

Arhitectura Calculatoarelor

Oprițoiu Flavius
flavius.opritoiu@cs.upt.ro

4 Decembrie 2024

11 Decembrie 2024

18 Decembrie 2024

8 Ianuarie 2025

15 Ianuarie 2025

Cap. 4 Analiza Funcțională și Sinteza Dispozitivelor de Înmulțire Binară

4.1 - Metode de înmulțire

Un înmulțitor calculează produsul $P = X \cdot Y$, unde

- ▶ operandul X se numește înmulțitor,
- ▶ operandul Y se numește deînmulțit

Se consideră operanzii fără semn X și Y , pe 4 biți:

$$X = 11_{(10)} = 1011_{(2)}$$

$$Y = 12_{(10)} = 1100_{(2)}$$

4.1 - Metode de înmulțire (contin.)

Ⓐ: Paper and pencil

$$\begin{array}{r}
 \begin{array}{cccccccc}
 & & & & 1 & 1 & 0 & 0 & \text{-----} & Y \\
 & & & & 1 & 0 & 1 & 1 & = x_3 x_2 x_1 x_0 & \text{---} & X \\
 \hline
 & & & & 1 & 1 & 0 & 0 & \text{-----} & x_0 Y 2^0 \\
 & & & 1 & 1 & 0 & 0 & \text{-----} & x_1 Y 2^1 \\
 & & 0 & 0 & 0 & 0 & \text{-----} & x_2 Y 2^2 \\
 & 1 & 1 & 0 & 0 & \text{-----} & x_3 Y 2^3 \\
 \hline
 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \text{---} & P = \sum_{i=0}^3 x_i Y 2^i = 132
 \end{array}
 \end{array}$$

Investiție hardware:

- ▶ 2 registre pe 4 biți pentru X și Y
- ▶ sumator multi-operand
- ▶ "gating"-ul de înmulțitului

4.1 - Metode de înmulțire (contin.)

(B): Păstrarea fixă a produselor parțiale

	1	1	0	0	-----	Y					
	1	0	1	1	$= x_3 x_2 x_1 x_0$	---	X				
0	0	0	0	0	0	0	0	0	-----	$P_0 := 0$	
	1	1	0	0	-----	$x_0 Y 2^0$					+
0	0	0	0	1	1	0	0	0	-----	$P_1 := P_0 + x_0 Y 2^0$	
	1	1	0	0	-----	$x_1 Y 2^1$					+
0	0	1	0	0	1	0	0	0	-----	$P_2 := P_1 + x_1 Y 2^1$	
	0	0	0	0	-----	$x_2 Y 2^2$					+
0	0	1	0	0	1	0	0	0	-----	$P_3 := P_2 + x_2 Y 2^2$	
	1	1	0	0	-----	$x_3 Y 2^3$					+
1	0	0	0	0	1	0	0	0	-----	$P_4 := P_3 + x_3 Y 2^3 = P$	

4.1 - Metode de înmulțire (contin.)

Ⓑ: Păstrarea fixă a produselor parțiale (continuare)

Înmulțirea este realizată printr-o secvență de pași, cu pasul de iterație:

$$P_{i+1} = P_i + x_i \cdot Y \cdot 2^i, \text{ pentru } i \geq 0, P_0 := 0$$

P_i este un produs parțial și expresia $x_i \cdot Y \cdot 2^i$ este un produs de 1 bit.

Investiție hardware:

- ▶ 2 registre de 4 biți pentru X și Y
- ▶ registru de 8 biți pentru produsele parțiale
- ▶ sumator pe 8 biți
- ▶ mecanism de aliniere a produselor de 1 bit

4.1 - Metode de înmulțire (contin.)

Ⓒ: Păstrarea fixă a produselor de 1 bit

	1	1	0	0	-----	Y				
	1	0	1	1	$= x_3x_2x_1x_0$	---	X			
0000	0	0	0	0	-----	$P_0 := 0$				
	1	1	0	0	-----	x_0Y	+			
0000	1	1	0	0	-----	$P_0 := P_0 + x_0Y$				
000	0	1	1	0	0	-----	$P_1 := P_0 \cdot 2^{-1}$			
	1	1	0	0	-----	x_1Y	+			
001	0	0	1	0	0	-----	$P_1 := P_1 + x_1Y$			
00	1	0	0	1	0	0	-----	$P_2 := P_1 \cdot 2^{-1}$		
	0	0	0	0	-----	x_2Y	+			
00	1	0	0	1	0	0	-----	$P_2 := P_2 + x_2Y$		
0	0	1	0	0	1	0	0	-----	$P_3 := P_2 \cdot 2^{-1}$	
	1	1	0	0	-----	x_3Y	+			
1	0	0	0	0	1	0	0	-----	$P_3 := P_3 + x_3Y$	
	1	0	0	0	0	1	0	0	--	$P_4 := P_3 \cdot 2^{-1} = P$

4.1 - Metode de înmulțire (contin.)

Ⓒ: Păstrarea fixă a produselor de 1 bit (continuare)

Înmulțirea este realizată printr-o secvență de pași, cu pasul de iterație:

$$\begin{cases} P_i = P_i + x_i \cdot Y \\ P_{i+1} = P_i \cdot 2^{-1} \end{cases}, \text{ pentru } i \geq 0, P_0 := 0$$

Investiție hardware:

- ▶ 2 registre de 4 biți pentru X și Y
- ▶ registru de 8 biți pentru produsele parțiale cu facilitare de deplasare la dreapta
- ▶ sumator pe 4 biți

4.2 - Înmulțire binară secvențială pentru numere în Semn-Mărire

Fie X și Y 2 numere fracționare, pe 8 biți, reprezentate în S.-M.

$$\begin{array}{rcccccccc} X & = & x_7 & . & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 \\ Y & = & y_7 & . & y_6 & y_5 & y_4 & y_3 & y_2 & y_1 & y_0 \\ & & \underbrace{}_{\text{semn}} & & \underbrace{}_{2^{-1}} & \underbrace{}_{2^{-2}} & \underbrace{}_{2^{-3}} & \underbrace{}_{2^{-4}} & \underbrace{}_{2^{-5}} & \underbrace{}_{2^{-6}} & \underbrace{}_{2^{-7}} \end{array}$$

←————— ponderi —————→

Produsul $P = X \cdot Y$ este un număr fracționar în SM, pe 16 biți

$$P = p_{15} . p_{14} p_{13} \cdots p_2 p_1 p_0$$

- ▶ Most Significant Bit (MSB) este semnul: $p_{15} = x_7 \oplus y_7$
- ▶ părțile de magnitudine ale operandilor au 7 biți \Rightarrow magnitudinea produsului va avea 14 biți: $p_{14}, p_{13}, \dots, p_1$
- ▶ al 16-lea bit este p_0 și are valoarea 0

4.2 - Înmulțire binară secvențială pentru numere în Semn-Mărire (contin.)

multiplier 2

declare register $A[7 : 0]$, $Q[7 : 0]$, $M[7 : 0]$, $COUNT[2 : 0]$;

declare bus $INBUS[7 : 0]$, $OUTBUS[7 : 0]$;

BEGIN : $A := 0$, $COUNT := 0$, } \leftarrow -----{ c_0 }

INPUT : $M := INBUS$;

$Q := INBUS$; \leftarrow -----{ c_1 }

TEST1 : if $Q[0] = 0$ then go to *RSHIFT*,

ADD : $A[7 : 0] := A[6 : 0] + M[6 : 0]$; \leftarrow -----{ c_2 }

RSHIFT : $A[7] := 0$, $A[6 : 0].Q := A.Q[7 : 1]$, } \leftarrow -----{ c_3 }

INCR. : $COUNT := COUNT + 1$;

TEST2 : if $COUNT \neq 1$ then go to *TEST1*,

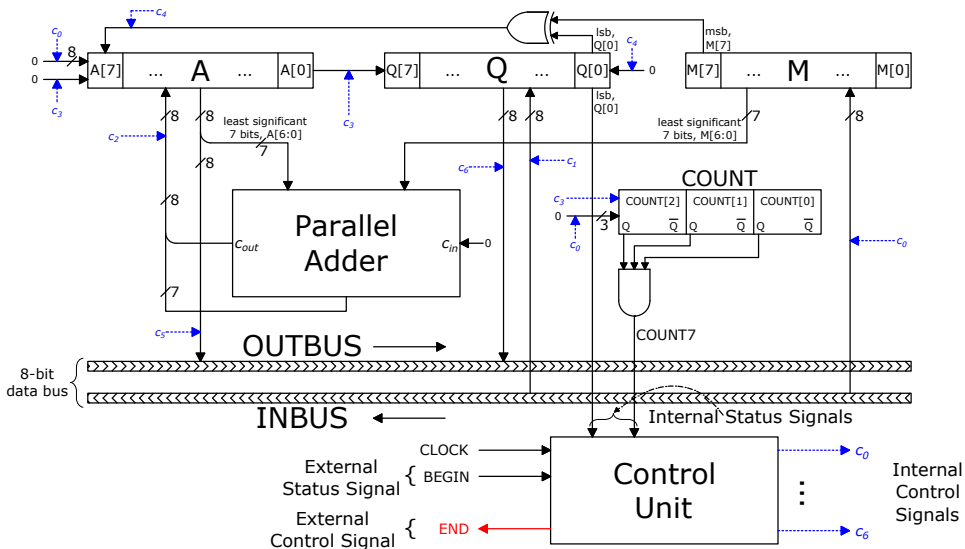
SIGN : $A[7] := Q[0]$ exor $M[7]$, $Q[0] := 0$; \leftarrow -----{ c_4 }

OUTPUT : $OUTBUS := A$; \leftarrow -----{ c_5 }

$OUTBUS := Q$; \leftarrow -----{ c_6 }

END : ----- \rightarrow { **END** }

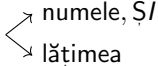
4.2 - Înmulțire binară secvențială pentru numere în Semn-Mărire (contin.)



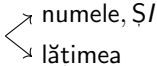
4.2 - Înmulțire binară secvențială pentru numere în Semn-Mărime (contin.)

Pseudolimbajul folosit de algoritm:


1. declare registers

- ▶ definește regiștri; specifică  numele, Ș/
lățimea
- ▶ operatorul de concatenare: •; ex. $A[6 : 0].Q := A.Q[7 : 1]$

2. declare bus

- ▶ definește magistrale specificând  numele, Ș/
lățimea

3. Execuție sincronă:

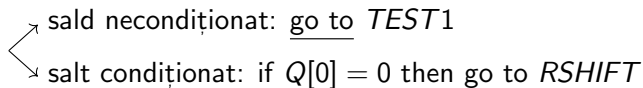
- 
- operații non-conflictuale: executate concurent, separate prin 9
 - operații secvențiale: executate secvențial, separate prin 9

4.2 - Înmulțire binară secvențială pentru numere în Semn-Mărire (contin.)

Pseudolimbajul folosit de algoritm (continuare):

4. Operatorul de atribuire este $\boxed{:=}$ și este folosit pentru încărcarea de valori binare în regiștri sau magistrale
- ▶ exor indică o operație cablată: în $A[7] := Q[0] \text{ exor } M[7]$ este implementată printr-o poartă EXOR

5. Controlul fluxului de execuție:



6. Citire/scriere simultană din/în registre:

$$A[7] := Q[0] \text{ exor } M[7], Q[0] := 0$$

- ▶ bit-ul $Q[0]$ este atât citit cât și scris în același ciclu de ceas

4.2 - Înmulțire binară secvențială pentru numere în Semn-Mărire (contin.)

Elementele platformei HW:

- ▶ acumulatorul A : este folosit pentru adunarea produselor de 1-bit la produsul parțial; are facilități de deplasare la dreapta; stochează biții mai semnificativi ai produselor parțiale
- ▶ registrul înmulțitor Q : încărcat, inițial, cu înmulțitorul X ; are facilități de deplasare la dreapta (numele Q provine de la platforma HW specifică împărțirii binare)
- ▶ registrul deînmulțit M : stochează deînmulțitul Y ;
- ▶ sumatorul paralel: pe 7 biți; utilizat pentru adunarea produselor de 1-bit la produsul parțial
- ▶ contorul $COUNT$: păstrează evidența numărului de iterații executate
- ▶ unitatea de control generează semnalele de control (c_0, \dots, c_6 , **END**) în secvența corectă de execuție a algoritmului

4.2 - Înmulțire binară secvențială pentru numere în Semn-Mărimă (contin.)

Algoritmul folosește metoda a 3-a de înmulțire (păstrarea fixă a produselor de 1-bit):

$$\begin{cases} P_i = P_i + x_i \cdot Y & \longrightarrow \text{etichetele TEST1 și ADD} \\ P_{i+1} = P_i \cdot 2^{-1} & \longrightarrow \text{eticheta RSHIFT} \end{cases}$$

Produsele parțiale:

- ▶ până la setarea semnului rezultatului, (eticheta **SIGN**), toate produsele parțiale sunt pozitive, reprezentate în S.-M., independent de semnele operanzilor
- ▶ la începutul algoritmului, P_0 ocupă registrul A ($P_0 := 0$)
- ▶ după prima iterație, P_1 ocupă întreg registrul A și MSB-ul lui Q (P_1 este în $A[7 : 0].Q[7]$)
- ▶ după a doua iterație, P_2 ocupă întreg A -ul și primii 2 MSBs ai lui Q (P_2 este în $A[7 : 0].Q[7 : 6]$)
- ▶ în general, P_{i+1} ocupă un bit suplimentar în Q la fiecare nouă iterație; aceasta se petrece la eticheta **RSHIFT**, unde A concatenat cu Q este deplasat la dreapta

4.2 - Înmulțire binară secvențială pentru numere în Semn-Mărire (contin.)

În fiecare iterație, registrul Q este deplasat la dreapta:

- ▶ la începutul algoritmului, $Q[0]$ conține pe x_0
- ▶ după prima iterație, $Q[0]$ conține pe x_1
- ▶ după a doua iterație, $Q[0]$ conține pe x_2

⇒ în oricare iterație, bitul curent al lui X , x_i , se află în $Q[0]$.

Acesta este motivul pentru care la **TEST1**, algoritmul testează bitul $Q[0]$.

Sumator paralel:

- ▶ pe 7 biți pentru că produsul parțial P_i și deînmulțitul Y sunt în S.-M. iar în S.-M. adunarea nu poate calcula semnul rezultatului în mod corect
- ▶ transportul de ieșire al adunării este stocat în $A[7]$
 - ▶ ⇒ se evită *overflow*-ul

4.2 - Înmulțire binară secvențială pentru numere în Semn-Mărire (contin.)

Contorul:

- ▶ numără 7 iterații:
 - ▶ pentru că mărimea înmulțitorului X are o lățime de 7 biți
- ▶ incrementat la eticheta **RSHIFT**
- ▶ semnalul $COUNT7$ este activat când conținutul numărătorului este 7 ($7_{(10)} = 111_{(2)}$)

Unitatea de control:

- ▶ secvențiază operațiile prin activarea semnalelor de control (c_0 , ..., c_6 , **END**)
- ▶ în fiecare ciclu de tact, cel puțin un semnal de control este activat
- ▶ semnalele de control sunt reprezentate prin linie întreruptă
-----▶●
- ▶ liniile de date sunt reprezentate prin linie solidă ———

4.2 - Înmulțire binară secvențială pentru numere în Semn-Mărire (contin.)

Se consideră operanzii pe 4 biți, în S.-M., fracționari:

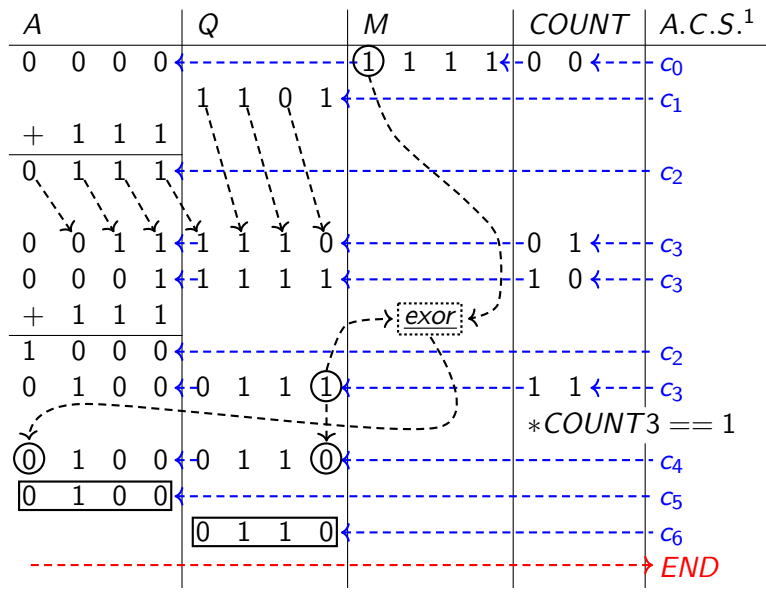
$$X = -0.625 = -5 \cdot 2^{-3} = 1.101_{S.-M.}$$

$$Y = -0.875 = -7 \cdot 2^{-3} = 1.111_{S.-M.}$$

Produsul P ai celor 2 operanzi, reprezentat în S.-M., pe 8 biți:

$$\begin{aligned} P &= X \cdot Y = (-5 \cdot 2^{-3}) \cdot (-7 \cdot 2^{-3}) = 35 \cdot 2^{-6} \\ &= 0.1000110_{S.-M.} \end{aligned}$$

4.2 - Înmulțire binară secvențială pentru numere în Semn-Mărire (contin.)



¹Semnale de control activate

4.2 - Înmulțire binară secvențială pentru numere în Semn-Mărire (contin.)

Se consideră operanzii pe 4 biți, în S.-M., întregi:

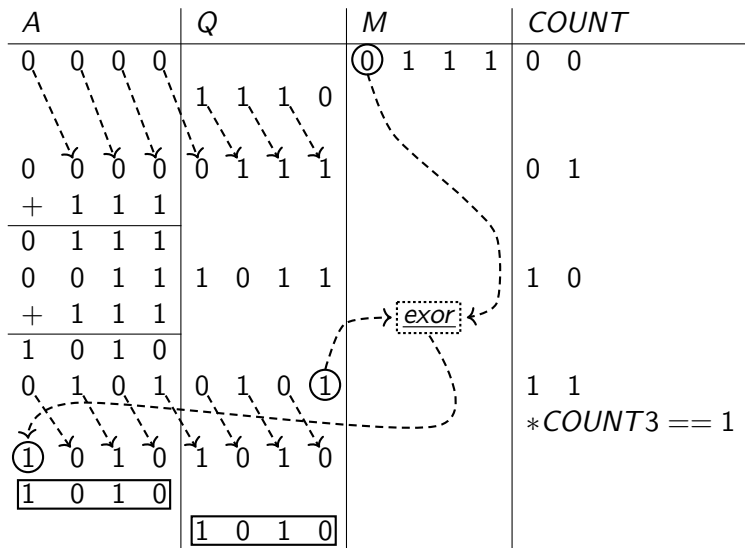
$$X = -6 = 1110_{S.-M.}$$

$$Y = +7 = 0111_{S.-M.}$$

Produsul P ai celor 2 operanzi este obținut ca:

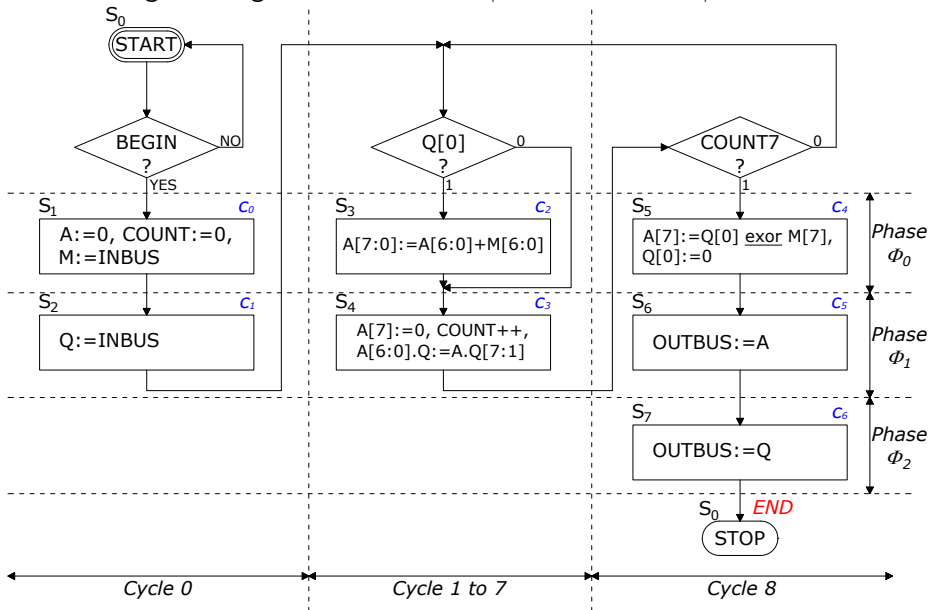
$$\begin{aligned} P &= X \cdot Y = -6 \cdot +7 = -35 \\ &= 10101010_{S.-M.} \end{aligned}$$

4.2 - Înmulțire binară secvențială pentru numere în Semn-Mărire (contin.)



4.3 - Elemente de sinteza unităților de control

Ordinograma algoritmului de înmulțire binară secvențială în S.-M.:



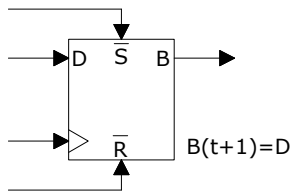
4.3 - Elemente de sinteza unităților de control (contin.)

(A): Metoda tabelului de stare

(B): Metoda One Hot

- ▶ utilizează câte un element de stocare pentru fiecare stare
- ▶ la orice moment de timp, doar un element de stocare este activ \Rightarrow
acel element se spune că este "hot"

Se vor utiliza bistabile de tip D:



Note: ieșirea bistabilelor este notată B pentru a evita suprapunerea cu biți ai registrului Q .

4.3 - Elemente de sinteza unităților de control (contin.)

(B): Metoda One Hot (continuare)

Fiecare stare are asociat un bistabil \Rightarrow sunt folosite 8 variabile de stare: $B_0, B_1, B_2, B_3, B_4, B_5, B_6, B_7$; codificarea stării este prezentată în tabelul următor:

State	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
S_0	0	0	0	0	0	0	0	1
S_1	0	0	0	0	0	0	1	0
S_2	0	0	0	0	0	1	0	0
S_3	0	0	0	0	1	0	0	0
S_4	0	0	0	1	0	0	0	0
S_5	0	0	1	0	0	0	0	0
S_6	0	1	0	0	0	0	0	0
S_7	1	0	0	0	0	0	0	0

4.3 - Elemente de sinteza unităților de control (contin.)

Ⓑ: Metoda One Hot (continuare)

Ecuatii de feedback

$$D_0 = B_0 \cdot \overline{BEGIN} \text{ or } B_7$$

$$D_1 = B_0 \cdot BEGIN$$

$$D_2 = B_1$$

$$D_3 = B_2 \cdot Q[0] \text{ or } B_4 \cdot \overline{COUNT7} \cdot Q[0]$$

$$D_4 = B_2 \cdot \overline{Q[0]} \text{ or } B_3 \text{ or } B_4 \cdot \overline{COUNT7} \cdot \overline{Q[0]}$$

$$D_5 = B_4 \cdot COUNT7$$

$$D_6 = B_5$$

$$D_7 = B_6$$

Ecuatii de ieșire

$$c_0 = B_1$$

$$c_1 = B_2$$

$$c_2 = B_3$$

$$c_3 = B_4$$

$$c_4 = B_5$$

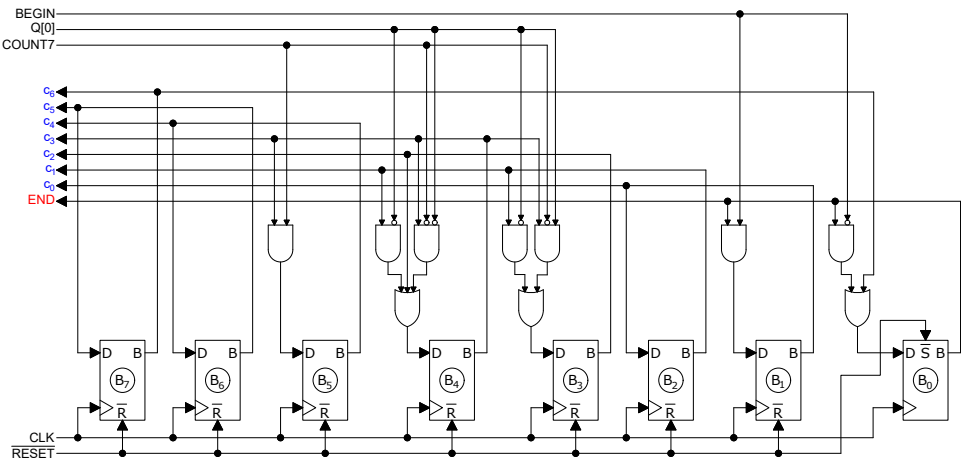
$$c_5 = B_6$$

$$c_6 = B_7$$

$$END = B_0$$

4.3 - Elemente de sinteza unităților de control (contin.)

(B): Metoda One Hot (continuare)

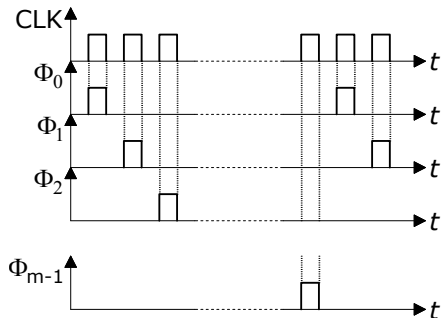


Notă: Arhitectura este afectată de **clock skew**

4.3 - Elemente de sinteza unităților de control (contin.)

Ⓒ: Metoda Sequence Counter

- ▶ construită în jurul unui *Sequence Counter*
 - ▶ un numărător de secvență generează la ieșiri impulsuri de fază non-suprapuse ($\Phi_0, \Phi_1, \Phi_2, \dots, \Phi_{m-1}$) ca mai jos

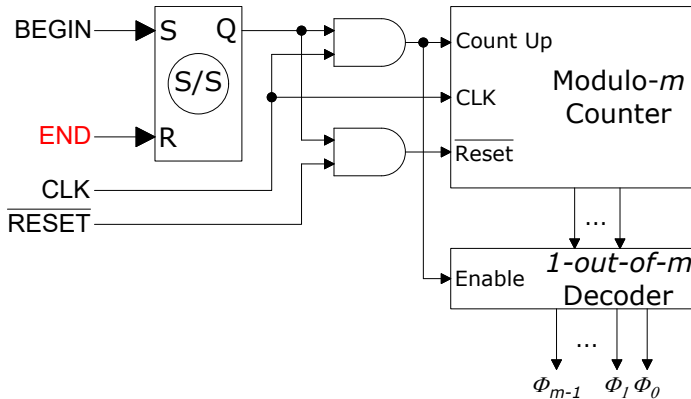


- ▶ Impulsurile de fază sunt folosite pentru a activa semnale de control
- ▶ Datorită naturii repetitive a impulsurilor de fază, metoda Sequence Counter poate executa secvențe iterative

4.3 - Elemente de sinteza unităților de control (contin.)

(C): Metoda Sequence Counter (continuare)

Numărătorul de secvență încorporează un numărător modulo- m , a cărui ieșiri sunt decodificate în m impulsuri de fază

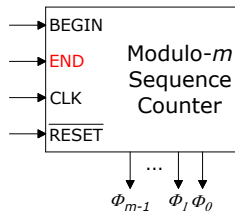


S/S : Latch-ul Start/Stop controlează avansarea în secvența de numărare

4.3 - Elemente de sinteza unităților de control (contin.)

Ⓒ: Metoda Sequence Counter (continuare)

Simbolul numărătorului de secvență este ilustrat mai jos:

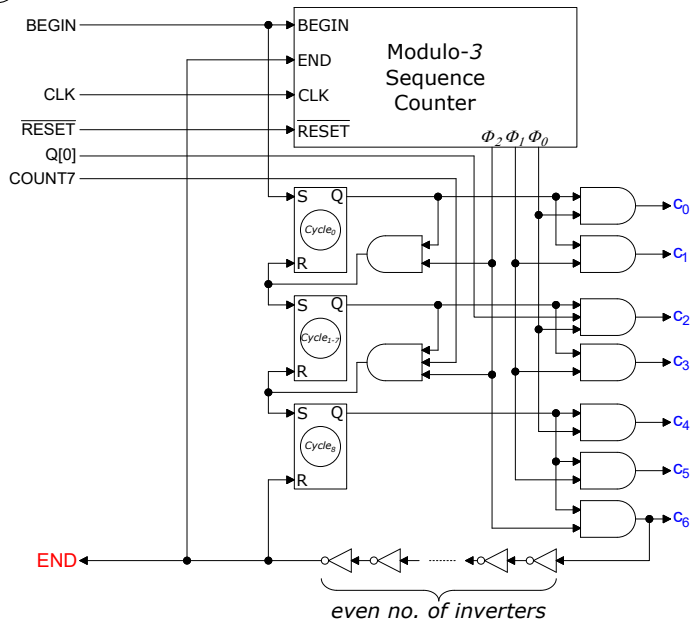


Determinarea lui m (valoarea modulo a numărătorului):

- ▶ m este dat de cel mai lung ciclu
- ▶ pornind de la ordinogramă, se obține $m = 3$: numărătorul de secvență va avea 3 faze

4.3 - Elemente de sinteză unităților de control (contin.)

Ⓒ: Metoda Sequence Counter (continuare)



4.3 - Elemente de sinteza unităților de control (contin.)

Ⓒ: Metoda Sequence Counter (continuare)

- ▶ Utilizează câte un latch S/R pentru fiecare ciclu al algoritmului
 - ▶ Latch-ul este activat când execuția algoritmului ajunge în ciclul asociat
- ▶ Semnalul **END** este obținut prin întârzierea semnalului de control c_6 printr-un număr par de inversoare
 - ▶ activarea lui **END** determină dezactivarea lui c_6 : dacă c_6 nu rămîne activ suficient de mult timp pentru terminarea operației asociate ($OUTPUT := Q$), se va adăuga un nou ciclu algoritmului

4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Robertson

Fie X_{C2} un număr pe n biți reprezentat în C2

$$X_{C2} = \underbrace{x_7}_{\text{sign}} \cdot \underbrace{x_6 x_5 x_4 x_3 x_2 x_1 x_0}_{\text{"magnitude" bits}}$$

Conform interpretării lui Robertson, valoarea lui X este:

► pentru X întreg:

$$X = \underbrace{-x_{n-1} \cdot 2^{n-1}}_{\text{Correction}} + \underbrace{0x_{n-2}x_{n-3} \cdots x_1x_0}_{\text{Positive in C2 and S.-M.}}$$

► pentru X fracțional:

$$X = \underbrace{-x_{n-1} \cdot 2^0}_{\text{Correction}} + \underbrace{0.x_{n-2}x_{n-3} \cdots x_1x_0}_{\text{Positive in C2 and S.-M.}}$$

4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Robertson (contin.)

Interpretarea lui Robertson: valoarea lui X , reprezentat în C2 este valoarea numărului pozitiv obținut prin ștegerea bitului de semn al lui X , la care este adunată o corecție. *Observație:* dacă X este pozitiv, termenul de corecție ($-x_{n-1} \cdot 2^{n-1}$ or $-x_{n-1} \cdot 2^0$) devine 0.

Fie X și Y 2 numere fracționare în C2:

$$\begin{aligned}
 X_{C2} \cdot Y &= \left(\underbrace{-x_{n-1} \cdot 2^0}_{X_{SM}^* \cdot Y} + \underbrace{0.x_{n-2}x_{n-3} \cdots x_1x_0}_{\text{SM positive} \stackrel{\text{def}}{=} X_{SM}^*} \right) \cdot Y \\
 &= \underbrace{X_{SM}^* \cdot Y}_{\text{SM "multiplication"}} - \underbrace{x_{n-1} \cdot Y \cdot 2^0}_{\text{Final correction}}
 \end{aligned}$$

4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Robertson (contin.)

Termenul $X_{SM}^* \cdot Y$ reprezintă înmulțirea lui Y cu un înmulțitor în S.-M., deci, va fi realizată pornind de la algoritmul de înmulțire secvențială în S.-M.. Dacă Y este negativ \Rightarrow produsele parțiale vor fi negative.

În consecință, **totate** produsele parțiale au semnul lui Y . Singura excepție la observația anterioară este cazul primelor iterații pe durata cărora produsele parțială rămân 0 (pentru că adăugarea unui bit de semn de 1 la un câmp de magnitudine format doar din biți de 0 produce valoarea 2^{n-1} pe n biți).

Dacă X_{C2} este pozitiv, termenul de corecție finală, $x_{n-1} \cdot Y \cdot 2^0$ devine 0, în consecință pasul de corecție trebuie evitat.

4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Robertson (contin.)

multiplier 3

declare register $A[7 : 0]$, $Q[7 : 0]$, $M[7 : 0]$, $COUNT[2 : 0]$, F ;

declare bus $INBUS[7 : 0]$, $OUTBUS[7 : 0]$;

BEGIN : $A := 0, COUNT := 0, F := 0$ } \leftarrow -----{ c_0 }

INPUT : $M := INBUS$;
 $Q := INBUS$; \leftarrow -----{ c_1 }

TEST1 : if $Q[0] = 0$ then go to *RSHIFT*,

ADD : $A := A + M, F := F$ or $(Q[0]$ and $M[7])$; \leftarrow -----{ c_2 }

RSHIFT : $A[7] := F, A[6 : 0].Q := A.Q[7 : 1]$, } \leftarrow -----{ c_3 }

INCR. : $COUNT := COUNT + 1$;

TEST2 : if $COUNT \neq 1$ then go to *TEST1*,

TEST3 : if $Q[0] = 0$ then go to *OUTPUT*,

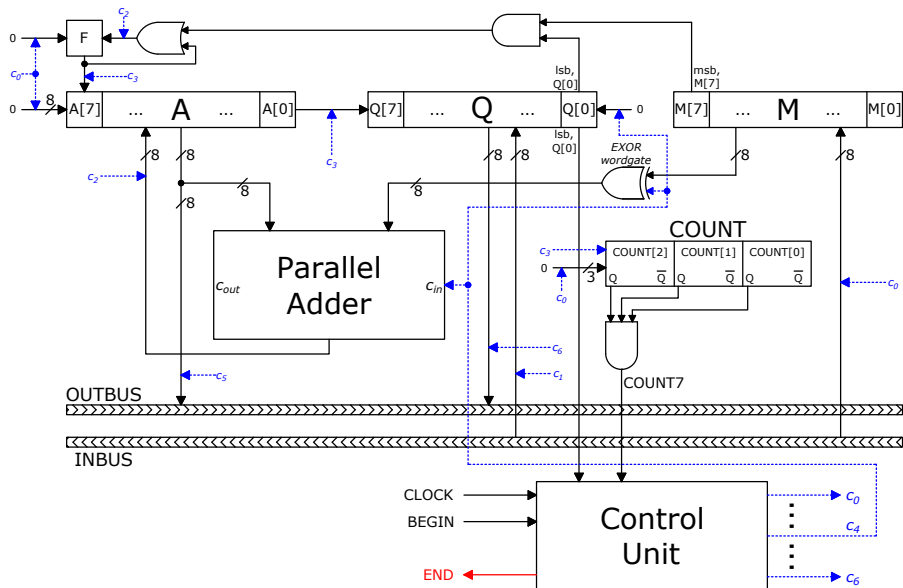
CORRECTION : $A := A - M, Q[0] := 0$; \leftarrow -----{ c_2, c_4 }

OUTPUT : $OUTBUS := A$; \leftarrow -----{ c_5 }

$OUTBUS := Q$; \leftarrow -----{ c_6 }

END : ----- \rightarrow { *END* }

4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Robertson (contin.)



4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Robertson (contin.)

Înmulțitorul Robertson folosește a 3-ametodă de înmulțire (ca și înmulțitorul SM din secțiunea 4.2):

$$\begin{cases} P_i = P_i + x_i \cdot Y & \longrightarrow \text{labels } \textcolor{red}{\text{TEST1}} \text{ and } \textcolor{red}{\text{ADD}} \\ P_{i+1} = P_i \cdot 2^{-1} & \longrightarrow \text{label } \textcolor{red}{\text{RSHIFT}} \end{cases}$$

Bitul curent x_i se află în $Q[0]$.

- ▶ pornind de la $i = 0$, cât timp $x_i = 0$, cu $i \geq 0 \Rightarrow$ produsele parțiale P_i sunt 0
 - ▶ nu se efectuează adunare (salt la eticheta $\textcolor{red}{\text{ADD}}$)
 - ▶ la eticheta $\textcolor{red}{\text{RSHIFT}}$, semnul lui P_i (P_i care este 0) trebuie să rămână 0 și să **nu** fie setat la valoarea semnului lui Y (stocat în $M[7]$)
 - ▶ \Rightarrow flagul F va rămâne la 0 cât timp $x_i = 0$, cu $i \geq 0$ și va fi setat la valoarea lui $M[7]$ odată cu primul bit $x_i = 1$ întâlnit
 - ▶ acesta este motivul pentru care F este actualizat la eticheta $\textcolor{red}{\text{ADD}}$

Din observațiile anterioare rezultă că F este actualizat după ecuația $F := F \underline{\text{or}} (Q[0] \underline{\text{and}} M[7])$

4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Robertson (contin.)

Sumatorul paralel:

- ▶ pe 8 biți ignorând transportul de ieșire
 - ▶ operanzii (din registrele A și M) sunt numere în $C2$
 - ▶ pentru operanzi $C2$, semnul final al produsului nu trebuie corectat (acesta a fost cazul pentru înmulțitorul SM , la eticheta **SIGN**)
 - ▶ în $C2$, semnul rezultatului este calculat corect tratând biții de semn ca oricare bit de magnitudine (motiv pentru care sumatorul are 8 biți și nu doar 7, ca în cazul înmulțitorului SM)
- ▶ un operand (registrul M) folosește un wordgate EXOR
 - ▶ întreaga structură este un sumator/scăzător (vezi Capitolul 2, secțiunea 2.2.3)
 - ▶ eticheta **ADD** folosește sumatorul, eticheta **CORRECTION** - scăzătorul
 - ▶ pentru scădere se activează c_2 (stocarea rezultatului sumatorului în registrul A) și c_4 (transforma sumatorul paralel într-un scăzător paralel)
- ▶ unitatea tratează corect situațiile de overflow
 - ▶ overflow-ul apare când semnul rezultatului este incorect calculat
 - ▶ la eticheta **RSHIFT** semnul corect al produsului parțial ($A[7]$) este restaurat din F (vezi observația anterioară)

4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Robertson (contin.)

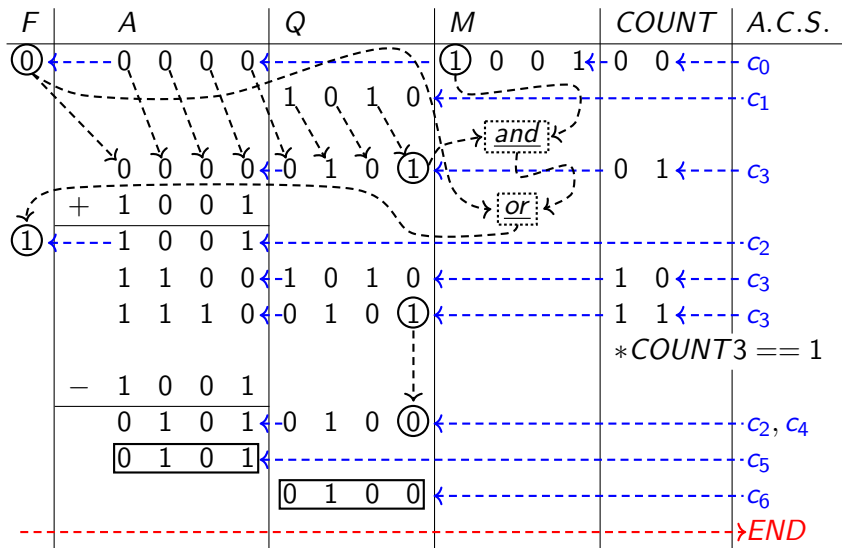
Se consideră operanzii pe 4 biți, în C2, fractionari:

$$\begin{aligned}X &= -0.75 = -3 \cdot 2^{-2} = 1.110_{S.-M.} = 1.010_{C2} \\Y &= -0.875 = -7 \cdot 2^{-3} = 1.111_{S.-M.} = 1.001_{C2}\end{aligned}$$

Produsul P ai celor 2 operanzi, reprezentat în C2, pe 8 biți:

$$\begin{aligned}P &= X \cdot Y = (-3 \cdot 2^{-2}) \cdot (-7 \cdot 2^{-3}) = 21 \cdot 2^{-5} \\&= 0.1010100_{S.-M.} = 0.1010100_{C2}\end{aligned}$$

4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Robertson (contin.)



4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Robertson (contin.)

Se consideră operanzii pe 4 biți, în C2, întregi:

$$X = 6 = 0110.C_2$$

$$Y = -5 = 1011.C_2$$

Produsul P ai celor 2 operanzi, este obținut astfel:

$$\begin{aligned} P &= X \cdot Y = 6 \cdot (-5) = -30 \\ &= 11100010.C_2 \end{aligned}$$

4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Robertson (contin.)

<i>F</i>	<i>A</i>	<i>Q</i>	<i>M</i>	<i>COUNT</i>
① 0	0 0 0 0	0 1 1 0	① 0 1 1	0 0
	0 0 0 0	0 0 1 ①	and	0 1
① +	1 0 1 1	1 0 0 1	or	1 0
	1 0 1 1			
	1 0 0 0	0 1 0 0		1 1
	1 1 0 0	0 0 1 0		*COUNT3 == 1
	1 1 1 0	0 0 1 0		

4.5 - Înmulțirea în Complement de doi bazată pe procedura lui Booth

Pasul de iterație al celei de-a 3-a metodă de înmulțire:

$$\begin{cases} P_i = P_i + x_i \cdot Y \\ P_{i+1} = P_i \cdot 2^{-1} \end{cases}$$

Pentru fiecare bit $x_i = 1$, iterația consumă 2 cicluri de tact (1 pentru adunare, 1 pentru deplasare), pe când dacă $x_i = 0$, consumă un singur ciclu de tact. În consecință, cu cât sunt mai mulți biți de 1 în reprezentarea lui $X \Rightarrow$ cu atât mai multe cicluri de tact revendică înmulțirea, deci, cu atât este mai lentă.

Ideea lui Andrew Booth:


- ▶ inspectarea unei perechi de biți $x_i x_{i-1}$ în fiecare iterație, în locul unui singur bit, x_i
 - ▶ sunt analizate tranziții în interiorul lui X

4.5 - Înmulțirea în Complement de doi bazată pe procedura lui Booth (contin.)

Se consideră X și Y , reprezentate în C2 pe n biți. Fără a pierde din generalitate, se consideră X și Y ca fiind întregi:

Fie următoarea reprezentare a lui X :

$$\begin{array}{ccccccccccccccccc} X & = & x_{n-1} & x_{n-2} & \cdots & x_{i+k+1} & x_{i+k} & \cdots & x_{i+1} & x_i & x_{i-1} & \cdots & x_1 & x_0 \\ & = & 0 & 0 & \cdots & 0 & 1 & \cdots & 1 & 1 & 0 & \cdots & 0 & 0 \end{array}$$


continuous run of $k + 1$ bits of 1s

Produsul $P = X \cdot Y$ este calculat astfel:

$$\begin{aligned} P &= \sum_{j=i}^{i+k} x_j \cdot Y \cdot 2^j = Y(2^{i+k} + 2^{i+k-1} + \cdots + 2^i) \\ &= +2^{i+k+1} \cdot Y - 2^i \cdot Y \end{aligned}$$

4.5 - Înmulțirea în Complement de doi bazată pe procedura lui Booth (contin.)

În locul efectuării a $k + 1$ adunări (pornind de la produsul parțial inițial $P_0 = 0$), produsul poate fi obținut în urma efectuării a doar 2 operații: o adunare și o scădere.

$$\begin{array}{rcl}
 X & = & x_{n-1} \quad x_{n-2} \quad \cdots \quad \boxed{x_{i+k+1} \quad x_{i+k}} \quad \cdots \quad x_{i+1} \quad \boxed{x_i \quad x_{i-1}} \quad \cdots \quad x_1 \quad x_0 \\
 & = & 0 \quad 0 \quad \cdots \quad \boxed{0 \quad 1} \quad \cdots \quad 1 \quad \boxed{1 \quad 0} \quad \cdots \quad 0 \quad 0 \\
 P & = & \quad \quad \quad \boxed{+2^{i+k+1} \cdot Y} \quad \quad \quad \boxed{-2^i \cdot Y}
 \end{array}$$

În pasul i , perechea $x_i x_{i-1} = 10$ și termenul $2^i \cdot Y$ trebuie **scăzut** din produsul parțial curent. În pasul $i + k + 1$, perechea $x_{i+k+1} x_{i+k} = 01$ și termenul $2^{i+k+1} \cdot Y$ trebuie **adunat** la produsul parțial curent.

Pentru perechile $x_j x_{j-1} = 00$ sau $x_j x_{j-1} = 11$, nu se efectuează nicio operație aritmetică.

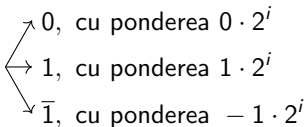
4.5 - Înmulțirea în Complement de doi bazată pe procedura lui Booth (contin.)

În prima iterație se inspectează perechea x_0x_{-1} , care are însă nevoie de extinderea lui X cu bitul x_{-1} în cea mai puțin semnificativă poziție:

- ▶ ponderea lui x_{-1} este $\frac{1}{2}$ din ponderea lui x_0
- ▶ valoarea lui x_{-1} este 0, pentru a nu modifica valoarea lui X

Recodificare Booth:

- ▶ folosește cifre cu semn:

- ▶ cifra x_{iB} poate fi 

- ▶ o cifră cu semn se reprezintă pe cel puțin 2 biți
- ▶ este o reprezentare redundantă

4.5 - Înmulțirea în Complement de doi bazată pe procedura lui Booth (contin.)

Procedura de recodificare Booth:

- ▶ se extinde operandul inițial cu x_{-1} (descriș anterior)
- ▶ se scanează operandul inițial din dreapta spre stânga și se înlocuiește fiecare pereche $x_i x_{i-1}$ potrivit tabelului de mai jos

x_i	x_{i-1}	x_{iB}
0	0	0
0	1	1
1	0	$\overline{1}$
1	1	0

4.5 - Înmulțirea în Complement de doi bazată pe procedura lui Booth (contin.)

Exercițiu: Pentru $X = -1 \cdot 2^{-3}$, pe 4 biți, determinați recodificarea lui Booth

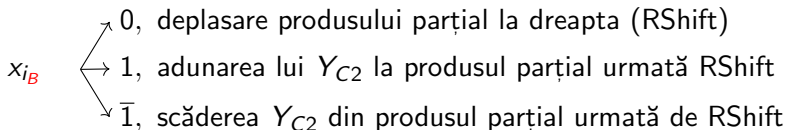
Ranks \ Number	x_3	x_2	x_1	x_0	x_{-1}
	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
$X_{S.-M.}$	1.	0	0	1	
X_{C2}	1.	1	1	1	0
X_B	0.	0	0	$\bar{1}$	

Thus $X_B = \bar{1} \cdot 2^{-3} = -1 \cdot 2^{-3}$.

4.5 - Înmulțirea în Complement de doi bazată pe procedura lui Booth (contin.)

Pentru că $X_B = X_{C2}$, urmează că $X_{C2} \cdot Y_{C2} = X_B \cdot Y_{C2}$

- ▶ în fiecare iterație, dependent de bitul x_{i_B} a lui X_B , următoarele operații au loc:



Pentru că operandu Y poate fi adunat sau scăzut la/din produsul parțial

- ▶ \Rightarrow registrul A poate avea semne diferite în iterații diferite
 - ▶ \Rightarrow deplasarea la dreapta trebuie să fie aritmetică

4.5 - Înmulțirea în Complement de doi bazată pe procedura lui Booth (contin.)

multiplier 4

declare register $A[7 : 0]$, $Q[7 : -1]$, $M[7 : 0]$, $COUNT[2 : 0]$;

declare bus $INBUS[7 : 0]$, $OUTBUS[7 : 0]$;

BEGIN : $A := 0$, $COUNT := 0$, }
INPUT : $M := INBUS$; } ←-----{ C_0 }

$Q[7 : 0] := INBUS[7 : 0]$, $Q[-1] := 0$; ←-----{ C_1 }

TEST1 : if $Q[0]Q[-1] = 01$ then $A := A + M$, go to *TEST2*; ←-----{ C_2 }

if $Q[0]Q[-1] = 10$ then $A := A - M$; ←-----{ C_2, C_3 }

TEST2 : if $COUNT7 = 1$ then go to *OUTPUT*,

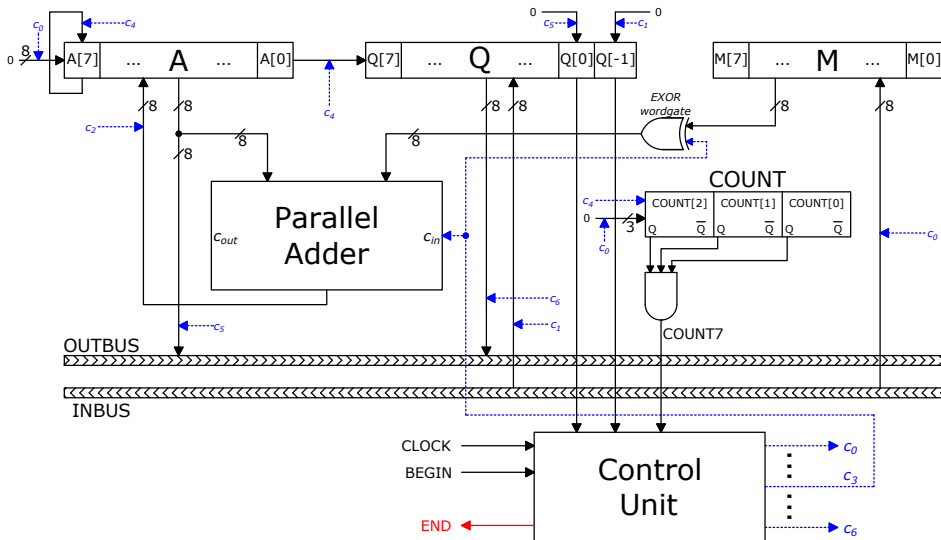
RSHIFT : $A[7] := A[7]$, $A[6 : 0].Q := A.Q[7 : 0]$, }
INCR. : $COUNT := COUNT + 1$, go to *TEST1*; } ←-----{ C_4 }

OUTPUT : $OUTBUS := A$, $Q[0] := 0$; ←-----{ C_5 }

$OUTBUS[7 : 0] := Q[7 : 0]$; ←-----{ C_6 }

END : -----→ { *END* }

4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Booth (contin.)



4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Booth (contin.)

Condiția de overflow:

- ▶ datorită recodificării Booth, nu se pot genera două cifre cu semn consecutive avînd ambele valoarea 1 sau $\bar{1}$
 - ▶ \Rightarrow nu pot exista două iterații consecutive în care să se efectueze aceeași operație aritmetică (adunare sau scădere)
 - ▶ overflow-ul nu poate apărea

Spre deosebire de metoda lui Robertson, nu este necesară corecție finală (X recodificat Booth nu are bit de semn).

O operație aritmetică finală (adunare sau scădere) poate fi necesară după ce COUNT7 a devenit 1, pentru că MSB-ul înmulțitorului recodificat Booth nu mai este un bit de semn ci o cifră cu semn, parte a magnitudinii lui X . Odată ce COUNT7 a devenit 1, nu se mai fac deplasări la dreapta ale produsului parțial.

4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Booth (contin.)

Se consideră operanzii pe 4 biți, în C2, fractionari:

$$\begin{aligned}X &= -0.375 = -3 \cdot 2^{-3} = 1.011_{S.-M.} = 1.101_{C2} \\Y &= -0.875 = -7 \cdot 2^{-3} = 1.111_{S.-M.} = 1.001_{C2}\end{aligned}$$

Produsul P ai celor 2 operanzi, reprezentat în C2, pe 8 biți:

$$\begin{aligned}P &= X \cdot Y = (-3 \cdot 2^{-3}) \cdot (-7 \cdot 2^{-3}) = 21 \cdot 2^{-6} \\&= 0.0101010_{S.-M.} = 0.0101010_{C2}\end{aligned}$$

4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Booth (contin.)

A	Q	M	COUNT	A.C.S.
0 0 0 0		1 0 0 1	0 0	c_0
- 1 0 0 1	1 1 0 1 0			c_1
0 1 1 1				c_2, c_3
0 0 1 1	1 1 1 0 1		0 1	c_4
+ 1 0 0 1				
1 1 0 0				c_2
1 1 1 0	0 1 1 1 0		1 0	c_4
- 1 0 0 1				
0 1 0 1				c_2, c_3
0 0 1 0	1 0 1 1 1		1 1	c_4
			$*COUNT3 == 1$	
0 0 1 0				c_5
	1 0 1 0			c_6
				END

4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Booth (contin.)

Se consideră operanzii pe 4 biți, în C2, întregi:

$$X = -4 = 1100.C_2$$

$$Y = 7 = 0111.C_2$$

Produsul P ai celor 2 operanzi, este obținut astfel:

$$\begin{aligned} P &= X \cdot Y = (-4) \cdot 7 = -28 \\ &= 11100100.C_2 \end{aligned}$$

4.4 - Înmulțirea în Complement de doi bazată pe procedura lui Booth (contin.)

A	Q	M	COUNT
0 0 0 0		0 1 1 1	0 0
0 0 0 0	1 1 0 0 0		
0 0 0 0	0 1 1 0 0		0 1
0 1 1 1	0 0 1 1 0		1 0
1 0 0 1			
1 1 0 0	1 0 0 1 1		1 1
1 1 1 0	0 1 0 0 1		*COUNT3 == 1
1 1 1 0	0 1 0 0		