# 6. UNITATEA ARITMETICĂ ŞI LOGICĂ

Unitatea aritmetică şi logică (UAL) este acea parte a calculatorului care efectuează operaţiile aritmetice şi logice specificate de instrucţiuni asupra datelor.

Figura 6.1 prezintă modul general de interconectare a UAL cu restul UCP.

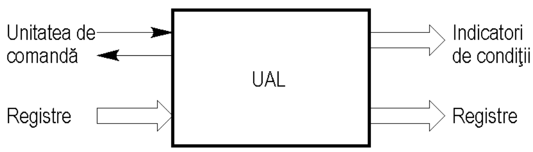


**Figura 6.1.** Intrările şi ieşirile UAL.

Datele sunt încărcate înaintea operaţiei în registrele UCP. Aceste registre sunt conectate prin căi de date la UAL. În urma decodificării instrucţiunii se transmite de unitatea de comandă succesiunea semnalelor de comandă necesare pentru execuţia operaţiei respective. După efectuarea operaţiei, se transmite un semnal de la UAL către unitatea de comandă, pentru a-l informa asupra terminării execuţiei. Rezultatele sunt depuse în registre, UAL poziţionând indicatorii de condiţii ca rezultat al operaţiei. De exemplu, va fi setat la 1 un indicator de depăşire dacă rezultatul depăşeşte lungimea registrului în care trebuie memorat rezultatul.

## 6.1. Circuite pentru adunarea a două cifre binare

Considerăm două numere, $A$ şi $B$, de câte $n$ biţi fiecare:

$A$: $A_{n-1} \ldots A_i A_{i-1} \ldots A_1 A_0$
$B$: $B_{n-1} \ldots B_i B_{i-1} \ldots B_1 B_0$

Pentru adunarea bit cu bit a numerelor, este nevoie de un circuit de adunare a două cifre binare de un anumit rang $i$, $A_i$ şi $B_i$, care să ţină cont de cifra de transport de la rangul anterior $i$-1, $T_{i-1}$. Circuitul trebuie să genereze cifra sumă $S_i$ şi cifra de transport $T_i$ către rangul următor.

Dacă nu se ține cont de transportul $T_{i\text{-}1}$, circuitul se numeşte *semisumator elementar*. Are două intrări, $A_i$ şi $B_i$, şi două ieşiri, $S_i$ şi $T_i$. Reprezentarea simbolică este prezentată în Figura 6.2.

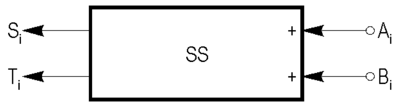


**Figura 6.2.** Reprezentarea simbolică a semisumatorului elementar.

Tabelul de adevăr al acestui circuit este prezentat în Tabelul 6.1.

**Tabelul 6.1.** Tabelul de adevăr al semisumatorului elementar.

| $A_iB_i$ | $S_i$ | $T_i$ |
|---|---|---|
| 0 0 | 0 | 0 |
| 0 1 | 1 | 0 |
| 1 0 | 1 | 0 |
| 1 1 | 0 | 1 |

Ecuațiile logice ale ieşirilor sunt următoarele:

$$S_i = \overline{A_i}B_i + A_i\overline{B_i} = A_i \oplus B_i \tag{6.1}$$

$$T_i = A_iB_i \tag{6.2}$$

Schema logică rezultată este prezentată în Figura 6.3.

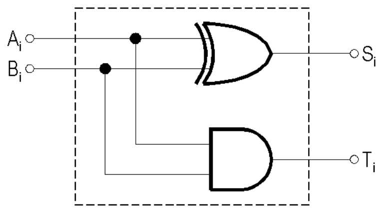


**Figura 6.3.** Schema logică a semisumatorului elementar.

Funcția SAU EXCLUSIV se poate implementa prin porți ŞI-NU, prin transformarea ecuației pentru $S_i$, ținând cont de legile *De Morgan*:

$$\overline{x + y} = \overline{x}\,\overline{y} \tag{6.3}$$

$$\overline{xy} = \overline{x} + \overline{y} \tag{6.4}$$

Se obține:

$$\begin{aligned} S_i &= \overline{A_i}B_i + A_i\overline{B_i} = B_i(\overline{A_i}+\overline{B_i}) + A_i(\overline{A_i}+\overline{B_i}) = \\ &= \overline{\overline{B_i(\overline{A_i}+\overline{B_i})}\cdot\overline{A_i(\overline{A_i}+\overline{B_i})}} = \overline{\overline{B_i\cdot\overline{A_iB_i}}\cdot\overline{A_i\cdot\overline{A_iB_i}}} \end{aligned} \tag{6.5}$$

Rezultă schema logică din Figura 6.4.

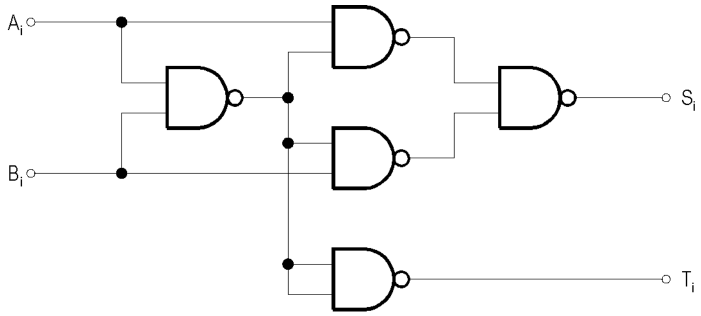


**Figura 6.4.** Schema logică a semisumatorului elementar implementat cu porți ŞI-NU.

În cazul în care se ține cont de cifra de transport de la rangul anterior, circuitul se numeşte *sumator elementar*. Are trei intrări, $A_i$, $B_i$, $T_{i\text{-}1}$, şi două ieşiri, $S_i$ şi $T_i$. Reprezentarea simbolică este dată în Figura 6.5.

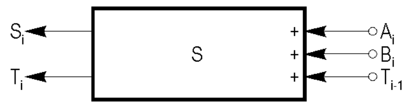


**Figura 6.5.** Reprezentarea simbolică a sumatorului elementar.

Tabelul de adevăr al circuitului este prezentat în Tabelul 6.2.

**Tabelul 6.2.** Tabelul de adevăr al sumatorului elementar.

| $A_iB_iT_{i-1}$ | $S_i$ | $T_i$ |
|---|---|---|
| 0 0 0 | 0 | 0 |
| 0 0 1 | 1 | 0 |
| 0 1 0 | 1 | 0 |
| 0 1 1 | 0 | 1 |
| 1 0 0 | 1 | 0 |
| 1 0 1 | 0 | 1 |
| 1 1 0 | 0 | 1 |
| 1 1 1 | 1 | 1 |

Ecuațiile logice care rezultă sunt următoarele:

$$S_i = \overline{A_i}\,\overline{B_i}T_{i-1} + \overline{A_i}B_i\overline{T_{i-1}} + A_i\overline{B_i}\,\overline{T_{i-1}} + A_iB_iT_{i-1} \tag{6.6}$$

$$T_i = \overline{A_i}B_iT_{i-1} + A_i\overline{B_i}T_{i-1} + A_iB_i\overline{T_{i-1}} + A_iB_iT_{i-1} \tag{6.7}$$

De aici rezultă o primă formă a circuitului. O altă formă se poate obține grupând termenii din exprimarea funcției $S_i$, şi ținând cont că în ultimii doi termeni ai funcției $T_i$ apare atât $T_{i-1}$ cât şi $\overline{T_{i-1}}$ . Rezultă:

$$\begin{aligned} S_i &= (\overline{A_i}\,\overline{B_i} + A_iB_i)T_{i-1} + (\overline{A_i}B_i + A_i\overline{B_i})\overline{T}_{i-1} \\ &= \overline{(A_i \oplus B_i)}T_{i-1} + (A_i \oplus B_i)\overline{T}_{i-1} \\ &= (A_i \oplus B_i) \oplus T_{i-1} = A_i \oplus B_i \oplus T_{i-1} \end{aligned} \tag{6.8}$$

$$T_i = (\overline{A_i}B_i + A_i\overline{B_i})T_{i-1} + A_iB_i = (A_i \oplus B_i)T_{i-1} + A_iB_i \tag{6.9}$$

Deci:

$$S_i = A_i \oplus B_i \oplus T_{i-1} \tag{6.10}$$

$$T_i = (A_i \oplus B_i)T_{i-1} + A_iB_i \tag{6.11}$$

Din aceste ecuații rezultă posibilitatea realizării sumatorului elementar prin reunirea a două semisumatoare. Schema corespunzătoare este prezentată în Figura 6.6.

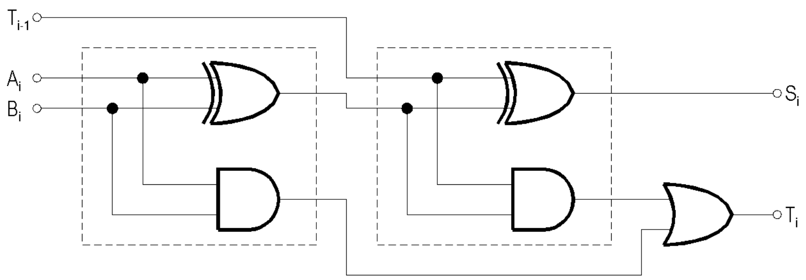


**Figura 6.6.** Schema logică a sumatorului elementar realizat prin reunirea a două semisumatoare elementare.

# 6.2. Operații cu numere în virgulă fixă

## 6.2.1. Adunarea

Operația de adunare se poate efectua în serie, când biții celor doi operanzi sunt prelucrați succesiv, sau în paralel, când biții sunt prelucrați simultan. Dispozitivele de adunare corespunzătoare sunt *sumatorul serie*, respectiv *sumatorul paralel*.

### 6.2.1.1. Sumatorul serie

Schema unui sumator serie de *n* biți este prezentată în Figura 6.7.

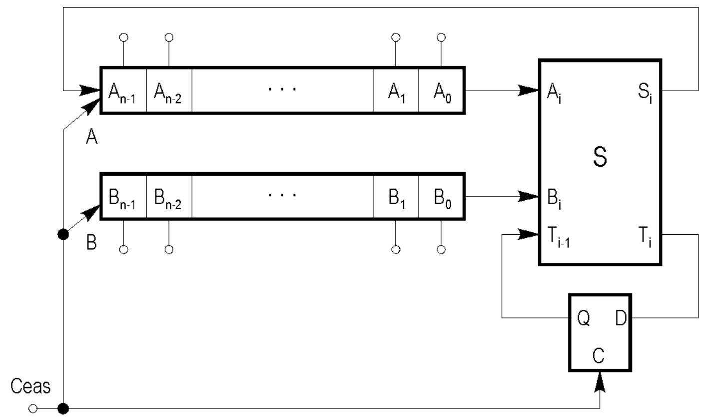


**Figura 6.7.** Sumator serie de *n* biți.

Sumatorul se compune din două registre de deplasare *A* şi *B*, de câte *n* biți fiecare, cu posibilitatea de încărcare paralelă, un singur sumator elementar *S*, şi un element de întârziere de un tact, format dintr-un bistabil *D*. Operanzii de câte *n* biți se încarcă în paralel în registrele *A* şi *B*. Bistabilul *D* este şters, astfel încât inițial $T_{i-1}$ va fi 0.

Biții operanzilor se aplică la intrările sumatorului în mod secvențial, inițial aplicându-se biții c.m.p.s. ($A_0$, $B_0$). În fiecare moment *i*, sumatorul generează suma $S_i$ şi cifra de transport $T_i$. La fiecare impuls de ceas, registrele se deplasează la dreapta cu o poziție, aplicând la intrările sumatorului următorii biți, iar suma $S_i$ se introduce în registrul în care s-a păstrat unul din operanzi, pe poziția care s-a eliberat (se utilizează intrarea serie în acest scop). În acelaşi timp, transportul $T_i$ se introduce în elementul de întârziere cu un impuls de ceas (bistabilul *D*, care repetă intrarea cu o întârziere de un impuls de ceas), pentru utilizarea lui la impulsul de ceas următor, când $T_i$ se transformă în cifra $T_{i-1}$ a rangului următor.

Întregul proces are loc în *n* impulsuri de ceas, obținându-se rezultatul în registrul *A*. După *n* impulsuri de ceas, bistabilul *D* conține transportul de la rangul cel mai semnificativ. Dacă acest transport este 1, suma are *n*+1 biți, deci are loc o depăşire a capacității registrului sumă *A*.

Registrul *A* conține atât unul din operanzi, cât şi rezultatul. Acest registru reprezintă deci un *acumulator*, iar această structură se numeşte *structură cu acumulator*.

### 6.2.1.2. Sumatorul paralel

Adunarea a doi operanzi se poate efectua în paralel dacă biții operanzilor sunt accesibili în paralel. Schema unui sumator paralel de *n* biți este prezentată în Figura 6.8.

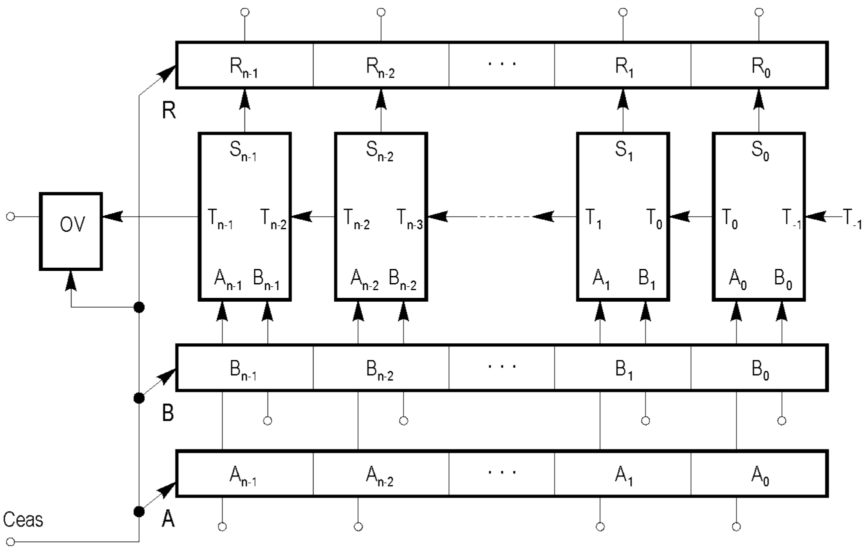


**Figura 6.8.** Sumator paralel de *n* biţi.

Sumatorul este format din două registre paralele *A* şi *B*, pentru cei doi operanzi de *n* biţi, un registru *R* care păstrează suma, şi *n* sumatoare elementare. Fiecare sumator de rang *i* primeşte de la rangul *i*-1 transportul $T_{i-1}$ şi generează transportul $T_i$ pentru rangul următor.

Primul sumator, cel corespunzător cifrei c.m.p.s., poate fi înlocuit cu un semisumator, deoarece nu există transport de la rangul precedent. Transportul de la rangul c.m.s. este folosit pentru a indica depăşirea de capacitate a registrului sumă. Acest transport se păstrează în bistabilul *OV* (*Overflow*). Intrarea $T_{-1}$ şi ieşirea $T_{n-1}$ se pot utiliza pentru extinderea sumatorului la un număr mai mare de biţi.

La impulsul de ceas se înscriu operanzii în registrele *A* şi *B*. După un timp de propagare, la ieşirile sumatoarelor elementare apare suma şi transportul la ieşire. La următorul impuls de ceas, rezultatul se înscrie în registrul *R*.

Se poate utiliza şi în acest caz un registru acumulator, pentru a păstra atât unul din operanzi, cât şi rezultatul. Un asemenea sumator paralel, obţinut prin interconectarea unor sumatoare elementare, se numeşte *sumator cu transport succesiv*.

Sumatorul paralel efectuează adunarea pe un impuls de ceas, iar cel serie pe *n* impulsuri de ceas. Totuşi, sumatorul paralel nu este de *n* ori mai rapid decât cel serie, deoarece transportul se propagă în serie prin cele *n* sumatoare. Deci, trebuie prevăzut un timp suficient pentru ca cifra de transport să se propage în cazul cel mai defavorabil (când acesta se propagă prin toate nivelele). Frecvenţa ceasului trebuie aleasă în funcţie de acest timp de propagare.

## 6.2.2. Înmulțirea binară

Există numeroase metode de înmulțire binară. Cele mai importante sunt următoarele:

- Metoda adunării repetate;
- Înmulțirea pe grupe de cifre;
- Înmulțirea simultană.

Fiecare din aceste metode are mai multe variante, care țin cont de modul de reprezentare a numerelor cu semn sau urmăresc creşterea vitezei de execuție a operației. Metoda adunării repetate este cea mai utilizată, şi se bazează pe adunarea repetată a unor produse parțiale. Dintre variantele acestei metode se amintesc următoarele:

- Metoda înmulțirii directe;
- Metoda *Burks-Goldstine-von Neumann*;
- Metoda *Robertson*;
- Metoda *Booth*;
- Metoda înmulțirii scurte.

Unele metode se pot utiliza numai pentru numere fără semn, pe când altele se pot utiliza şi pentru numere cu semn.

### 6.2.2.1. Metoda înmulțirii directe

Metoda se poate aplica pentru numere reprezentate în mărime şi semn. Cifra de semn este tratată separat. Dacă numerele sunt reprezentate în complemente, aplicarea metodei necesită complementarea operandului sau operanzilor negativi, pentru ca reprezentarea cifrelor de mărime să fie aceeaşi ca în MS, şi complementarea rezultatului, dacă acesta este negativ. Acest procedeu este recomandat în cazul numerelor reprezentate în C1, când complementarea se realizează simplu. Pentru numerele reprezentate în C2, se preferă utilizarea altor metode care operează direct asupra numerelor în această reprezentare.

Notând cu $X$ deînmulțitul şi cu $Y$ înmulțitorul, aceştia se pot exprima în mărime şi semn sub forma următoare:

$$X = x_n 2^n + \sum_{i=0}^{n-1} x_i 2^i = x_n 2^n + x \tag{6.12}$$

$$Y = y_n 2^n + \sum_{i=0}^{n-1} y_i 2^i = y_n 2^n + y \tag{6.13}$$

unde $x_n$, $y_n$ sunt cifrele de semn, iar $x$, $y$ reprezintă mărimile numerelor.

Produsul se poate scrie sub forma:

$$Z = X \cdot Y = z_n 2^{2n} + z \tag{6.14}$$

unde $z_n$ reprezintă semnul produsului,

$$z_n = x_n \oplus y_n \tag{6.15}$$

iar $z$ reprezintă mărimea produsului:

$$z = x \cdot y = xy_{n-1}2^{n-1} + xy_{n-2}2^{n-2} + \ldots + xy_1 2^1 + xy_0 2^0 \tag{6.16}$$

Deoarece coeficienții $y_i$ sunt fie 0, fie 1, înmulțirea binară este de fapt o adunare repetată a unor produse parțiale, care sunt egale fie cu 0, fie cu deînmulțitul deplasat la stânga, conform cu ponderile coeficienților $y_i$.

Produsele parțiale se pot scrie sub forma:

$$\begin{aligned} p_0 &= xy_0 2^0 \\ p_1 &= p_0 + xy_1 2^1 \\ p_2 &= p_1 + xy_2 2^2 \\ &\ldots \\ p_{n-2} &= p_{n-3} + xy_{n-2}2^{n-2} \\ p_{n-1} &= p_{n-2} + xy_{n-1}2^{n-1} \end{aligned} \tag{6.17}$$

Înlocuind produsele parțiale în expresia lui $p_{n-1}$, se obține:

$$p_{n-1} = x(y_{n-1}2^{n-1} + y_{n-2}2^{n-2} + \ldots + y_1 2^1 + y_0 2^0) = xy \tag{6.18}$$

Produsul final se poate scrie sub forma:

$$Z = (x_n \oplus y_n)\, 2^{2n} + p_{n-1} \tag{6.19}$$

unde $p_{n-1}$ se obține din relația de recurență:

$$p_{n-1} = p_{n-2} + xy_{n-1}2^{n-1} \tag{6.20}$$

cu $p_0 = xy_0 2^0$.

Fie $X$ = `12`, $Y$ = `6`. Considerând numere fără semn cu 4 biți de mărime, înmulțirea binară obişnuită a celor două numere se efectuează astfel:

```
   1100 ×          Deînmulțit (12)
   0110            Înmulțitor  (6)
   ----
   0000
  1100             Produse parțiale
 1100
0000
-------
1001000            Produs final (2^6 + 2^3 = 64+8 = 72)
```

Procesul constă din testarea succesivă a biților înmulțitorului, începând cu bitul cel mai puțin semnificativ. Dacă bitul testat este 1, se copiază deînmulțitul, în caz contrar fiind copiate zerouri.

Se pot formula următoarele observații:

1. Înmulțirea implică generarea unor produse parțiale, câte unul pentru fiecare bit al înmulțitorului. Aceste produse sunt apoi însumate pentru a se obține produsul final.
2. Dacă bitul testat al înmulțitorului este 0, produsul parțial corespunzător este 0. Dacă acest bit este 1, produsul parțial este egal cu deînmulțitul.
3. Pentru obținerea produsului final, fiecare produs parțial este deplasat la stânga cu o poziție relativ la produsul parțial precedent.
4. Înmulțirea a două numere binare fără semn de câte $n$ biți are ca rezultat un produs cu o lungime de până la $2n$ biți.

Față de înmulțirea obişnuită, se pot efectua unele modificări pentru ca operația să fie mai eficientă.

- Se poate realiza acumularea produselor parțiale într-un registru acumulator, deci adunarea lor pe măsură ce ele sunt obținute, în locul adunării tuturor produselor la final. Astfel se elimină necesitatea de a memora toate produsele parțiale, fiind necesare mai puține registre. După fiecare produs parțial obținut, deînmulțitul este deplasat la stânga cu o poziție.
- Pentru fiecare bit de 1 din înmulțitor, este necesară o adunare şi o deplasare. Pentru fiecare bit de 0 din înmulțitor, produsul parțial este 0, fiind necesară numai o deplasare.

Rezultă că operația de înmulțire se poate efectua conform următorului algoritm:

- Se inițializează produsul parțial cu zero.
- Se testează biții înmulțitorului, începând cu bitul cel mai puțin semnificativ. Dacă bitul testat este 1, se adună deînmulțitul la produsul parțial.
- După fiecare operație, la începutul pasului următor, se deplasează deînmulțitul la stânga cu o poziție.

Dacă numerele sunt cu semn, semnul rezultatului se poate stabili în funcție de semnul numerelor: dacă acestea au acelaşi semn, produsul este pozitiv, iar dacă numerele au semne contrare, produsul este negativ. Pentru numere cu $n$+1 biți (un bit pentru semn), produsul va avea $2n$+1 biți.

Produsul parțial se păstrează într-un registru acumulator. Acest registru necesită $2n$ poziții, unde $n$ este numărul biților de mărime ale deînmulțitului şi înmulțitorului. Deoarece deînmulțitul trebuie deplasat la stânga în fiecare pas al operației, pentru deînmulțit este necesar de asemenea un registru de lungime dublă.

### Exemplul 6.1

Considerăm din nou numerele fără semn $X$ = `12`, $Y$ = `6`, $n$ = `4`. Înmulțirea celor două numere, efectuată conform algoritmului de mai sus, este prezentată în Tabelul 6.3.

**Tabelul 6.3.** Exemplu de înmulţire prin metoda directă.

| Pas | A | X | Y |
|---|---|---|---|
| 0 | 0000 0000 | 0000 1100 | 0110 |
| 1 | 0000 0000 + | 0000 1100 | 0110 |
| 2 | 0001 1000<br>0001 1000 + | 0001 1000 | 0110 |
| 3 | 0011 0000<br>0100 1000 | 0011 0000 | 0110 |
| 4 | 0100 1000 | 0110 0000 | 0110 |

Rezultatul este: $0100\ 1000_2 = 48h = 4 \cdot 16+8 = 64+8 = 72$.

Acelaşi rezultat se obţine dacă nu se deplasează deînmulţitul la stânga, ci se deplasează produsul parţial la dreapta, astfel încât produsul parţial şi deînmulţitul să fie păstrate în aceeaşi poziţie relativă.

În locul testării înmulţitorului începând cu bitul cel mai puţin semnificativ, mai există posibilitatea ca testarea să se realizeze începând cu bitul cel mai semnificativ. În acest caz, se poate deplasa fie deînmulţitul la dreapta, fie produsul parţial la stânga.

Dintre aceste variante, cea mai avantajoasă este cea în care testarea înmulţitorului se realizează începând cu bitul c.m.p.s. şi produsul parţial este deplasat la dreapta. Avantajele acestei variante sunt următoarele:

- Este necesar un sumator de $n$ biţi, chiar dacă rezultatul va avea $2n+1$ biţi (inclusiv semnul).
- Deînmulţitul necesită un registru de $n+1$ biţi, în loc de $2n+1$ biţi.
- Registrul acumulator necesită $n+1$ biţi, în loc de $2n+1$ biţi.

Deoarece rezultatul are $2n+1$ biţi, registrul acumulator ar trebui completat cu un registru auxiliar pentru memorarea produsului complet. Deoarece după înmulţirea cu un bit al înmulţitorului acest bit nu mai este necesar, registrul acumulator se poate plasa în continuarea registrului înmulţitorului, formând un registru combinat care se deplasează la dreapta în fiecare pas al operaţiei. Astfel, în final registrul acumulator păstrează biţii c.m.s. ai rezultatului, iar registrul înmulţitorului păstrează biţii c.m.p.s.

Structura unui dispozitiv de tip paralel care implementează metoda înmulţirii directe este prezentată în Figura 6.9.

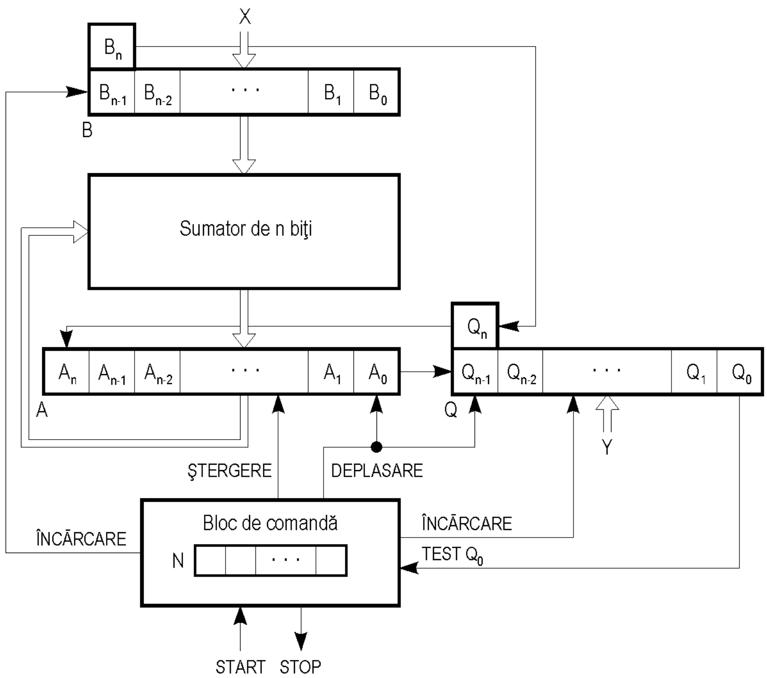


**Figura 6.9.** Schema bloc a unui dispozitiv de înmulţire prin metoda directă.

Dispozitivul de înmulţire cuprinde registrele A, B şi Q de $n$+1 biţi fiecare, un sumator paralel de $n$ biţi şi un bloc de comandă. Registrele sunt descrise în continuare.

A Acest registru este folosit ca şi acumulator pentru păstrarea produsului parţial. La sfârşitul operaţiei de înmulţire, $A_n$ va conţine semnul rezultatului. Registrul trebuie să aibă intrări paralele pentru înscrierea informaţiei de la sumator, să funcţioneze ca registru de deplasare spre dreapta şi în acest proces să permită transferul bitului $A_0$ în poziţia $n$-1 a registrului Q.

B Registrul B este folosit pentru încărcarea paralelă a deînmulţitului, iar conţinutul lui se transferă la una din intrările sumatorului.

Q Acest registru este folosit pentru încărcarea paralelă a înmulţitorului. Funcţionează ca un registru de deplasare la dreapta, cu excepţia bitului c.m.s. $Q_n$, şi permite înscrierea bitului c.m.p.s. al acumulatorului printr-o intrare serială.

Dacă dispozitivul trebuie să opereze şi asupra numerelor reprezentate în complemente, trebuie să permită şi complementarea deînmulţitului şi/sau a înmulţitorului, dacă aceştia sunt negativi, şi complementarea registrului combinat A_Q, dacă rezultatul este negativ.

Blocul de comandă al operaţiei de înmulţire generează semnalele de comandă în urma cărora se succed operaţiile elementare din care se compune operaţia de înmulţire.

Acest bloc conține un numărător N, care este inițializat cu numărul biților de mărime *n*, şi este decrementat în fiecare pas al operației.

Organigrama operației de înmulțire prin metoda directă este prezentată în Figura 6.10.

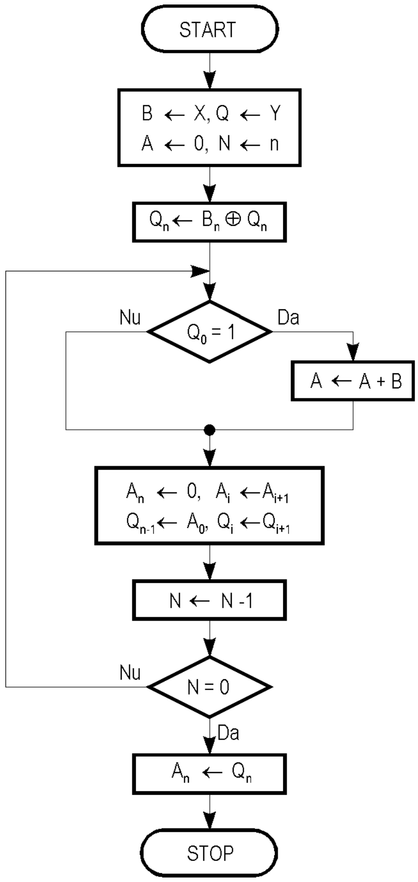


**Figura 6.10.** Organigrama operației de înmulțire prin metoda directă.

Operația începe cu inițializarea registrelor. Deînmulțitul şi înmulțitorul se încarcă în registrul B, respectiv Q, registrul acumulator este inițializat cu 0, iar numărătorul N este inițializat cu numărul cifrelor de mărime *n*. Se formează apoi semnul rezultatului, care este păstrat în bistabilul $Q_n$ al registrului Q.

Se testează bitul $Q_0$ al înmulțitorului, şi dacă acest bit este 1, se adună deînmulțitul la produsul parțial aflat în registrul acumulator. Se deplasează registrul combinat A_Q la dreapta cu o poziție; pe poziția c.m.s. a registrului A se introduce 0. Se decrementează apoi numărătorul N. Dacă numărătorul nu a ajuns la 0, se continuă cu pasul următor al operației, testând următorul bit al înmulțitorului. Dacă numărătorul este 0, se transferă semnul rezultatului din bistabilul $Q_n$ pe poziția c.m.s. a registrului acumulator.

### Exemplul 6.2

Considerăm $X$ = 9, $Y$ = -12. Execuția operației de înmulțire este ilustrată în Tabelul 6.4, care prezintă conținutul registrelor dispozitivului de înmulțire în fiecare pas al operației.

**Tabelul 6.4.** Execuția operației de înmulțire 9 × (-12) prin metoda directă.

| Pas | A | Q | B | $Q_n$ | $Q_0$ | N | Operații |
|---|---|---|---|---|---|---|---|
| 0 | 0 0000 | 110<u>0</u> | 1001 | 1 | 0 | 4 | Inițializare |
| 1 | 0 0000 | 011<u>0</u> | 1001 | 1 | 0 | 3 | Deplasare A_Q la dreapta |
| 2 | 0 0000 + | 001<u>1</u> | 1001 | 1 | 1 | 2 | Deplasare A_Q la dreapta |
| 3 | <u>1001</u><br>0 1001<br>0 0100 + | <br>0011<br>100<u>1</u> | 1001 | <br><br>1 | <br><br>1 | <br><br>1 | Adunare deînmulțit<br><br>Deplasare A_Q la dreapta |
| 4 | <u>1001</u><br>0 1101<br>0 0110 | <br>1001<br>110<u>0</u> | 1001 | <br><br>1 | <br><br>0 | <br><br>0 | Adunare deînmulțit<br><br>Deplasare A_Q la dreapta |
| 5 | 1 0110 | 1100 | 1001 | 1 | 0 | 0 | Stabilire semn rezultat |

Rezultatul este: $1\ 0110\ 1100_2 = -0110\ 1100_2$ = -6Ch = -(6·16+12) = -108.

#### 6.2.2.2. Metoda Booth

Dacă numerele din calculator sunt reprezentate în C2, se preferă o metodă de înmulțire care operează direct asupra numerelor în această reprezentare. Avantajul unei asemenea metode constă în eliminarea operației de complementare a operanzilor şi a rezultatului, dacă aceştia sunt negativi, ceea ce este important în cazul conversiei din C2 în MS, care este mai dificilă decât cea din C1 în MS. Asemenea metode sunt metoda Burks-Goldstine-von Neumann, metoda Robertson şi metoda Booth. Dintre acestea, metoda Booth (elaborată de A. D. Booth şi K. H. V. Booth), este cea mai utilizată.

Fie $X$ şi $Y$ deînmulțitul, respectiv înmulțitorul. Înmulțitorul se poate exprima sub forma:

$$Y = y_n 2^n + \sum_{i=0}^{n-1} y_i 2^i \tag{6.21}$$

unde $y_n$ este bitul de semn. Pentru început, se consideră că înmulțitorul este pozitiv, deci $y_n = 0$. Produsul $XY$ va fi:

$$XY = X\left(y_0 2^0 + y_1 2^1 + y_2 2^2 + \ldots + y_{n-1} 2^{n-1} + y_n 2^n\right) \tag{6.22}$$

Fiecare cifră $y_i$ se va înlocui cu diferența a două cifre alăturate ale înmulțitorului, $y_{i-1} - y_i$. Astfel, se vor efectua următoarele înlocuiri:

$y_0$ cu $y_{-1} - y_0$
$y_1$ cu $y_0 - y_1$
$y_2$ cu $y_1 - y_2$
…
$y_{n-1}$ cu $y_{n-2} - y_{n-1}$
$y_n$ cu $y_{n-1} - y_n$

Presupunând că $y_{-1} = 0$, produsul devine:

$$XY = X\left[-y_0 2^0 + (y_0 - y_1)2^1 + (y_1 - y_2)2^2 + \ldots + (y_{n-2} - y_{n-1})2^{n-1} + (y_{n-1} - y_n)2^n\right] \quad (6.23)$$

Se obține:

$$\begin{aligned} XY &= X\left[y_0\left(2^1 - 2^0\right) + y_1\left(2^2 - 2^1\right) + \ldots + y_{n-1}\left(2^n - 2^{n-1}\right) - y_n 2^n\right] \\ &= X\left(y_0 2^0 + y_1 2^1 + \ldots + y_{n-1} 2^{n-1} - y_n 2^n\right) \\ &= X\left(-y_n 2^n + \sum_{i=0}^{n-1} y_i 2^i\right) \end{aligned}$$

Deoarece $y_n = 0$, în paranteză se află valoarea înmulțitorului $Y$. Înlocuirea este deci corectă. Pentru cazul în care înmulțitorul este negativ, demonstrația se efectuează în mod similar.

Diferența $y_{i-1} - y_i$ a două cifre alăturate poate fi 1, –1, sau 0. Dacă diferența este 1, trebuie să se efectueze o adunare a deînmulțitului la produsul parțial. Dacă diferența este –1, trebuie să se efectueze o scădere a deînmulțitului din produsul parțial. Dacă diferența este 0, nu trebuie să se efectueze nici adunarea, nici scăderea deînmulțitului pentru această cifră $i$ a înmulțitorului.

Se observă că dacă diferența $y_{i-1} - y_i$ este 1, cele două cifre alăturate $y_i y_{i-1}$ ale înmulțitorului sunt 01, dacă diferența este –1, cele două cifre sunt 10, iar dacă diferența este 0, cele două cifre sunt 00 sau 11. Rezultă următorul algoritm al metodei Booth:

> În fiecare pas al operației se testează doi biți alăturați ai înmulțitorului, $y_i$ (bitul curent) şi $y_{i-1}$ (bitul testat în pasul precedent, numit şi bit de referință). Bitul de referință pentru bitul c.m.p.s. al înmulțitorului este $y_{-1} = 0$. Testarea se efectuează începând cu bitul c.m.p.s. al înmulțitorului. În funcție de valoarea biților testați $y_i y_{i-1}$, se efectuează operațiile indicate în Tabelul 6.5.

**Tabelul 6.5.** Operațiile efectuate la înmulțirea prin metoda Booth, în funcție de valoarea biților testați.

| $y_i$ $y_{i-1}$ | Operații |
|---|---|
| 0 0 | Deplasare produs parțial la dreapta |
| 0 1 | Adunare deînmulțit, deplasare produs parțial la dreapta |
| 1 0 | Scădere deînmulțit, deplasare produs parțial la dreapta |
| 1 1 | Deplasare produs parțial la dreapta |

Structura unui dispozitiv de înmulțire prin metoda Booth este prezentată în Figura 6.11.

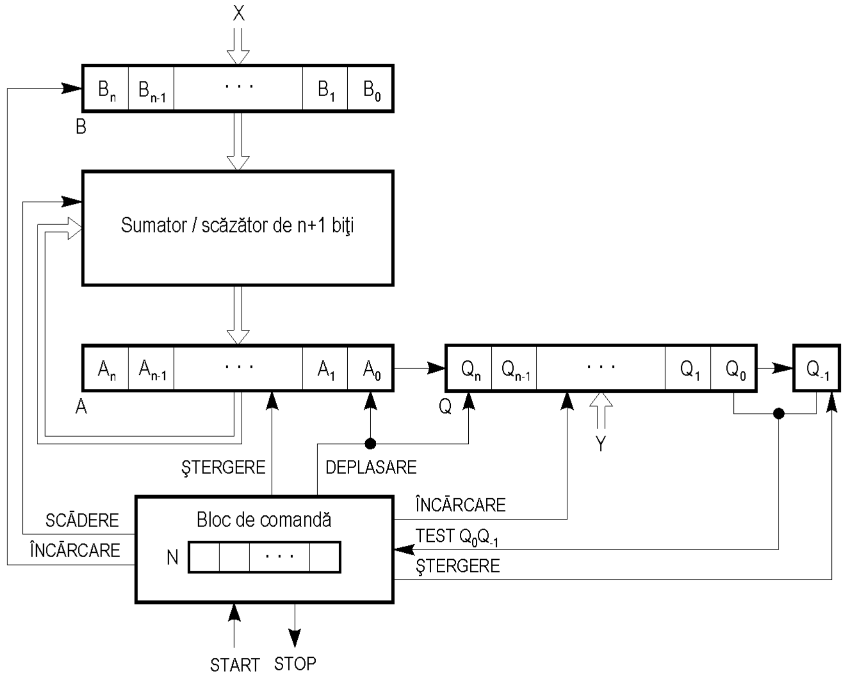


**Figura 6.11.** Schema bloc a unui dispozitiv de înmulțire prin metoda Booth.

Dispozitivul de înmulțire care utilizează această metodă conține un registru combinat A_Q, format din registrul acumulator A și registrul Q, de câte $n$+1 poziții, care păstrează produsul parțial, respectiv înmulțitorul. În acest caz, toate pozițiile înmulțitorului, inclusiv cea de semn, participă la deplasare. Registrul B, care păstrează deînmulțitul, are de asemenea $n$+1 poziții. Aceste registre trebuie să aibă posibilități similare cu cele utilizate la metoda de înmulțire directă. Mai este necesar un bistabil $Q_{-1}$, care păstrează bitul de referință. Sumatorul utilizat trebuie să aibă $n$+1 poziții, deoarece se operează cu toate cifrele, inclusiv cea de semn.

Blocul de comandă testează biții $Q_0$, $Q_{-1}$ în fiecare pas al operației. În funcție de valoarea acestor biți, se efectuează operațiile necesare, astfel:

$Q_0 \oplus Q_{-1} = 0$: Deplasare la dreapta a registrului A_Q

$\overline{Q}_0 Q_{-1} = 1$: Adunare deînmulțit, deplasare la dreapta a registrului A_Q

$Q_0 \overline{Q}_{-1} = 1$: Scădere deînmulțit, deplasare la dreapta a registrului A_Q

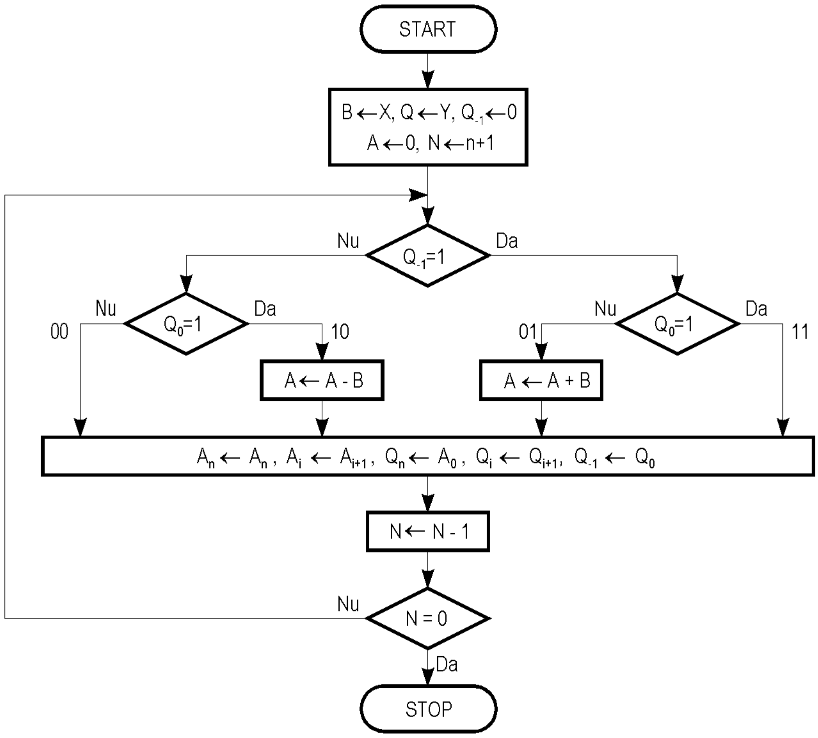


**Figura 6.12.** Organigrama operației de înmulțire prin metoda Booth.

## Observații

1. Inițial, bistabilul $Q_{-1}$ este şters, pentru ca bitul de referință pentru cifra c.m.p.s. a înmulțitorului să fie zero.
2. La deplasare se ține cont de regula de deplasare pentru numerele cu semn reprezentate în C2. În acest caz, o deplasare spre dreapta necesită introducerea în poziția de semn a bitului aflat aici înainte de deplasare. Bitul de semn din registrul A, deci $A_n$, este deplasat în $A_{n-1}$, dar rămâne şi în $A_n$ (se efectuează o deplasare aritmetică).
3. Se testează toți biții înmulțitorului, inclusiv bitul de semn.
4. Rezultatul se formează în registrul A_Q de $2n+2$ poziții, bitul de semn repetându-se pe primele două poziții (rezultatul propriu-zis are $2n+1$ biți).

Organigrama operației de înmulțire prin metoda Booth este prezentată în Figura 6.12.

În etapa de inițializare, se încarcă deînmulțitul și înmulțitorul în registrul B, respectiv Q, bistabilul $Q_{-1}$ și registrul acumulator sunt inițializate cu 0, iar numărătorul N este inițializat cu $n$+1.

În continuare se testează bitul de referință $Q_{-1}$ și bitul $Q_0$ al înmulțitorului. Dacă $Q_0\,Q_{-1}$ = 01, se adună deînmulțitul din registrul B la produsul parțial din registrul A. Dacă $Q_0\,Q_{-1}$ = 10, se scade deînmulțitul din produsul parțial. Pentru celelalte două combinații ale biților testați, în această etapă nu se efectuează nici o operație. În etapa următoare se deplasează la dreapta registrul combinat A_Q, astfel încât în poziția bitului de semn $A_n$ să rămână aceeași valoare (păstrarea bitului de semn este indicată în organigramă prin transferul $A_n \leftarrow A_n$). Se decrementează numărătorul N, şi dacă acesta nu a ajuns la zero, se continuă operația cu testarea biților $Q_0\,Q_{-1}$. Dacă numărătorul N a ajuns la zero, operația se termină.

### Exemplul 6.3

Considerăm $X$ = 13, $Y$ = -10. Exprimarea binară a acestor numere în C2 este următoarea:

$X$ = $0\ 1101_2$
$Y$ = $1\ 0110_2$

Scăderea deînmulțitului se efectuează prin adunarea complementului față de 2 al acestuia:

$X$ (C2) = $-X$ = $1\ 0011_2$

**Tabelul 6.6.** Execuția operației de înmulțire 13 × (–10) prin metoda Booth.

| Pas | A | Q | $Q_{-1}$ | B | $Q_0Q_{-1}$ | N | Operații |
|---|---|---|---|---|---|---|---|
| 0 | 0 0000 | 1 0110 | 0 | 0 1101 | 0 0 | 5 | Inițializare |
| 1 | 0 0000 + | 0 1011 | 0 | 0 1101 | 1 0 | 4 | Deplasare A_Q la dreapta |
| 2 | 1 0011<br>1 0011<br>1 1001 | <br>0 1011<br>1 0101 | <br>0<br>1 | 0 1101 | <br><br>1 1 | <br><br>3 | Scădere deînmulțit<br><br>Deplasare A_Q la dreapta |
| 3 | 1 1100 + | 1 1010 | 1 | 0 1101 | 0 1 | 2 | Deplasare A_Q la dreapta |
| 4 | 0 1101<br>0 1001<br>0 0100 + | <br>1 1010<br>1 1101 | <br>1<br>0 | 0 1101 | <br><br>1 0 | <br><br>1 | Adunare deînmulțit<br><br>Deplasare A_Q la dreapta |
| 5 | 1 0011<br>1 0111<br>1 1011 | <br>1 1101<br>1 1110 | <br>0<br>1 | 0 1101 | <br><br>0 1 | <br><br>0 | Scădere deînmulțit<br><br>Deplasare A_Q la dreapta |

Conținutul registrelor dispozitivului de înmulțire în fiecare pas al operației este prezentat în Tabelul 6.6.

Rezultatul este: $1\ 0111\ 1110_2 = -1000\ 0010_2$ = -82h = -(8·16+2) = -130.