CAPITOLUL 5

CIRCUITE ARITMETICE SECVENȚIALE

În acest capitol se descriu diferite metode de înmulțire și împărțire. Pentru fiecare metodă se descrie și implementarea acesteia printr-un circuit aritmetic secvențial. Metodele descrise sunt înmulțirea prin deplasare și adunare, înmulțirea Booth, împărțirea cu refacerea restului parțial și împărțirea fără refacerea restului parțial. La sfârșitul capitolului se descrie proiectarea unui circuit secvențial de înmulțire care utilizează metoda Booth și implementarea acesteia pe o placă de dezvoltare.

5.1 Înmulțirea binară

5.1.1 Metode de înmulțire binară

Există numeroase metode de înmulțire binară. Fiecare din aceste metode are mai multe variante, care țin cont de modul de reprezentare a numerelor cu semn sau urmăresc creșterea vitezei de execuție a operației. Cele mai importante metode sunt înmulțirea prin deplasare și adunare, înmulțirea prin metoda Burks-Goldstine-von Neumann, înmulțirea prin metoda Robertson, înmulțirea prin metoda Booth și înmulțirea într-o bază superioară. Unele metode se pot utiliza numai pentru numere fără semn, în timp ce altele se pot utiliza și pentru numere cu semn.

5.1.2 Înmulțirea prin deplasare și adunare

Metoda de înmulțire prin deplasare și adunare se poate aplica pentru numere reprezentate în mărime și semn. Cu această metodă, cifra de semn este tratată separat. Dacă numerele sunt reprezentate în complement față de unu (C1) sau complement față de doi (C2), aplicarea metodei necesită complementarea operandului sau a operanzilor negativi, pentru ca reprezentarea cifrelor de mărime să fie aceeași ca în mărime și semn,

iar apoi 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 și cu Y înmulțitorul, aceștia se pot exprima în mărime și semn sub forma următoare:

$$X = x_n 2^n + \sum_{i=0}^{n-1} x_i 2^i = x_n 2^n + x \quad (5.1)$$

$$Y = y_n 2^n + \sum_{i=0}^{n-1} y_i 2^i = y_n\, 2^n + y \quad (5.2)$$

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

Produsul se poate scrie sub forma:

$$Z = X * Y = z_n 2^{2n} + z \quad (5.3)$$

unde z_n reprezintă semnul produsului,

$$z_n = x_n \oplus y_n \quad (5.4)$$

iar z reprezintă mărimea produsului:

$$z = x * y = x\, y_{n-1} 2^{n-1} + x\, y_{n-2} 2^{n-2} + \ldots + x\, y_1 2^1 + x\, y_0 2^0 \quad (5.5)$$

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 următoare:

$$p_0 = x\, y_0 2^0 \quad (5.6)$$

$$p_1 = p_0 + x\, y_1 2^1 \quad (5.7)$$

$$p_2 = p_1 + x\, y_2 2^2 \quad (5.8)$$

$$\ldots$$

$$p_{n-2} = p_{n-3} + x\, y_{n-2} 2^{n-2} \quad (5.9)$$

$$p_{n-1} = p_{n-2} + x\, y_{n-1} 2^{n-1} \quad (5.10)$$

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

$$p_{n-1} = x * (y_{n-1} 2^{n-1} + y_{n-2} 2^{n-2} + \ldots + y_1 2^1 + y_0 2^0) = x * y \quad (5.11)$$

Produsul final se poate scrie sub forma:

$$Z = (x_n \oplus y_n)\, 2^{2n} + p_{n-1} \quad (5.12)$$

unde p_{n-1} se obține din următoarea relație de recurență:

$$p_{n-1} = p_{n-2} + x\, y_{n-1} 2^{n-1} \quad (5.13)$$

cu $p_0 = x\, y_0 2^0$.

Fie $X = 12$ și $Y = 6$. Considerând numere fără semn cu patru biți de mărime, înmulțirea binară a celor două numere se efectuează în modul indicat în continuare.

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

Procesul de înmulțire constă în testarea succesivă a biților înmulțitorului, începând cu bitul cel mai puțin semnificativ. Dacă bitul testat este 1, produsul parțial corespunzător este copia deînmulțitului, iar în caz contrar este zero.

Se pot formula următoarele observații:

- Înmulțirea implică generarea unor produse parțiale, câte unul pentru fiecare bit al înmulțitorului. Aceste produse sunt apoi însumate pentru a se obține produsul final.
- Dacă bitul testat al înmulțitorului este 0, produsul parțial corespunzător este zero. Dacă acest bit este 1, produsul parțial este egal cu deînmulțitul.
- Fiecare produs parțial este deplasat la stânga cu o poziție relativ la produsul parțial precedent.
- 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 obișnuită, se pot efectua unele modificări pentru ca operația să fie mai eficientă. De exemplu, 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. Deoarece nu este necesară memorarea tuturor produselor parțiale, astfel vor fi 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 și o deplasare. Pentru fiecare bit de 0 din înmulțitor, produsul parțial este zero, fiind necesară numai o deplasare.

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

1. Se inițializează produsul parțial cu zero.
2. Se testează biții înmulțitorului, începând cu bitul cel mai puțin semnificativ. Dacă bitul testat este 1, se adună deînmulțitul la produsul parțial.
3. 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. Astfel, dacă numerele au același semn, produsul este pozitiv, iar

în caz contrar produsul este negativ. Pentru numere de n+1 biți (cu 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 ai deînmulțitului și înmulțitorului. Deoarece deînmulțitul trebuie deplasat la stânga în fiecare pas al operației, și pentru deînmulțit este necesar un registru de lungime dublă.

Exemplul 5.1

Considerăm din nou numerele fără semn $X = 12$ și $Y = 6$, cu patru biți de mărime. Înmulțirea celor două numere, efectuată conform algoritmului descris anterior, este prezentată în tabelul 5.1.

Tabelul 5.1. Exemplu de înmulțire prin deplasare și adunare.

Pas	A	X	Y
0	0000 0000	0000 1100	0110
1	0000 0000 +	0000 1100	0110
2	0001 1000 0001 1000 +	0001 1000	0110
3	0011 0000 0100 1000	0011 0000	0110
4	0100 1000	0110 0000	0110

Rezultatul din registrul A este $0100\ 1000_2 = 48_{16} = 4*16 + 8 = 64 + 8 = 72$.

Același rezultat se obține prin deplasarea produsului parțial la dreapta în locul deplasării deînmulțitului la stânga, astfel încât produsul parțial și deînmulțitul să fie păstrate în aceeași poziție relativă. O altă variantă este ca în locul testării înmulțitorului începând cu bitul cel mai puțin semnificativ să se testeze înmulțitorul începând cu bitul său cel mai semnificativ. În acest caz, se poate deplasa fie deînmulțitul la dreapta, fie produsul parțial la stânga.

Dintre aceste variante, cea mai avantajoasă este cea în care testarea înmulțitorului se realizează începând cu bitul cel mai puțin semnificativ și produsul parțial este deplasat la dreapta. Avantajele acestei variante sunt următoarele:

- Este necesar un sumator de n biți, chiar dacă rezultatul va avea $2n$ biți (considerând numere fără semn).
- Deînmulțitul necesită un registru de n biți în loc de $2n$ biți.
- Registrul acumulator necesită n+1 biți (cu un bit suplimentar pentru a păstra transportul de la operația de adunare) în loc de $2n$+1 biți.

Deoarece rezultatul are $2n$ biți, registrul acumulator ar trebui completat cu un registru auxiliar pentru memorarea produsului complet. Însă, deoarece după înmulțirea cu un bit al înmulțitorului acest bit nu mai este necesar, registrul acumulator se poate plasa

la stânga registrului înmulțitorului, formând cu acesta un registru combinat care se deplasează la dreapta în fiecare pas al operației. Astfel, în final registrul acumulator păstrează biții cei mai semnificativi ai rezultatului, iar registrul înmulțitorului păstrează biții cei mai puțin semnificativi ai rezultatului.

Schema bloc a unui circuit care implementează metoda de înmulțire prin deplasare și adunare pentru numere fără semn de câte *n* biți este prezentată în figura 5.1.

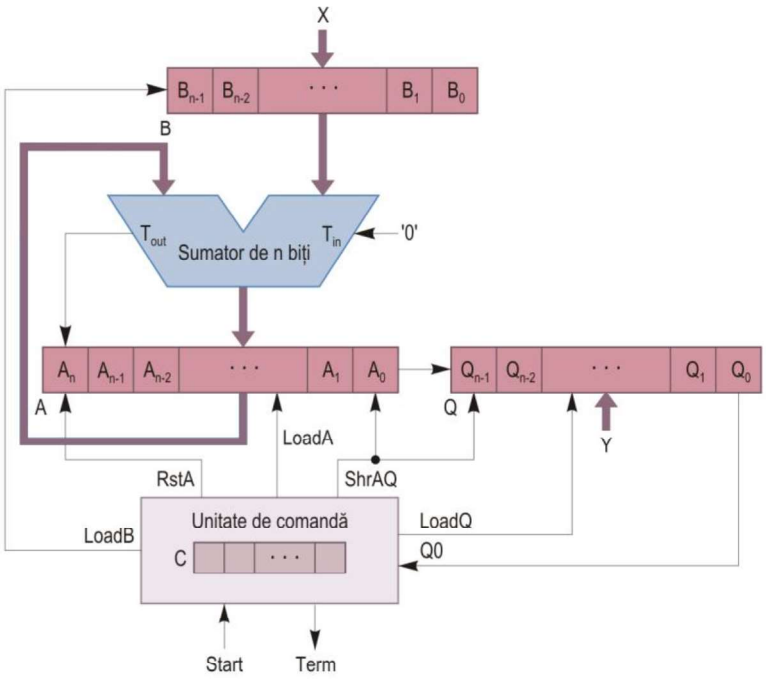

Figura 5.1. Schema bloc a unui circuit de înmulțire prin deplasare și adunare pentru numere fără semn de *n* biți.

Circuitul de înmulțire conține registrele B și Q de câte *n* biți, registrul A de *n*+1 biți, un sumator paralel de *n* biți și o unitate de comandă. Registrele circuitului sunt descrise în continuare.

A Registrul A este utilizat ca și acumulator pentru păstrarea produsului parțial. Poziția A_n a acestui registru păstrează bitul de transport de la sumator. La sfârșitul operației de înmulțire, registrul A va conține partea mai semnificativă a produsului. Acest registru trebuie să aibă intrări paralele pentru înscrierea informației de la sumator, să funcționeze ca registru de deplasare la dreapta, iar în acest proces să permită transferul bitului A_0 în poziția *n*-1 a registrului Q.

B Registrul B este utilizat pentru păstrarea deînmulțitului. Conținutul acestui registru se transferă la una din intrările sumatorului.

Q Registrul Q este utilizat pentru păstrarea înmulțitorului. Acest registru funcționează ca un registru de deplasare la dreapta și permite introducerea bitului cel mai puțin semnificativ al acumulatorului în poziția Q_{n-1} printr-o intrare serială. La sfârșitul operației, registrul Q va conține partea mai puțin semnificativă a produsului.

Unitatea de comandă generează semnalele de comandă în urma cărora se succed operațiile elementare din care se compune operația de înmulțire. Această unitate conține un numărător C, care este inițializat cu numărul *n* al biților operanzilor și este decrementat în fiecare pas al operației.

Organigrama operației de înmulțire prin deplasare și adunare pentru numere fără semn este prezentată în figura 5.2.

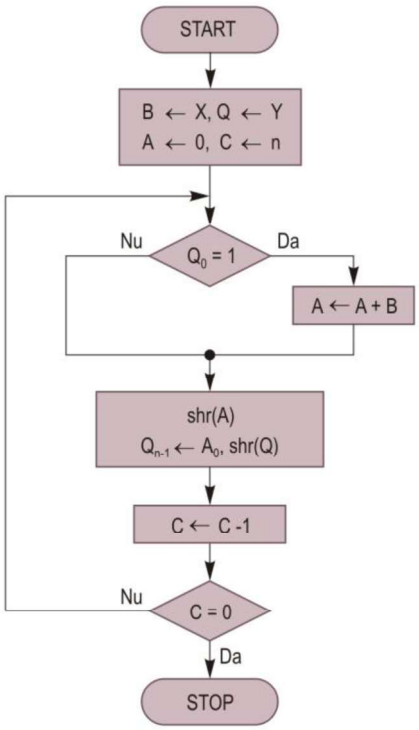

Figura 5.2. Organigrama operației de înmulțire prin deplasare și adunare pentru numere fără semn.

Operația începe cu inițializarea registrelor. Deînmulțitul *X* se încarcă în registrul B, înmulțitorul *Y* se încarcă în registrul Q, registrul acumulator A este inițializat cu 0, iar numărătorul C este inițializat cu numărul biților operanzilor (*n*). În fiecare pas al operației, se testează bitul Q_0 al înmulțitorului. Dacă acest bit este 1, se adună deînmulțitul la produsul parțial aflat în registrul acumulator A. Apoi se deplasează registrele A și Q la dreapta cu o poziție; în figura 5.2, aceasta se indică prin shr(A), respectiv shr(Q). Bitul

cel mai semnificativ al registrului A se setează la 0, iar în poziția cea mai semnificativă a registrului Q se transferă bitul 0 al registrului A. Se decrementează apoi numărătorul C. Dacă numărătorul nu a ajuns la zero, operația continuă cu pasul următor, în care se testează următorul bit al înmulțitorului. Dacă numărătorul a ajuns la zero, operația se termină.

Exemplul 5.2

Considerăm $X = 9$ și $Y = 12$. Execuția operației de înmulțire este ilustrată în tabelul 5.2, care prezintă conținutul registrelor circuitului în fiecare pas al operației.

Tabelul 5.2. Execuția operației de înmulțire 9 × 12 cu metoda prin deplasare și adunare.

Pas	A	Q	B	Q_0	C	Operații
0	0 0000	1100	1001	0	4	Inițializare
1	0 0000	0110	1001	0	3	Deplasare A_Q la dreapta
2	0 0000 +	0011	1001	1	2	Deplasare A_Q la dreapta
3	1001 0 1001 0 0100 +	 0011 1001	1001	 1	 1	Adunare deînmulțit Deplasare A_Q la dreapta
4	1001 0 1101 0 0110	 1001 1100	1001	 0	 0	Adunare deînmulțit Deplasare A_Q la dreapta

Rezultatul din registrele A și Q este $0110\ 1100_2 = 6C_{16} = 6*16 + 12 = 108$.

5.1.3. Înmulțirea prin 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 și a rezultatului, dacă aceștia sunt negativi, ceea ce este important în cazul conversiei din C2 în mărime și semn, care este mai dificilă decât conversia din C1. Asemenea metode sunt metoda Burks-Goldstine-von Neumann, metoda Robertson și metoda Booth. Dintre acestea, metoda Booth, care a fost elaborată de Andrew Donald Booth în anul 1950, este cea mai utilizată.

Fie X deînmulțitul și Y înmulțitorul. Înmulțitorul Y se poate exprima sub forma următoare:

$$Y = y_n 2^n + \sum_{i=0}^{n-1} y_i 2^i \tag{5.14}$$

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

$$X * Y = X * (y_0 2^0 + y_1 2^1 + y_2 2^2 + \ldots + y_{n-1} 2^{n-1} + y_n 2^n) \tag{5.15}$$

Se propune înlocuirea fiecărei cifre y_i cu diferența a două cifre alăturate ale înmulțitorului, $y_{i-1} - y_i$. Astfel, se vor efectua următoarele înlocuiri:

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

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

$$X * Y = X * \begin{bmatrix} -y_0 2^0 + (y_0 - y_1)\, 2^1 + (y_1 - y_2)\, 2^2 + \cdots \\ +(y_{n-2} - y_{n-1})\, 2^{n-1} + (y_{n-1} - y_n)\, 2^n \end{bmatrix} \tag{5.16}$$

Se obține:

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

Deoarece $y_n = 0$, în paranteză se află valoarea înmulțitorului Y. Deci, înlocuirea este corectă. În cazul unui înmulțitor negativ, demonstrația se efectuează în mod similar.

Diferența $y_{i-1} - y_i$ dintre 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 de înmulțire prin metoda Booth se testează doi biți alăturați ai înmulțitorului, y_i (bitul curent) și y_{i-1} (bitul testat în pasul precedent, numit și bit de referință). Bitul de referință pentru bitul cel mai puțin semnificativ al înmulțitorului este $y_{-1} = 0$. Testarea se efectuează începând cu bitul cel mai puțin semnificativ al înmulțitorului. În funcție de valoarea biților testați $y_i y_{i-1}$, se efectuează operațiile indicate în tabelul 5.3.

Tabelul 5.3. Operații 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

Schema bloc a unui circuit de înmulțire prin metoda Booth pentru numere cu semn de câte $n + 1$ biți este prezentată în figura 5.3.

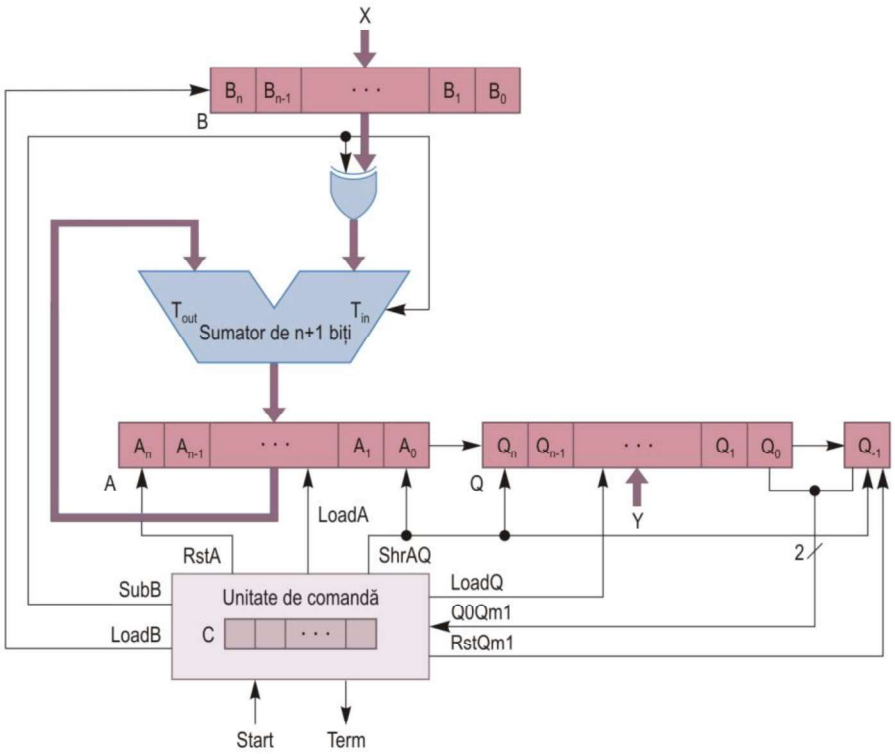

Figura 5.3. Schema bloc a unui circuit de înmulțire prin metoda Booth.

Circuitul 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 n+1 poziții. Aceste registre trebuie să aibă caracteristici similare cu cele utilizate la metoda de înmulțire prin deplasare și adunare. 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.

Unitatea de comandă testează biții Q_0 și Q_{-1} în fiecare pas al operației. În funcție de valoarea acestor biți, se efectuează operațiile următoare:

$\overline{Q}_0 Q_{-1} = 1$: Adunarea deînmulțitului, deplasarea la dreapta a registrului A_Q.
$Q_0 \overline{Q}_{-1} = 1$: Scăderea deînmulțitului, deplasarea la dreapta a registrului A_Q.
$Q_0 \oplus Q_{-1} = 0$: Deplasarea la dreapta a registrului A_Q.

Scăderea deînmulțitului din produsul parțial se efectuează prin adunarea complementului față de doi al deînmulțitului la produs. Pentru generarea complementului față de doi, se utilizează n+1 porți SAU EXCLUSIV cu câte două intrări și semnalul *SubB*.

Dacă acest semnal este '1', la ieșirile porților SAU EXCLUSIV se obține complementul față de unu al deînmulțitului, care se adună cu valoarea '1' aplicată la intrarea de transport T_{in} a sumatorului și cu produsul parțial. Deci, complementul față de doi se obține prin adunarea valorii 1 la complementul față de unu.

Observații

1. Inițial, bistabilul Q_{-1} este resetat, pentru ca bitul de referință pentru cifra cea mai puțin semnificativă 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 la dreapta necesită introducerea în poziția bitului de semn a semnului aflat în acea poziție înainte de deplasare. Bitul de semn din registrul A (A_n) este deplasat în poziția A_{n-1}, dar rămâne și în A_n (aceasta este o deplasare aritmetică la dreapta).
3. Se testează toți biții înmulțitorului, inclusiv bitul de semn.
4. Rezultatul se formează în registrul combinat A_Q de $2n$+2 poziții, bitul de semn fiind repetat în primele două poziții (rezultatul propriu-zis are $2n$+1 biți).

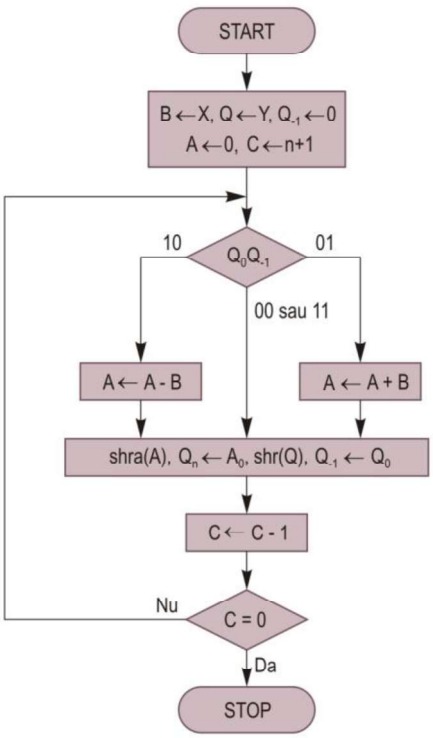

Figura 5.4. Organigrama operației de înmulțire prin metoda Booth.

Organigrama operației de înmulțire prin metoda Booth este ilustrată în figura 5.4. În etapa de inițializare, se încarcă deînmulțitul X în registrul B, se încarcă înmulțitorul Y în registrul Q, bistabilul Q_{-1} și registrul acumulator A sunt inițializate cu 0, iar numărătorul C 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 registrele A și Q, astfel încât în poziția bitului de semn A_n a registrului A să rămână aceeași valoare (în figura 5.4, aceasta se indică prin shra(A), ceea ce înseamnă o deplasare aritmetică la dreapta). În poziția cea mai semnificativă a registrului Q se transferă bitul 0 al registrului A. Se decrementează numărătorul C, iar dacă acesta nu a ajuns la zero, se continuă operația cu testarea biților $Q_0\,Q_{-1}$. Dacă numărătorul C a ajuns la zero, operația se termină.

Exemplul 5.3

Considerăm numerele $X = 13$ și $Y = -10$. Exprimarea acestor numere în C2 este următoarea:

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

Scăderea deînmulțitului se efectuează prin adunarea valorii negative a acestuia, exprimată în C2:

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

Conținutul registrelor circuitului în fiecare pas al operației este prezentat în tabelul 5.4.

Tabelul 5.4. Execuția operației de înmulțire 13 × (-10) prin metoda Booth.

Pas	A	Q	Q_{-1}	B	Q_0Q_{-1}	C	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			0 1101			Scădere deînmulțit
	1 0011	0 1011	0				
	1 1001	1 0101	1		1 1	3	Deplasare A_Q la dreapta
3	1 1100 +	1 1010	1	0 1101	0 1	2	Deplasare A_Q la dreapta
4	0 1101			0 1101			Adunare deînmulțit
	0 1001	1 1010	1				
	0 0100 +	1 1101	0		1 0	1	Deplasare A_Q la dreapta
5	1 0011			0 1101			Scădere deînmulțit
	1 0111	1 1101	0				
	1 1011	1 1110	1		0 1	0	Deplasare A_Q la dreapta

Rezultatul obținut în registrele A și Q, fără repetarea bitului de semn, este 1 0111 $1110_2 = -1000\ 0010_2 = -82_{16} = -(8*16 + 2) = -130$.

5.2 Împărțirea binară

5.2.1 Principiul împărțirii binare

Fiind dat deîmpărțitul X și împărțitorul Y de câte n cifre de mărime, operația de împărțire constă în determinarea câtului Q și a restului R, astfel încât să fie satisfăcută relația:

$$X = Q \cdot Y + R \tag{5.17}$$

Împărțirea binară se efectuează în mod asemănător cu împărțirea zecimală. Algoritmul de împărțire zecimală încearcă să determine cifrele câtului prin alegerea unei cifre și scăderea produsului dintre această cifră și împărțitor din restul parțial (care este o parte a deîmpărțitului). Dacă rezultatul scăderii este un număr pozitiv mai mic decât împărțitorul, cifra aleasă este corectă. În caz contrar, se alege o altă cifră și operația se repetă. În fiecare etapă a operației, se obține o cifră a câtului.

În cazul împărțirii binare, dacă se alege în mod eronat o cifră a câtului, o nouă alegere nu mai este necesară, existând doar două cifre, 0 și 1. Operația de împărțire se reduce la o serie de scăderi ale împărțitorului din restul parțial, care se efectuează numai dacă restul parțial este mai mare decât împărțitorul, caz în care cifra câtului este 1; în caz contrar, cifra corespunzătoare a câtului este 0.

Exemplul 5.4

Considerăm împărțirea numerelor fără semn 147 și 11. Exprimarea acestor numere în binar este următoarea: $147_{10} = 1001\ 0011_2$, $11_{10} = 1011_2$. Împărțirea celor două numere se efectuează în modul indicat în continuare.

```
10010011 : 1011 = 00001101      Cât (13)
 1011
  1110-          Rest parțial
  1011
    1111-        Rest parțial
    1011
     100         Rest (4)
```

Pentru împărțire, se examinează biții deîmpărțitului de la stânga la dreapta, până când setul biților examinați reprezintă un număr mai mare sau egal cu împărțitorul. Primul bit examinat este 1, apoi se examinează setul de biți 10, după care se continuă cu seturile de biți 100, 1001 și 10010. Cât timp setul biților examinați este mai mic decât împărțitorul, se obțin cifre de 0 pentru cât. Atunci când acest set este mai mare sau egal cu împărțitorul, se obține o cifră de 1 pentru cât, împărțitorul fiind scăzut din partea corespunzătoare a deîmpărțitului, obținându-se un rest parțial. În continuare, în fiecare pas al operației se adaugă câte un bit din deîmpărțit la restul parțial, până când se obține un număr mai mare sau egal cu împărțitorul. Se scade apoi împărțitorul

din restul parțial, obținându-se un nou rest parțial. Operația continuă până când se examinează toți biții deîmpărțitului. Se obține câtul 13 și restul 4.

La operația de împărțire, dacă împărțitorul este zero, operația trebuie oprită, ceea ce necesită testarea împărțitorului înaintea operației. Se poate testa și deîmpărțitul înaintea operației, iar dacă acesta este zero, operația este oprită și câtul va fi zero.

Pentru împărțirea a două numere de câte *n* biți, este necesar un registru acumulator de lungime dublă față de cea a numerelor, de 2*n* biți. În acest registru se încarcă deîmpărțitul în pozițiile mai puțin semnificative ale sale. În locul unui registru acumulator de lungime dublă, se poate utiliza un registru combinat, format din registrul acumulator de *n* biți și registrul care păstrează deîmpărțitul de *n* biți. Pentru a se obține un cât cu un număr de biți egal cu cel al deîmpărțitului și împărțitorului (*n*), trebuie să se efectueze un număr de comparații egal cu *n*. Pentru scăderea împărțitorului din pozițiile corespunzătoare ale restului parțial, în fiecare etapă a algoritmului se deplasează registrul combinat care păstrează acest rest la stânga cu o poziție.

Este posibil ca deîmpărțitul să aibă o lungime dublă față de cea a împărțitorului. În această situație, partea mai semnificativă a deîmpărțitului se încarcă în registrul acumulator, iar partea mai puțin semnificativă a acestuia se încarcă în registrul destinat deîmpărțitului. În acest caz, poate apare o depășire, deci câtul obținut poate fi mai mare decât numărul maxim reprezentabil într-un registru de *n* biți. Pentru a nu apare o depășire, trebuie ca partea deîmpărțitului care se află în registrul acumulator să fie mai mică decât împărțitorul.

Exemplul 5.5

Considerăm operația de împărțire a numerelor fără semn 29 și 3. Exprimarea acestor numere în binar este următoarea: $29_{10} = 0001\ 1101_2$, $3_{10} = 0011_2$. Deîmpărțitul se încarcă în registrele A și X, iar împărțitorul se încarcă în registrul Y. Execuția operației de împărțire este prezentată în tabelul 5.5.

Tabelul 5.5. Execuția operației de împărțire 29 : 3.

Pas	A	X	Y	Cifra câtului
0	0001	1101	0011	
1	0011 – 0011 0000	1010 1010	0011	 1
2	0001	0100	0011	0
3	0010	1000	0011	0
4	0101 – 0011 0010	0000 0000	0011	 1

Câtul obținut este $1001_2 = 9$, iar restul obținut în registrul A este $0010_2 = 2$.

5.2.2 Metode de împărțire binară

Există trei metode principale de împărțire. Acestea sunt metoda comparației, metoda refacerii restului parțial și metoda fără refacerea restului parțial.

În cazul *metodei comparației*, circuitul de împărțire conține un circuit comparator utilizat pentru compararea restului parțial cu împărțitorul. Deoarece circuitele de comparare sunt destul de complexe, această metodă nu se utilizează în mod obișnuit. În cazul *metodei refacerii restului parțial*, în fiecare etapă a operației se efectuează o scădere a împărțitorului din restul parțial, urmând ca atunci când rezultatul scăderii este un număr negativ, deci restul parțial este mai mic decât împărțitorul, să se refacă restul parțial la valoarea anterioară prin adunarea împărțitorului la restul parțial. În cazul *metodei fără refacerea restului parțial*, dacă în urma scăderii se obține un număr negativ, restul parțial nu se reface, dar în etapa următoare a operației se efectuează o adunare a împărțitorului la restul parțial în locul unei scăderi.

Metoda care poate fi utilizată depinde de forma de reprezentare a numerelor. Metodele comparației și refacerii restului parțial se pot aplica numai pentru numere reprezentate în mărime și semn. Dacă numerele sunt reprezentate într-o altă formă, este necesară conversia reprezentării în mărime și semn înaintea aplicării metodei și conversia rezultatului după operație. Metoda fără refacerea restului parțial se poate aplica oricărei forme de reprezentare, dar sunt necesare unele corecții ale rezultatului.

5.2.3 Împărțirea cu refacerea restului parțial

Metoda de împărțire cu refacerea restului parțial este asemănătoare cu metoda comparației, dar comparația este înlocuită cu o scădere. Fiecare etapă a operației de împărțire începe cu o deplasare a restului parțial la stânga cu o poziție. Se efectuează apoi scăderea împărțitorului din restul parțial, obținându-se noul rest parțial. Dacă se obține un număr pozitiv, cifra corespunzătoare a câtului este 1. Dacă se obține un număr negativ, cifra corespunzătoare a câtului este 0, împărțitorul fiind adunat la restul parțial pentru a-l reface. Pentru obținerea unui cât cu n cifre de mărime, aceste operații se repetă de n ori. Procesul de împărțire se poate opri dacă restul parțial devine zero.

Presupunem că se utilizează un registru combinat format din registrul acumulator și registrul deîmpărțitului, iar inițial registrul acumulator este 0 (deci, deîmpărțitul nu necesită un registru de lungime dublă). Notăm resturile parțiale cu $r_0, r_1, r_2, \ldots, r_n$. Restul parțial inițial este 0:

$$r_0 = 0 \tag{5.18}$$

Se deplasează registrul combinat la stânga cu o poziție. Primul rest parțial va fi restul parțial inițial deplasat la stânga ($2r_0$), la care se adaugă cifra cea mai semnificativă a deîmpărțitului, din care se scade sau nu împărțitorul:

$$r_1 = 2r_0 + x_{n-1} - q_{n-1}Y \tag{5.19}$$

unde $r_0 = 0$, iar q_{n-1} este 1 sau 0. Dacă $x_{n-1} \geq Y$, q_{n-1} este 1, iar ecuația (5.19) devine:

$$r_1 = x_{n-1} - Y$$

Dacă $x_{n-1} < Y$, q_{n-1} este 0, iar ecuația (5.19) devine:

$$r_1 = x_{n-1}$$

Următoarele resturi parțiale vor fi:

$$\begin{aligned} r_2 &= 2r_1 + x_{n-2} - q_{n-2}Y \\ &= 2^1 x_{n-1} + x_{n-2} - 2q_{n-1}Y - q_{n-2}Y \end{aligned} \tag{5.20}$$

$$\begin{aligned} r_3 &= 2r_2 + x_{n-3} - q_{n-3}Y \\ &= 2^2 x_{n-1} + 2^1 x_{n-2} + x_{n-3} - \\ &\quad -2^2 q_{n-1}Y - 2^1 q_{n-2}Y - q_{n-3}Y \end{aligned} \tag{5.21}$$

…

$$\begin{aligned} r_n &= 2r_{n-1} + x_0 - q_0 Y \\ &= 2^{n-1} x_{n-1} + 2^{n-2} x_{n-2} + \cdots + 2^0 x_0 - \\ &\quad -(2^{n-1} q_{n-1} + 2^{n-2} q_{n-2} + \ldots + 2^0 q_0)\, Y \end{aligned} \tag{5.22}$$

unde $\sum_{i=0}^{n-1} 2^i x_i = X$. Ecuația (5.22) se poate scrie sub forma:

$$X = \left(\sum_{i=0}^{n-1} 2^i q_i\right) Y + r_n \tag{5.23}$$

Comparând ecuația (5.23) cu ecuația (5.17), se poate deduce expresia câtului și a restului:

$$Q = \sum_{i=0}^{n-1} 2^i q_i \tag{5.24}$$

$$R = r_n \tag{5.25}$$

Deci, procedeul de împărțire este corect, obținându-se cifrele câtului, ultimul rest parțial fiind restul împărțirii.

Schema bloc a circuitului de împărțire care implementează metoda refacerii restului parțial pentru numere fără semn este prezentată în figura 5.5. Registrele B și Q au câte *n* poziții. Registrul A are *n*+1 poziții pentru a putea păstra semnul scăderii care se efectuează în fiecare pas al operației de împărțire. Registrele A și Q pot funcționa ca registre de deplasare spre stânga. Registrul A permite introducerea bitului cel mai semnificativ al registrului Q în poziția A_0 printr-o intrare serială. Sumatorul este de *n* biți. Conținutul registrului B poate fi adunat la sau scăzut din conținutul registrului acumulator A. Scăderea registrului B conținând împărțitorul din registrul A, care conține restul parțial, se efectuează prin adunarea complementului față de doi al registrului B la registrul A. Complementul față de doi al registrului B se obține cu un grup de porți SAU EXCLUSIV și semnalul *SubB*, într-un mod similar cu aceeași operație executată pentru

înmulțirea prin metoda Booth. Dacă semnalul *SubB* este '1', se efectuează o operație de scădere, iar în caz contrar se efectuează o operație de adunare.

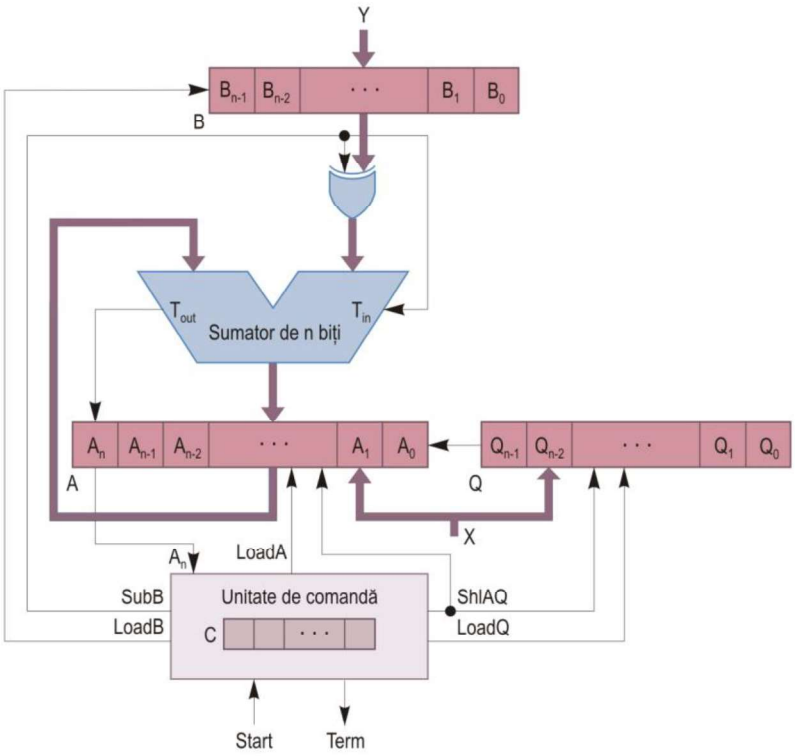

Figura 5.5. Schema bloc a unui circuit de împărțire prin metoda refacerii restului parțial pentru numere fără semn.

Organigrama operației de împărțire prin metoda refacerii restului parțial pentru numere fără semn este prezentată în figura 5.6. Deîmpărțitul *X* se încarcă în registrele A și Q, partea mai semnificativă fiind încărcată în registrul A. Împărțitorul *Y* se încarcă în registrul B. Numărătorul C se inițializează cu numărul cifrelor de mărime ale împărțitorului (*n*). În fiecare etapă a operației, se deplasează registrele A și Q la stânga cu o poziție, ceea ce se indică prin shl(A) și shl(Q), iar apoi se efectuează o scădere a împărțitorului (registrul B) din restul parțial (registrul A). Dacă rezultă un număr pozitiv în acumulator ($A_n = 0$), cifra câtului este 1, care se introduce în poziția Q_0 a registrului Q. Dacă rezultă un număr negativ în acumulator ($A_n = 1$), cifra câtului care se introduce în poziția Q_0 este 0. Deci, valoarea cifrei câtului care se obține în această etapă este:

$$Q_0 = \overline{A}_n$$

Dacă în urma scăderii rezultă un număr negativ, se reface restul parțial prin adunarea registrului B la acumulator. Se decrementează numărătorul C, iar dacă acesta nu este zero, operația continuă cu o nouă etapă. Dacă C este zero, operația este terminată, restul aflându-se în registrul A, iar câtul în registrul Q.

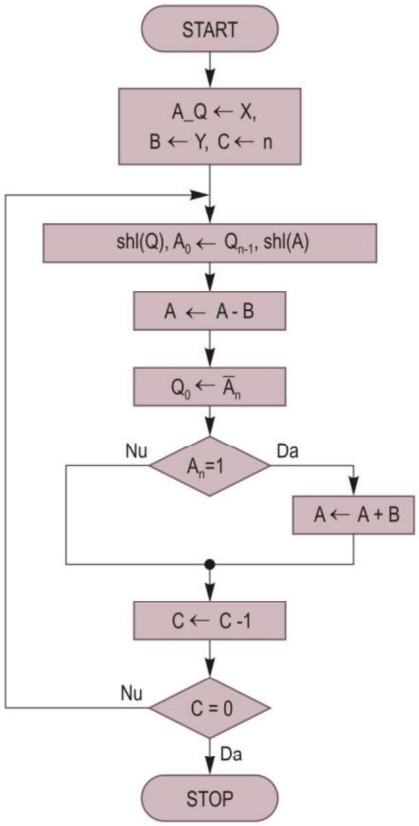

Figura 5.6. Organigrama operației de împărțire prin metoda refacerii restului parțial pentru numere fără semn.

În organigrama din figura 5.6 nu sunt indicate testele care trebuie efectuate după etapa de inițializare. Astfel, trebuie să se testeze dacă împărțitorul este zero; în caz afirmativ, operația se oprește și se indică o eroare de împărțire la zero. De asemenea, trebuie să se testeze dacă apare o depășire de capacitate. Depășirea de capacitate apare în cazul în care conținutul registrului A (partea mai semnificativă a deîmpărțitului) este mai mare sau egal cu conținutul registrului B (împărțitorul). Suplimentar, se poate testa dacă deîmpărțitul este mai mic decât împărțitorul; în caz afirmativ, câtul obținut va fi zero, iar restul va fi egal cu deîmpărțitul.

Exemplul 5.6

Considerăm numerele $X = 13$, $Y = 5$. Reprezentarea binară a acestor numere este următoarea:

$X = 1101_2$
$Y = 0101_2$

Valoarea negativă a împărțitorului, exprimată în C2, este:

$-Y\,(C2) = 1\ 1011_2$

Execuția operației este prezentată în tabelul 5.6.

Tabelul 5.6. Execuția operației de împărțire 13 : 5 prin metoda refacerii restului parțial.

Pas	A	Q	B	Q_0	C	Operații
0	0 0000	1101	0101	0	4	Inițializare
1	0 0001 + 1 1011 1 1100 + 0 0101 0 0001	1010 1010	0101	0	3	Deplasare A_Q la stânga Scădere împărțitor Adunare împărțitor
2	0 0011 + 1 1011 1 1110 + 0 0101 0 0011	0100 0100	0101	0	2	Deplasare A_Q la stânga Scădere împărțitor Adunare împărțitor
3	0 0110 + 1 1011 0 0001	1000 1001	0101	1	1	Deplasare A_Q la stânga Scădere împărțitor
4	0 0011 + 1 1011 1 1110 + 0 0101 0 0011	0010 0010	0101	0	0	Deplasare A_Q la stânga Scădere împărțitor Adunare împărțitor

Câtul obținut în registrul Q este $0010_2 = 2$, iar restul obținut în registrul A este $0\ 0011_2 = 3$.

5.2.4 Împărțirea fără refacerea restului parțial

Refacerea restului parțial determină creșterea timpului de execuție a operației de împărțire; în medie, această refacere se efectuează în 50% din cazuri. Prin eliminarea refacerii restului parțial, se pot îmbunătăți performanțele operației de împărțire.

Se poate observa că fiecare operație de refacere a restului parțial, deci de adunare a împărțitorului la restul parțial, este urmată de o scădere a împărțitorului în etapa următoare, după deplasarea restului parțial la stânga cu o poziție. Deplasarea la stânga cu o poziție este echivalentă cu înmulțirea cu 2. Succesiunea operațiilor efectuate este următoarea:

1. Se scade împărțitorul (Y) din restul parțial (R):

   $R \leftarrow R - Y$

2. Dacă rezultatul este negativ, se reface restul parțial, în aceeași etapă:

   $R \leftarrow R - Y + Y$

3. În etapa următoare, se deplasează restul parțial la stânga cu o poziție:

   $R \leftarrow 2 \times R$

4. Se scade împărțitorul din restul parțial, în aceeași etapă:

   $R \leftarrow 2 \times R - Y$

Același rezultat se poate obține printr-o altă succesiune de operații, în locul refacerii restului parțial și a scăderii împărțitorului în etapa următoare, astfel:

1. Se scade împărțitorul (Y) din restul parțial (R):

   $R \leftarrow R - Y$

2. Dacă rezultatul este negativ, nu se reface restul parțial, iar în etapa următoare se continuă cu deplasarea acestuia la stânga cu o poziție:

   $R \leftarrow 2 \times R - 2 \times Y$

3. Se adună împărțitorul la restul parțial, în aceeași etapă:

   $R \leftarrow 2 \times R - 2 \times Y + Y$

În concluzie, în fiecare pas al operației de împărțire fără refacerea restului parțial, după deplasarea registrelor A și Q la stânga cu o poziție, împărțitorul este scăzut din restul parțial sau este adunat la restul parțial în funcție de semnul restului parțial din pasul precedent. Dacă în pasul precedent restul parțial a fost pozitiv, se efectuează o scădere, iar dacă în pasul precedent restul parțial a fost negativ, se efectuează o adunare. În ultimul pas, dacă restul parțial este pozitiv, restul obținut este corect; în caz contrar, restul trebuie corectat prin refacerea acestuia (adunarea împărțitorului la rest).

Pe lângă creșterea vitezei operației de împărțire, metoda fără refacerea restului parțial are și avantajul că se poate utiliza pentru orice formă de reprezentare a numerelor. Pentru alte reprezentări decât cea în mărime și semn, sunt necesare anumite corecții.

Presupunând că numerele sunt reprezentate în mărime și semn, structura circuitului de împărțire care implementează această metodă este similară cu cea a circuitului care implementează metoda refacerii restului parțial (figura 5.5); diferă doar unitatea de comandă a operației. Această unitate conține un bistabil suplimentar BS, care indică operația care trebuie efectuată în pasul curent, astfel:

BS = 0: în pasul curent se efectuează scăderea împărțitorului din restul parțial;
BS = 1: în pasul curent se efectuează adunarea împărțitorului la restul parțial.

În primul pas se efectuează o scădere, deci bistabilul BS este inițializat cu 0. Organigrama operației de împărțire fără refacerea restului parțial pentru numere fără semn este prezentată în figura 5.7.

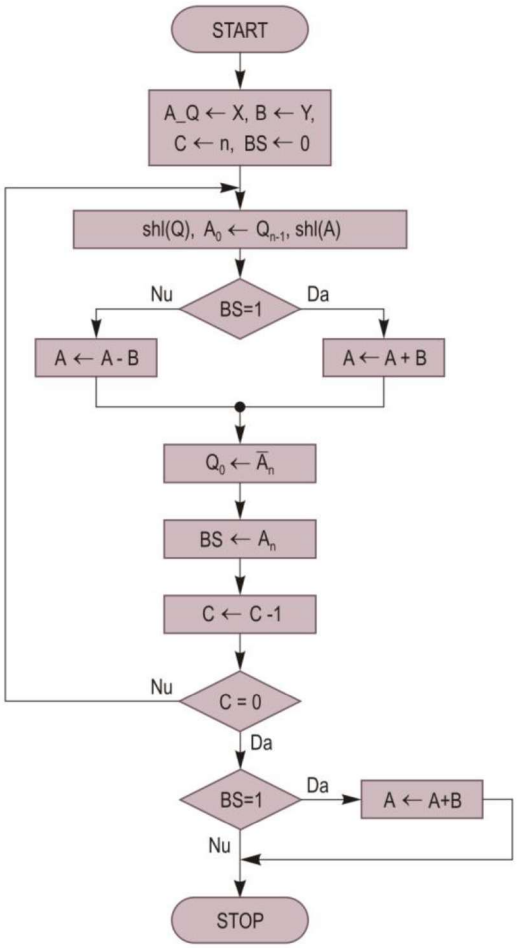

Figura 5.7. Organigrama algoritmului de împărțire fără refacerea restului parțial pentru numere fără semn.

Etapa de inițializare este similară cu cea de la metoda cu refacerea restului parțial. Se inițializează în plus bistabilul suplimentar BS cu 0. În fiecare etapă a operației, se deplasează registrele A și Q la stânga cu o poziție, iar apoi se efectuează fie o scădere a împărțitorului din restul parțial, fie o adunare a împărțitorului la restul parțial, în funcție de starea bistabilului BS. Dacă restul parțial este pozitiv ($A_n = 0$), cifra câtului este 1. În etapa următoare se efectuează o scădere, deci bistabilul BS este resetat la 0. Deci, operațiile efectuate sunt următoarele:

Dacă $A_n = 0$: $Q_0 \leftarrow 1$, BS $\leftarrow 0$

Dacă restul parțial este negativ ($A_n = 1$), cifra câtului este 0. În etapa următoare se efectuează o adunare, deci bistabilul BS este setat la 1:

Dacă $A_n = 1$: $Q_0 \leftarrow 0$, BS $\leftarrow 1$

În funcție de semnul restului parțial, operațiile efectuate sunt următoarele:

$Q_0 \leftarrow \overline{A}_n$, BS $\leftarrow A_n$

Aceste operații se repetă de *n* ori. Dacă după ultima operație BS = 0 (deci, s-a obținut un rest parțial pozitiv), restul final este corect. În caz contrar, restul trebuie corectat prin adunarea împărțitorului la restul parțial. Restul se obține în registrul A, iar câtul în registrul Q.

Exemplul 5.7

Considerăm numerele $X = 14$, $Y = 3$. Reprezentarea binară a acestor numere este următoarea:

$X = 1110_2$
$Y = 0011_2$

Valoarea negativă a împărțitorului, exprimată în C2, este:

$-Y\,(C2) = 1\ 1101_2$

Execuția operației este prezentată în tabelul 5.7.

Tabelul 5.7. Execuția operației de împărțire 14 : 3 prin metoda fără refacerea restului parțial.

Pas	A	Q	B	Q_0	BS	C	Operații
0	0 0000	1110	0011	0	0	4	Inițializare
1	0 0001 + 1 1101 1 1110 +	1100 1100	0011	0	1	3	Deplasare A_Q la stânga Scădere împărțitor
2	1 1101 + 0 0011 0 0000 +	1000 1001	0011	1	0	2	Deplasare A_Q la stânga Adunare împărțitor
3	0 0001 + 1 1101 1 1110	0010 0010	0011	0	1	1	Deplasare A_Q la stânga Scădere împărțitor
4	1 1100 + 0 0011 1 1111 +	0100 0100	0011	0	1	0	Deplasare A_Q la stânga Adunare împărțitor
5	0 0011 0 0010	 0100	0011	-	0	0	Adunare împărțitor

Câtul obținut în registrul Q este $0100_2 = 4$, iar restul obținut în registrul A este $0\ 0010_2 = 2$.

5.3 Module pentru circuite aritmetice secvențiale

Pentru implementarea circuitelor aritmetice secvențiale, se pot utiliza modulele descrise în continuare. Aceste module sunt similare cu componentele disponibile în diferite biblioteci pentru proiectarea cu circuite logice programabile.

5.3.1 Registru de n biți cu resetare sincronă

Intrarea de ceas a registrului FDN (figura 5.8) este *Clk*, intrarea de date este vectorul *D* de *n* biți, iar ieșirea de date este vectorul *Q* de *n* biți. Dacă semnalul *Rst* este 1 logic, registrul este resetat în mod sincron la frontul crescător al semnalului de ceas, ieșirile sale fiind resetate la 0 logic. În caz contrar, dacă semnalul *CE* (*Clock Enable*) este 1 logic, se realizează încărcarea paralelă a registrului cu datele aplicate la intrare.

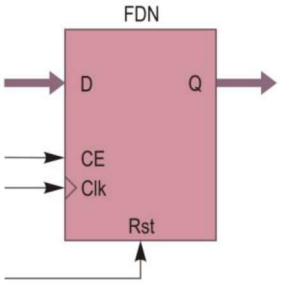

Figura 5.8. Simbolul registrului FDN.

5.3.2 Bistabil cu resetare sincronă

Intrarea de ceas a bistabilului FD (figura 5.9) este *Clk*, intrarea de date este *D*, iar ieșirea de date este *Q*. Funcționarea acestui bistabil este similară cu cea a registrului FDN, cu deosebirea că intrarea de date și ieșirea de date sunt de câte un bit în locul unor vectori.

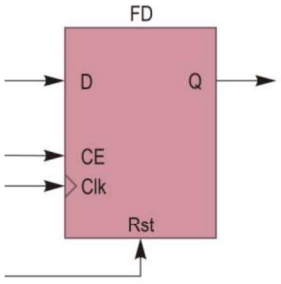

Figura 5.9. Simbolul bistabilului FD.

5.3.3 Registru de deplasare la dreapta de n biți cu resetare sincronă

Intrarea de ceas a registrului de deplasare SRRN (figura 5.10) este *Clk*, intrarea paralelă de date este vectorul *D* de *n* biți, iar intrarea serială de date este *SRI* (*Shift Right Input*). Ieșirea de date este vectorul *Q* de *n* biți. Dacă semnalul *Rst* este 1 logic, registrul este resetat în mod sincron la frontul crescător al semnalului de ceas, ieșirile de date fiind resetate la 0 logic. În caz contrar, dacă semnalul *Load* este 1 logic, se realizează încărcarea paralelă a registrului cu datele aplicate la intrare. Dacă semnalul *Load* nu este 1 logic, iar semnalul *CE* (*Clock Enable*) este 1 logic, se realizează deplasarea registrului cu o poziție la dreapta, iar data de pe intrarea serială *SRI* se încarcă în poziția Q_{n-1}. În mod similar se poate defini un registru de deplasare la stânga.

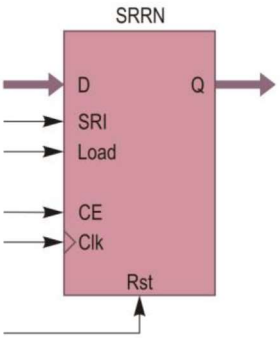

Figura 5.10. Simbolul registrului de deplasare la dreapta SRRN.

5.3.4 Sumator de n biți

Intrările sumatorului ADDN (figura 5.11) sunt operanzii *X* și *Y* care trebuie adunați, de câte *n* biți, și transportul de intrare *Tin*. Ieșirile sumatorului sunt suma *S* de *n* biți, transportul de ieșire *Tout* și semnalul de depășire *OVF* (*Overflow*). Ieșirea *OVF* se utilizează numai la adunarea numerelor cu semn.

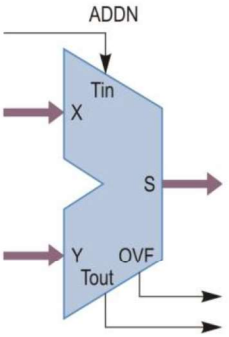

Figura 5.11. Simbolul sumatorului ADDN.

Aplicații

5.1 Răspundeți la următoarele întrebări:

a. Care sunt operațiile efectuate în fiecare pas al înmulțirii prin deplasare și adunare?

b. Indicați conținutul registrelor în fiecare pas al unei operații de înmulțire prin metoda Booth, similar cu exemplul prezentat în tabelul 5.4. Operanzii vor fi aleși în mod arbitrar, dar trebuie să fie diferiți de cei din exemplul prezentat.

c. Care sunt operațiile efectuate în fiecare pas al împărțirii fără refacerea restului parțial?

5.2 Proiectați un circuit de înmulțire secvențială prin metoda Booth pentru două numere cu semn cu n biți de mărime și un bit de semn. Intrările circuitului de înmulțire sunt următoarele:

- *Clk*: Semnal de ceas;
- *Rst*: Semnal de resetare sincronă;
- *Start*: Semnal pentru începerea operației de înmulțire;
- *X*: Vector de n biți care conține deînmulțitul;
- *Y*: Vector de n biți care conține înmulțitorul.

Ieșirile circuitului de înmulțire sunt următoarele:

- *A*: Vector de n biți care conține partea mai semnificativă a produsului;
- *Q*: Vector de n biți care conține partea mai puțin semnificativă a produsului;
- *Term*: Semnal care indică prin valoarea logică 1 terminarea operației.

Pentru proiectarea circuitului de înmulțire, parcurgeți etapele următoare.

1. Creați fișiere VHDL pentru fiecare din modulele FDN, FD, SRRN și ADDN descrise în secțiunea 5.3. Utilizați un generic n pentru a specifica dimensiunea modulelor. Descrieți aceste module utilizând procese.

   ### Observație

   Recapitulați utilizarea genericelor, care sunt prezentate în Anexa C, *Proiectarea structurală în limbajul VHDL*, secțiunea C.5.

2. Pe baza organigramei operației de înmulțire prin metoda Booth (figura 5.4), desenați diagrama de stare a unității de comandă al circuitului.

3. Definiți semnalele de comandă necesare funcționării circuitului, pe baza operațiilor care trebuie efectuate în fiecare etapă și a descrierii modulelor utilizate.

4. Creați un modul VHDL pentru unitatea de comandă. Intrările acestei unități sunt semnalele *Clk*, *Rst*, *Start*, bitul Q_0 al registrului Q și conținutul bistabilului Q_{-1}.

Ieșirile unității de comandă sunt semnalele de comandă pentru controlul funcționării circuitului și semnalul *Term*. Pentru numărătorul *C*, definiți un semnal și utilizați operatorul de scădere pentru decrementarea acestuia. Descrieți unitatea de comandă sub forma unui automat de stare.

5. Creați un fișier VHDL pentru modulul circuitului de înmulțire, cu intrările și ieșirile specificate la începutul descrierii acestei aplicații. În acest fișier, instanțiați direct entitățile modulelor utilizate și conectați porturile lor conform schemei bloc a circuitului de înmulțire Booth (figura 5.3). La instanțierea entităților FDN, ADDN și SRRN utilizați clauza `generic map` pentru a specifica dimensiunea necesară (8) a porturilor pentru înmulțirea numerelor cu 7 biți de mărime și un bit de semn.

6. Creați un fișier al bancului de test pentru circuitul de înmulțire. În bancul de test, generați mai întâi vectori de intrare *X* și *Y* cu valori apropiate de 0 (de exemplu, între –2 și 2). După generarea unei perechi de valori, activați semnalul *Start* și așteptați un timp suficient (de exemplu, 300 ns) pentru ca operația să fie terminată, după care verificați dacă rezultatul generat este corect. Simulați funcționarea circuitului de înmulțire cu simulatorul Vivado.

   ### Observații

   - Specificați utilizarea pachetelor **STD_LOGIC_SIGNED** și **STD_LOGIC_ARITH** din biblioteca IEEE în fișierul bancului de test.
   - La execuția simulării, în fereastra *Wave* selectați semnalele *A* și *Q*, executați un clic cu butonul din dreapta pe selecția semnalelor, alegeți opțiunea *New Virtual Bus* și introduceți un nume pentru semnalul creat. Acest semnal va conține rezultatul operației, reprezentând concatenarea semnalelor *A* și *Q*. Pentru a se afișa rezultatul ca un număr zecimal cu semn, se poate specifica opțiunea *Radix* → *Signed Decimal* pentru acest semnal.

7. Modificați fișierul bancului de test pentru a genera vectori de intrare *X* și *Y* cu valori apropiate de cele pozitive maxime (de exemplu, între 125 și 127), respectiv cu valori negative apropiate de cele minime (de exemplu, între –127 și –125), verificând dacă rezultatele generate sunt corecte. Simulați din nou funcționarea circuitului cu simulatorul Vivado.

5.3 Implementați pe o placă de dezvoltare circuitul de înmulțire prin metoda Booth creat pentru aplicația 5.2. În continuare, se presupune utilizarea plăcii de dezvoltare Nexys4 DDR sau Nexys A7-100T, dar se poate utiliza orice placă cu 16 comutatoare, două butoane și un afișaj cu șapte segmente conținând patru sau mai multe cifre. Deînmulțitul *X* va fi introdus de la primele opt comutatoare din stânga, SW15..SW8, iar înmulțitorul *Y* va fi introdus de la celelalte opt comutatoare, SW7..SW0. Semnalul *Start* va fi generat prin butonul BTNU. Semnalul

de resetare va fi generat prin butonul BTND. Starea comutatoarelor va fi afișată pe primele patru cifre ale afișajului cu șapte segmente, iar rezultatul va fi afișat pe ultimele patru cifre ale afișajului. Semnalul *Term* va fi afișat la una din diodele LED.

Parcurgeți etapele următoare pentru implementarea circuitului de înmulțire.

1. Adăugați la proiect fișierul sursă al multiplexorului pentru afișajul cu șapte segmente `displ7seg` creat pentru aplicația 1.3 din capitolul 1.
2. Creați un fișier sursă VHDL pentru modulul principal al circuitului de înmulțire. Porturile de intrare ale acestui modul vor fi *Clk*, *Rst*, *Start*, *X*(7:0) și *Y*(7:0). Porturile de ieșire ale modulului principal vor fi *An*(7:0), *Seg*(7:0) și *Term*.
3. În fișierul sursă al modulului principal, instanțiați entitatea circuitului de înmulțire și entitatea multiplexorului pentru afișajul cu șapte segmente. Declarați semnalele suplimentare care trebuie conectate la porturile acestor entități.
4. Adăugați la proiect un fișier de constrângeri care a fost utilizat într-un proiect anterior și modificați fișierul pentru ca numele semnalelor să corespundă cu cele ale porturilor de intrare și de ieșire.
5. Realizați elaborarea proiectului și corectați erorile, dacă există.
6. Setați opțiunile pentru sinteza și implementarea proiectului. Selectați strategia de rulare *Flow_RuntimeOptimized* pentru sinteză și pentru implementare.
7. Realizați sinteza și implementarea proiectului, iar apoi generați fișierul cu șirul de biți pentru configurarea circuitului FPGA.
8. Conectați placa de dezvoltare la un port USB al calculatorului și verificați funcționarea circuitului de înmulțire.

5.4 Proiectați un circuit de înmulțire secvențială prin metoda deplasării și adunării pentru două numere fără semn de câte 8 biți. Utilizați aceleași module ca și pentru circuitul de înmulțire prin metoda Booth, cu excepția modulului FD. Simulați funcționarea circuitului de înmulțire cu simulatorul Vivado.

5.5 Proiectați un circuit de împărțire secvențială pentru două numere fără semn utilizând metoda refacerii restului parțial. Deîmpărțitul este un număr de 16 biți, iar împărțitorul este un număr de 8 biți. Utilizați aceleași module ca și pentru circuitul de înmulțire prin metoda Booth, cu excepția modulului FD. Simulați funcționarea circuitului de împărțire cu simulatorul Vivado.

5.6 Modificați circuitul de împărțire proiectat pentru aplicația 5.5 pentru a utiliza metoda fără refacerea restului parțial. Simulați funcționarea circuitului de împărțire cu simulatorul Vivado.