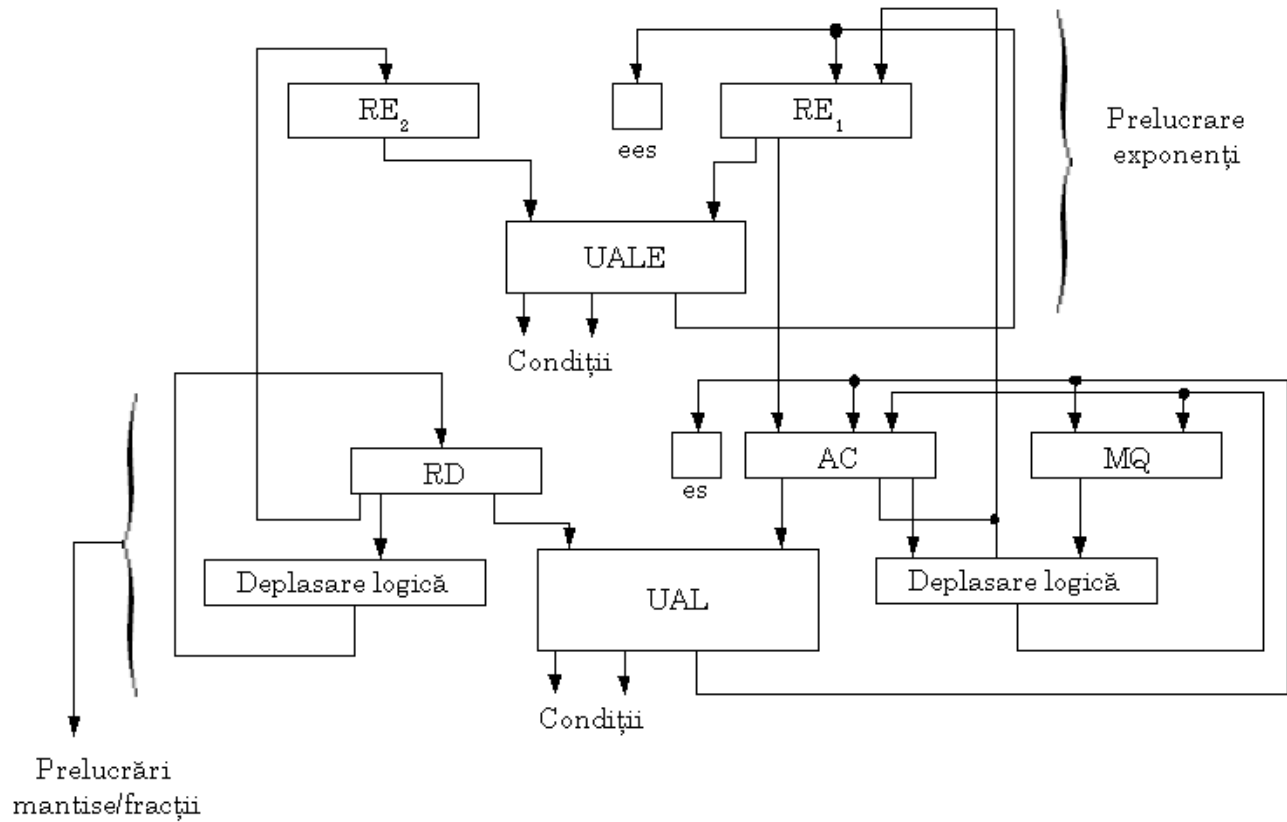
Organigrama pentru algoritmul lui Booth de înmulțire a două numere binare cu semn.



$Q_n$	$Q_{n+1}$	$\overline{BR} + 1 = 01001$	AC	QR	$Q_{n+1}$	SC
		Inițial	0000	10011	0	101
1	0	se scade BR	01001			
			<b>01001</b>			
		ashr	00100	11001	1	100
1	1	ashr	00010	01100	1	011
0	1	se adună BR	10111			
			<b>11001</b>			
		ashr	11100	10110	0	010
0	0	ashr	11110	01011	0	001
1	0	se scade BR	01001			
			<b>00111</b>			
		ashr	00011	10101	1	000

Exemplul numeric prezentat presupune că înmulțitorul are valoarea -9 și deînmulțitul are valoarea -13 (-9 x -13).

# OPERAȚII ARITMETICE ÎN VIRGULĂ MOBILĂ



Schema bloc a unității aritmetice în virgulă mobilă

Organigrama  
operației  
adunare/scădere  
în virgulă mobilă

