半加法器(half adder)

1.決定輸入變數與輸出變數的數目

輸入變數：加數與被加數，x代表加數以及y代表被加數 。

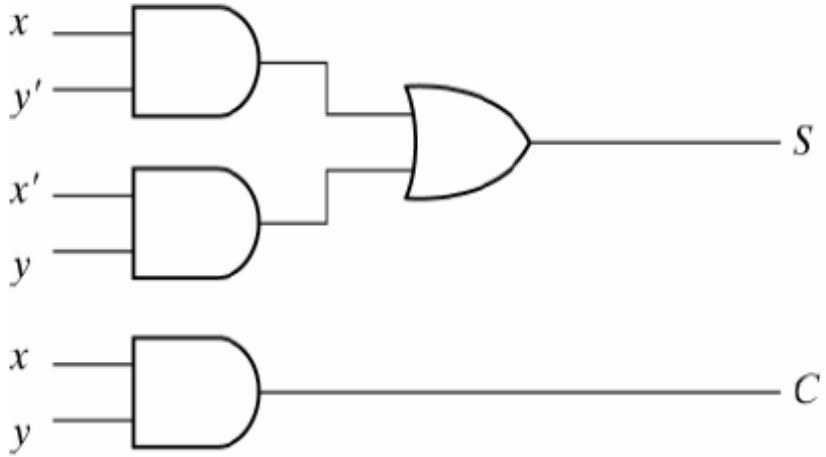
輸出變數：和與進位，S表示和以及C表示進位。

半加法器(half adder)

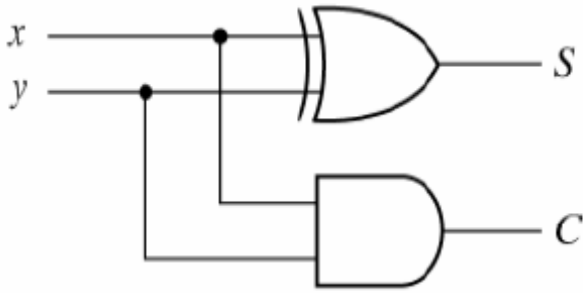
輸入變數：加數與被加數，*x*代表加數以及*y*代表被加數 。
輸出變數：和與進位，*S*表示和以及*C*表示進位。

x	*y*	*S*	*C*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

半加法器(half adder)



(a) $S = xy' + x'y$
$C = xy$



(b) $S = x \oplus y$
$C = xy$

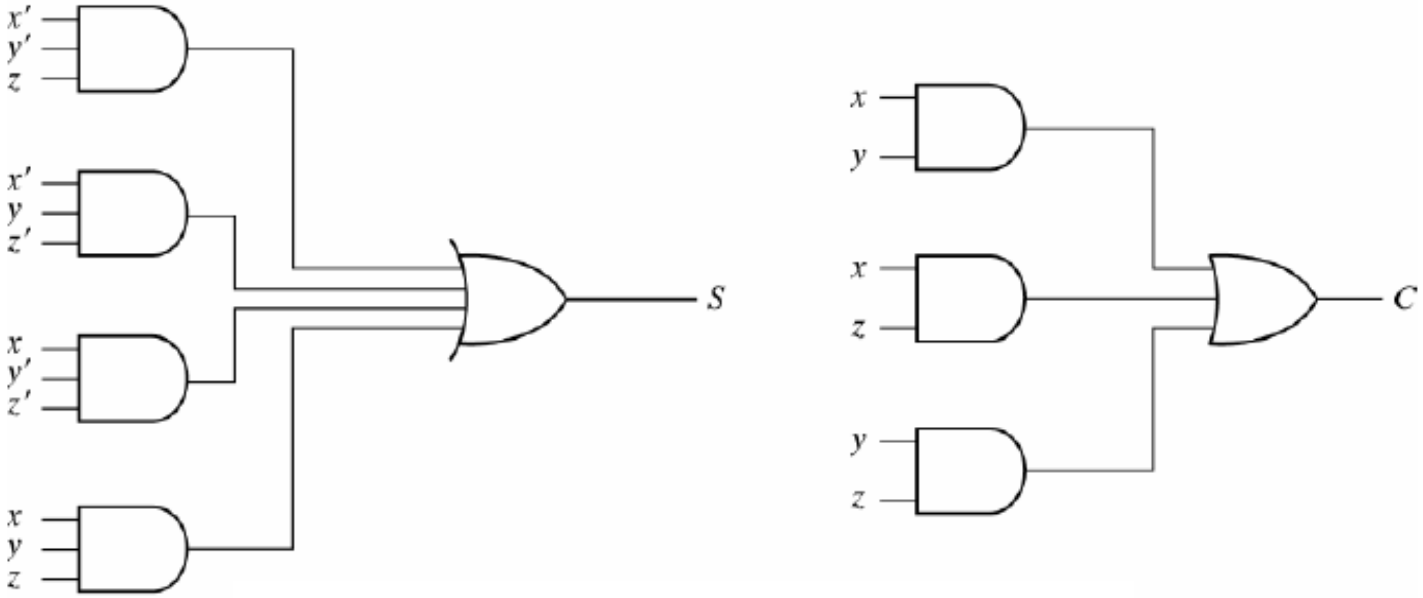
全加法器(full adder)

x	y	z	S	C
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

全加法器(full adder)

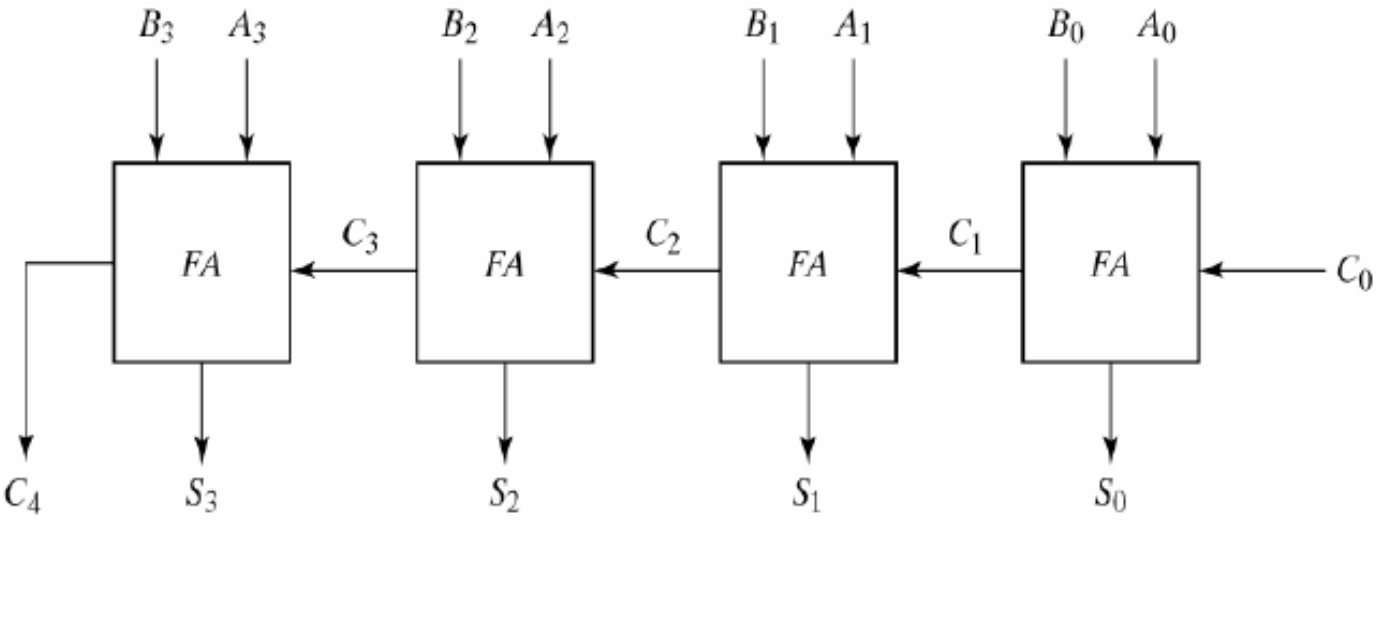
$$S = x'y'z + x'yz' + xy'z' + xyz$$

$$C = xy + xz + yz.$$



二進位加法器(binary adder)

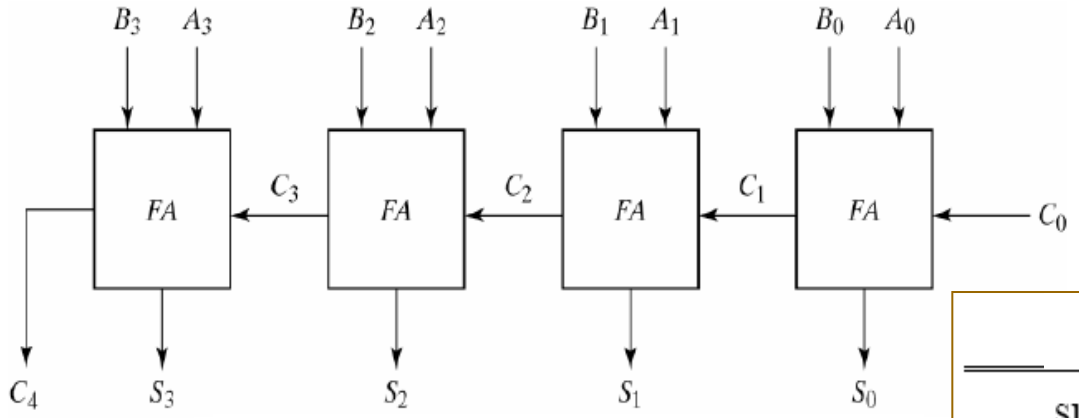
4位元的二進位漣波進位加法器



A是被加數，B是加數，它們的下標代表了位元的權重，例如被加數$A = A_3A_2A_1A_0$，A_0、A_1、A_2和A_3分別代表十進位數值的1、2、4和8，C_0必須是0，輸出S代表A和B相加的總和。

二進位加法器(binary adder)

4位元的二進位漣波進位加法器的操作範例：



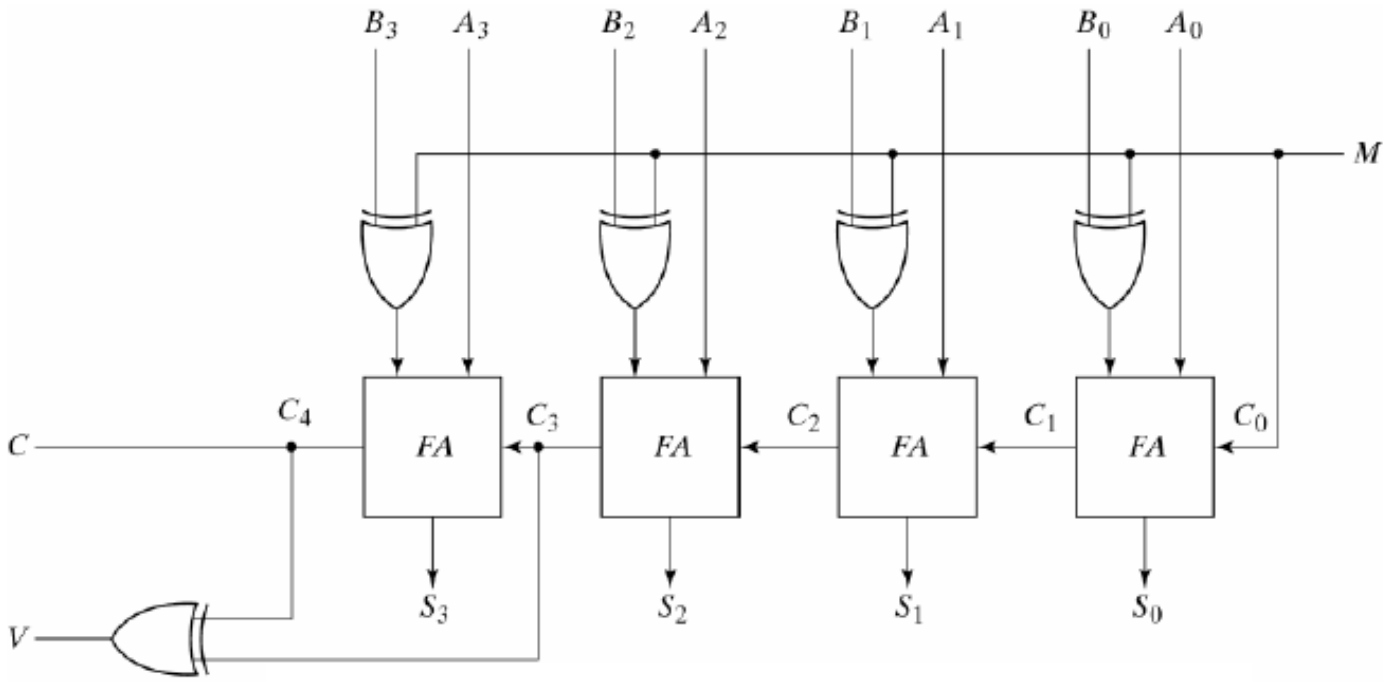
A=1011，B=0011

suffix i:	3	2	1	0	
input carry	0	1	1	0	C_i
augend	1	0	1	1	A_i
addend	0	0	1	1	B_i
sum	1	1	1	0	S_i
output carry	0	0	1	1	C_{i+1}

二進位減法運算

- 減法運算方式：A-B=A+(-B)=A+(B之2的補數)。
- 電腦中儲存負值通常採用2的補數之方式儲存。
- 1的補數:先取此負值的絕對值之二進位的表示後，再對此值做反向 (即1變0，0變1)。
- 2的補數: 1的補數+1。

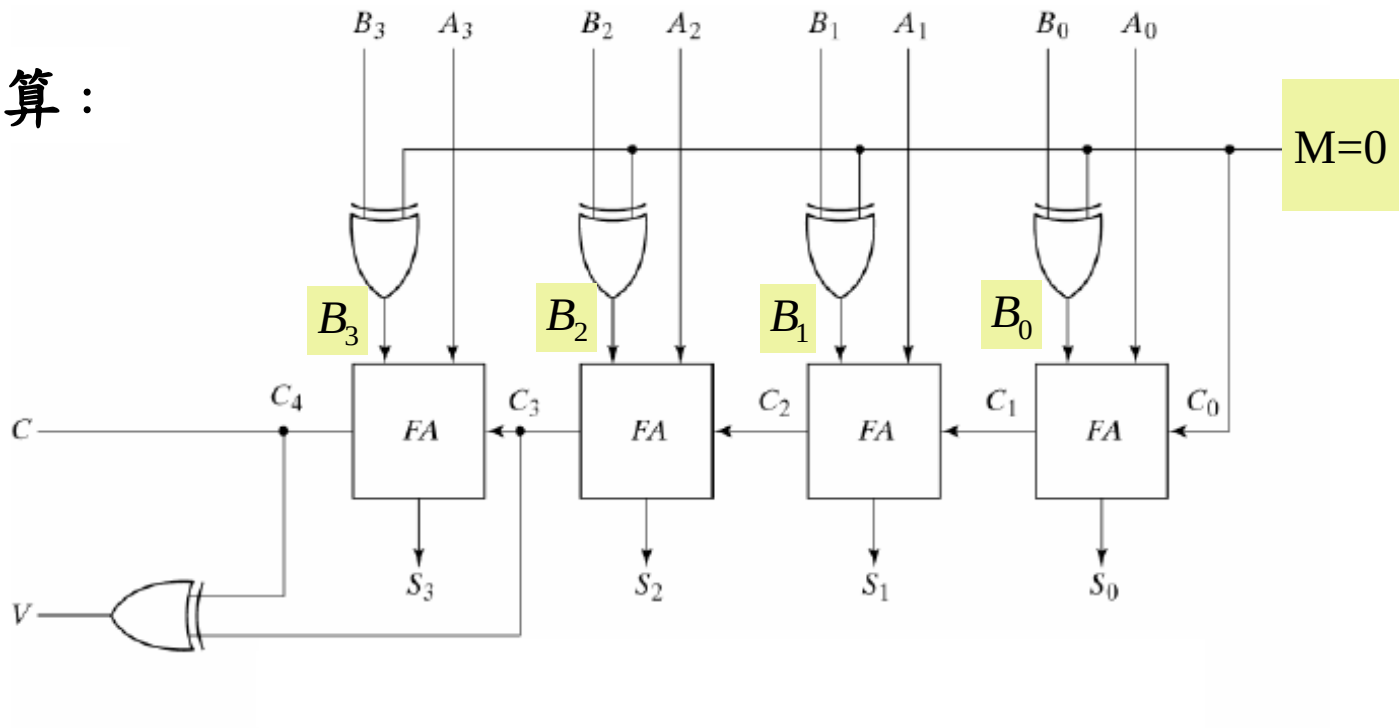
二進位加法器/減法器 (binary adder/subtractor)



M=0時，電路執行加法運算；M=1時，電路執行減法運算。

二進位加法器/減法器 (binary adder/subtractor)

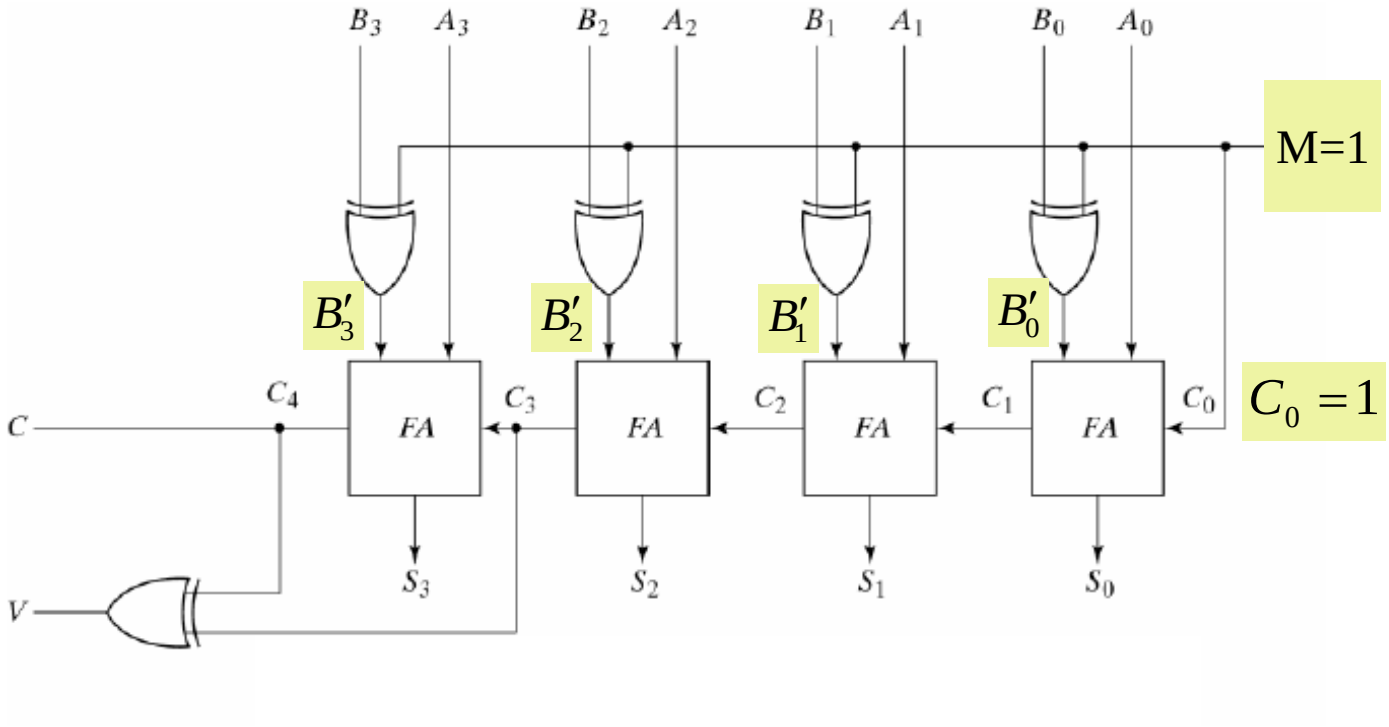
加法運算：



M=0時，$B_i \oplus 0 = B_i$以及$C_0 = 0$，此時每一個XOR閘的輸出為B_i，因此每一個全加法器執行$A_i + B_i$的運算，所以電路執行加法運算。

二進位加法器/減法器 (binary adder/subtractor)

減法運算：

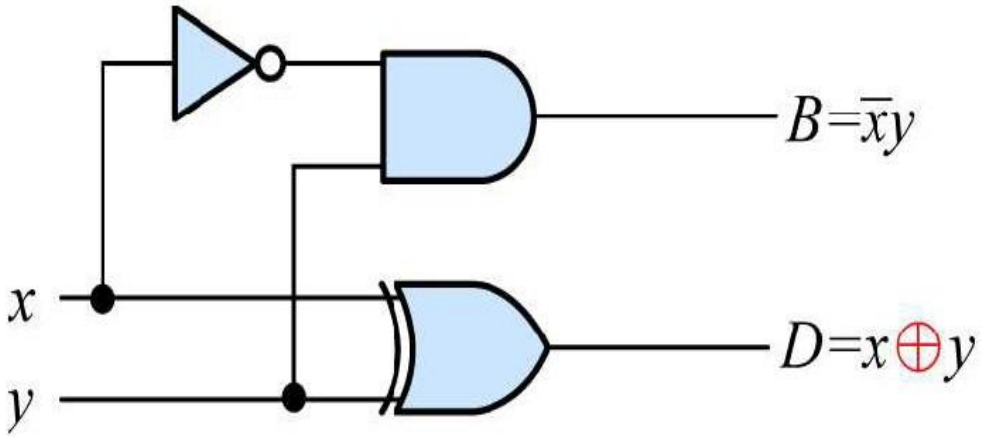


檢查是否有溢位的產生。

M=1時，$B \oplus 1 = B'$ 以及 $C_0 = 1$，此時4個XOR閘的輸出為B的1的補數，再把 $C_0 = 1$ 加入B的1的補數得B之2的補數，因此4個全加法器執行A+(B之2的補數)的運算，所以電路執行減法運算。

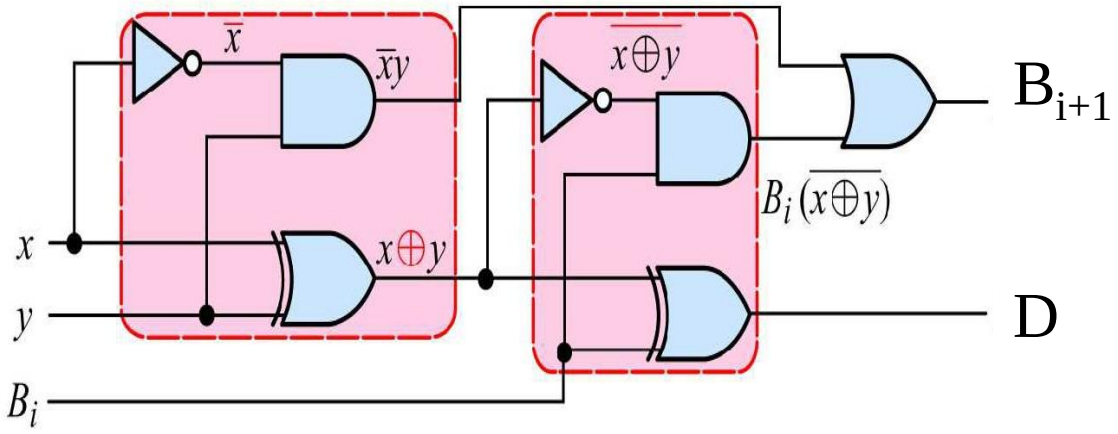
半减法器(half subtractor)

x	y	差 (D)	借位 (B)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

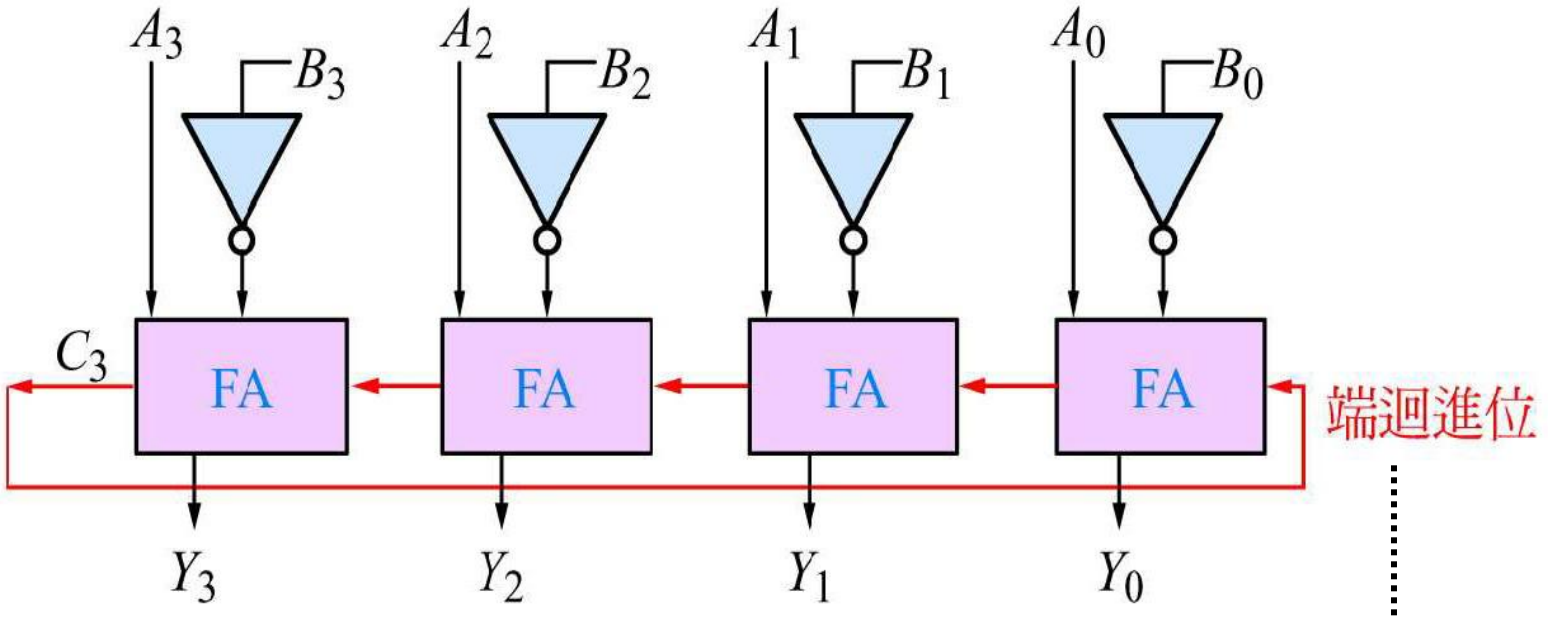


全减法器(full subtractor)

x	y	B_i	D	B_{i+1}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



1’s補數減法器

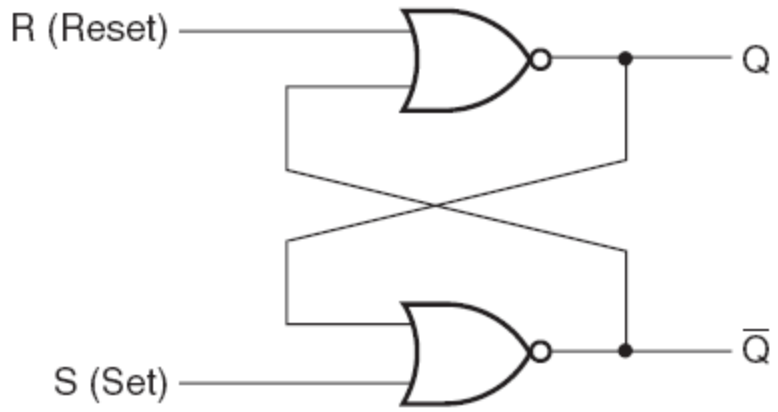


並列式4位元1’s補數減法器

執行2’s補數減1的動作

SR Latch（1/4）

- 由二個NOR構成的正緣觸發SR正反器
- S=1 ,Q設爲1 ； R=1 ,Q設爲0



(a) Logic diagram

S	R	Q	$\overline{Q}$	
1	0	1	0	Set state
0	0	1	0	
0	1	0	1	Reset state
0	0	0	1	
1	1	0	0	Undefined

(b) Function table

SR Latch（2/4）

- 正緣觸發的時序圖

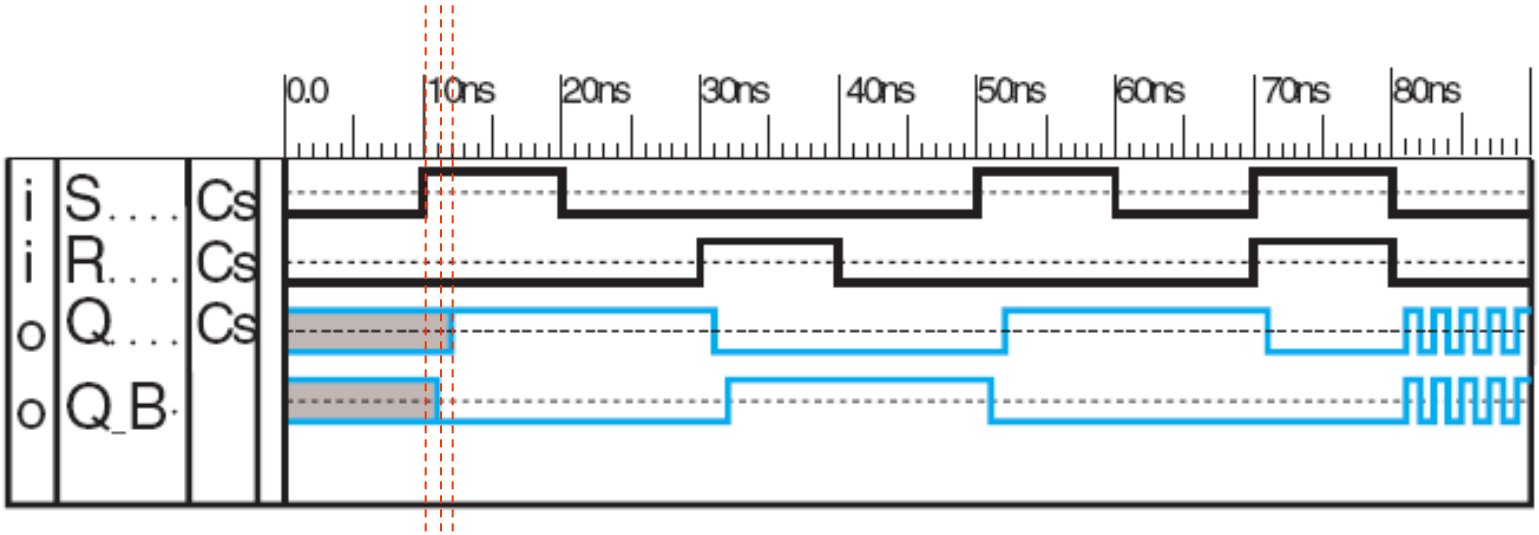
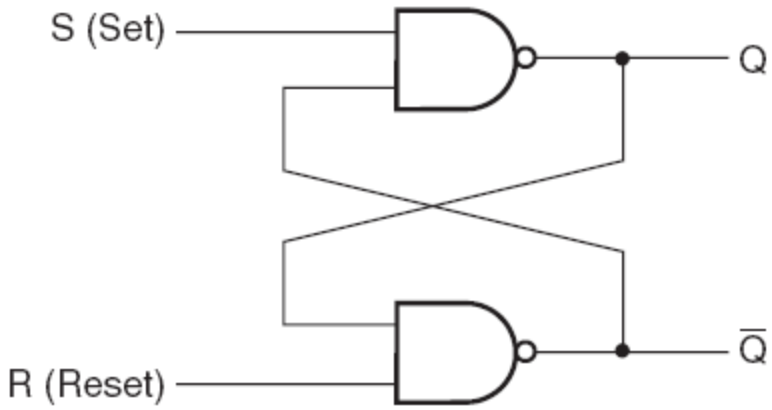


Fig. 4-5 Logic Simulation of *SR* Latch Behavior

SR Latch（3/4）

- 下圖的SR正反器由二個NAND構成，具有回饋電路，不同前者的是，以負緣觸發



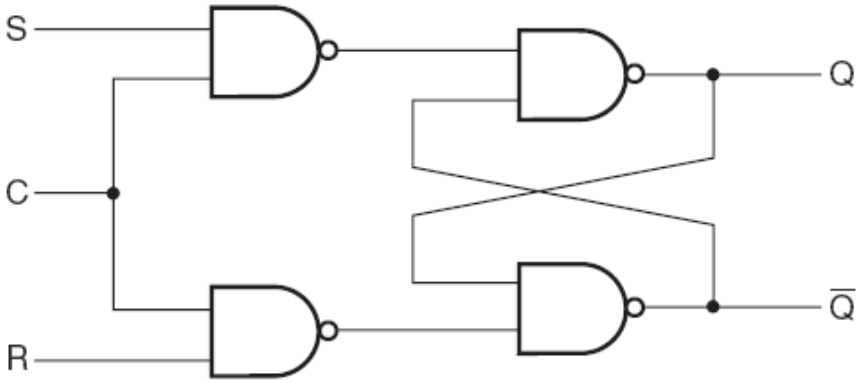
(a) Logic diagram

S	R	Q	$\overline{Q}$	
0	1	1	0	Set state
1	1	1	0	
1	0	0	1	Reset state
1	1	0	1	
0	0	1	1	Undefined

(b) Function table

SR Latch（4/4）

- 具有控制訊號的SR正反器

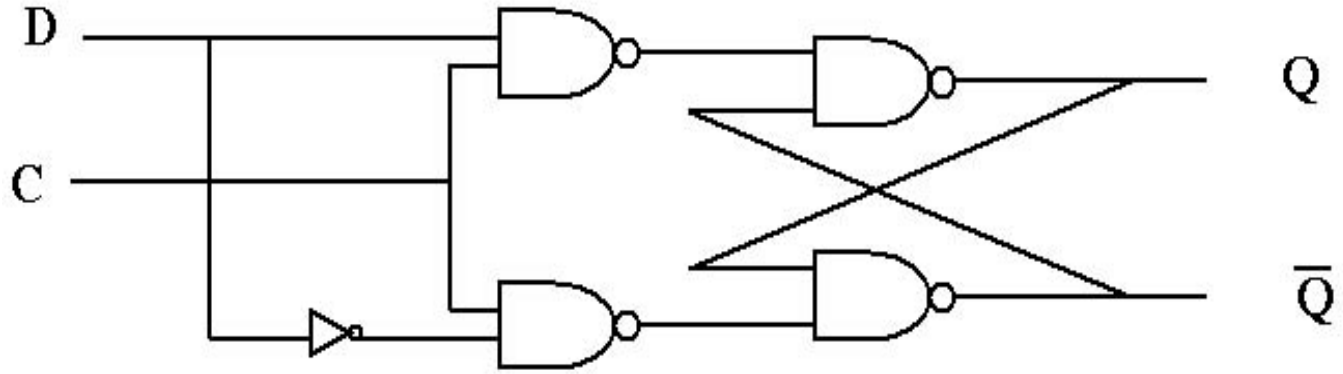


(a) Logic diagram

C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

(b) Function table

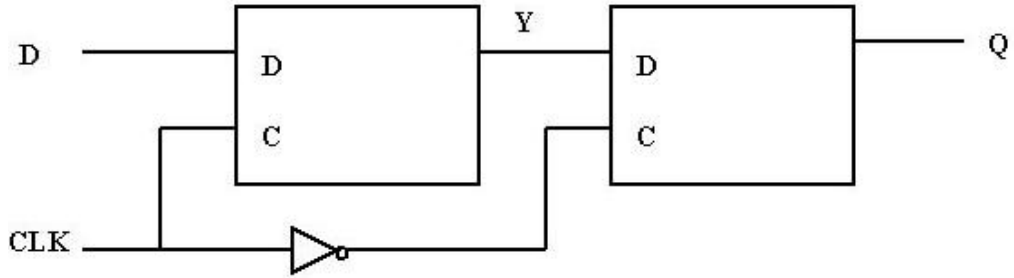
D Latch（1/2）

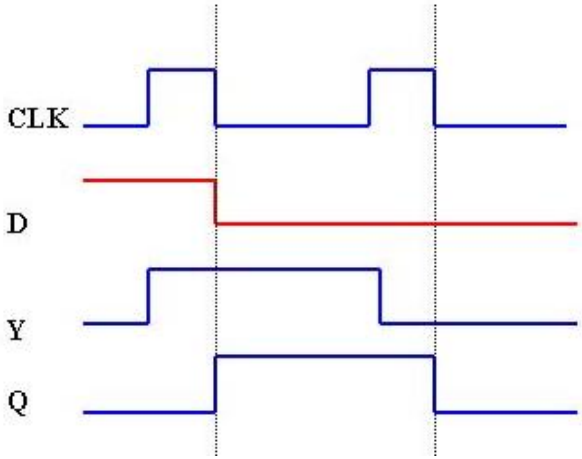


C	D	Q
0	x	No Change
1	0	0 (reset)
1	1	1 (set)

D Latch（2/2）

- 負緣觸發D正反器的時序圖





移位相加乘法運算

$$P_{i+1}=P_i + X_i \times 2^iY$$

2^iY為執行左移i個bit位置

	1 0 1 0	Y
×	1 1 0 1	X
0 0 0 0 0 0 0 0		$P_0=0$
+ 1 0 1 0		
0 0 0 0 1 0 1 0		$P_1=P_0+X_0Y$
+ 0 0 0 0		
0 0 0 0 1 0 1 0		$P_2=P_1+2X_1Y$
+ 1 0 1 0		
0 0 1 1 0 0 1 0		$P_3=P_2+2^2X_2Y$
+ 1 0 1 0		
1 0 0 0 0 0 1 0		$P_4=P_3+2^3X_3Y$

移位相加乘法運算

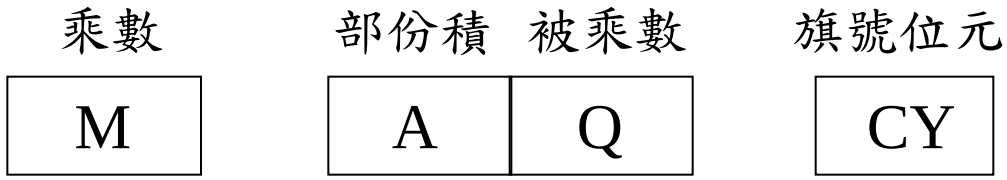
```
          1 0 1 0   Y
×         1 1 0 1   X
-------------------
  0 0 0 0 0 0 0 0
+         1 0 1 0
-------------------
    0 0 0 0 1 0 1 0
+         0 0 0 0
-------------------
      0 0 0 0 1 0 1 0
+         1 0 1 0
-------------------
        0 0 1 1 0 0 1 0
+         1 0 1 0
-------------------
        1 0 0 0 0 0 1 0
```

或可執行Y位置不變
右移P_i 1個bit位置

$$P_{i+1}=P_i + X_i \times 2^i Y$$

$$P_{i+1}=P_i + 2^i X_i \times Y$$

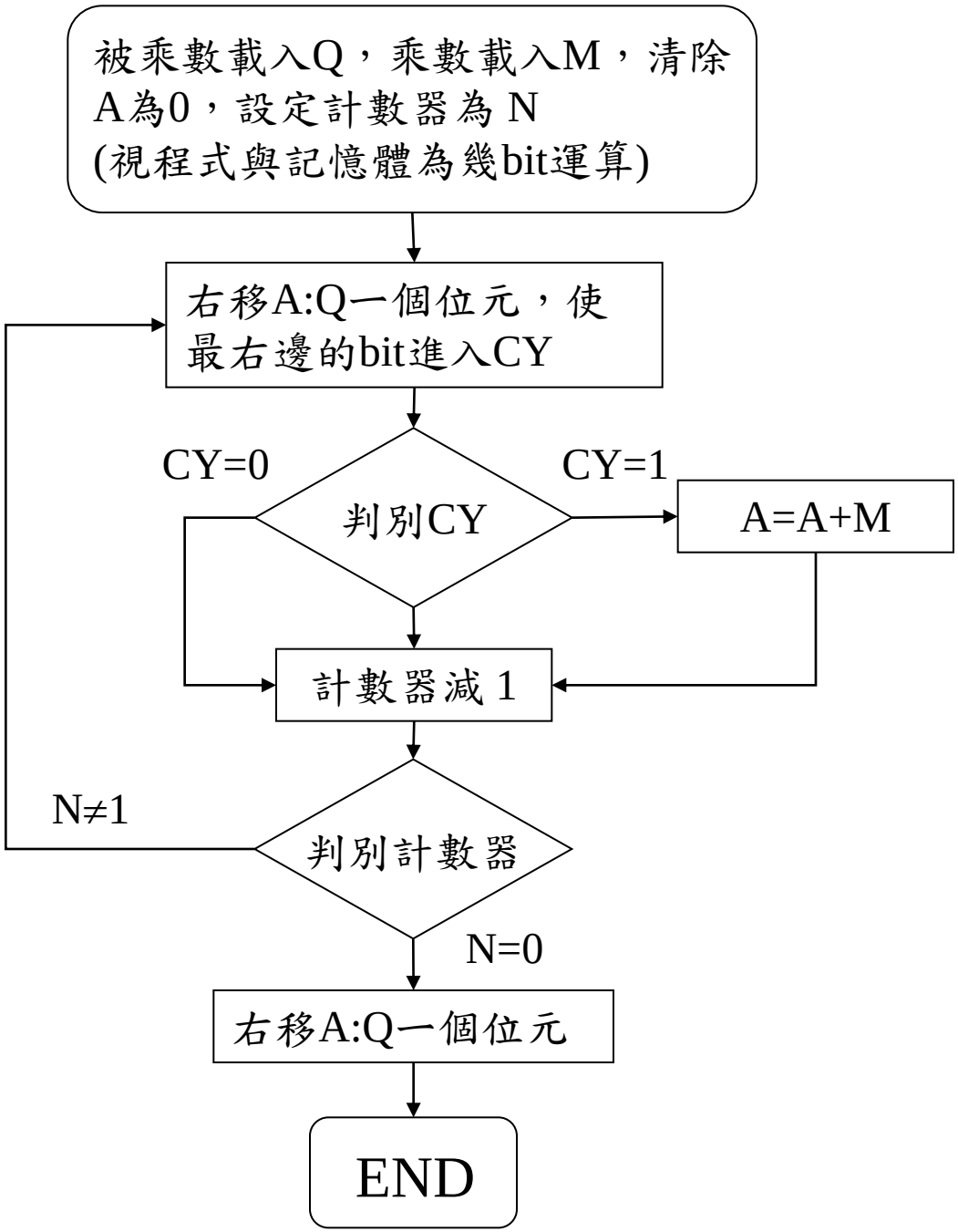
移位相加乘法運算



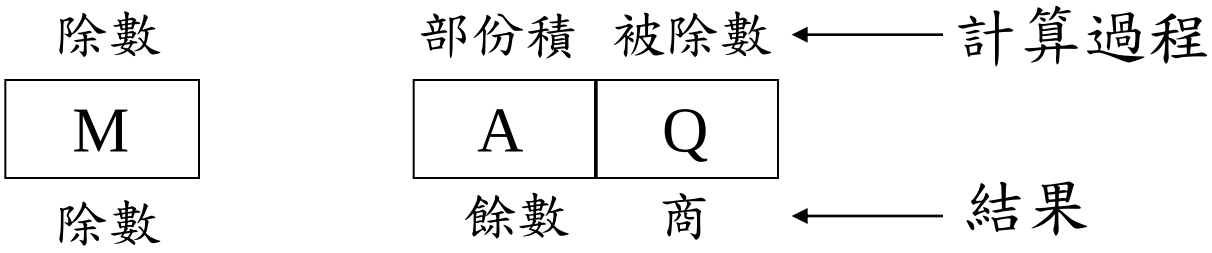
乘法運算暫存器配置

a. Logical right shift

b. Logical left shift


被乘數載入Q，乘數載入M，清除A為0，設定計數器為 N
(視程式與記憶體為幾bit運算)
右移A:Q一個位元，使最右邊的bit進入CY
CY=0
CY=1
判別CY
A=A+M
計數器減 1
N≠1
判別計數器
N=0
右移A:Q一個位元
END

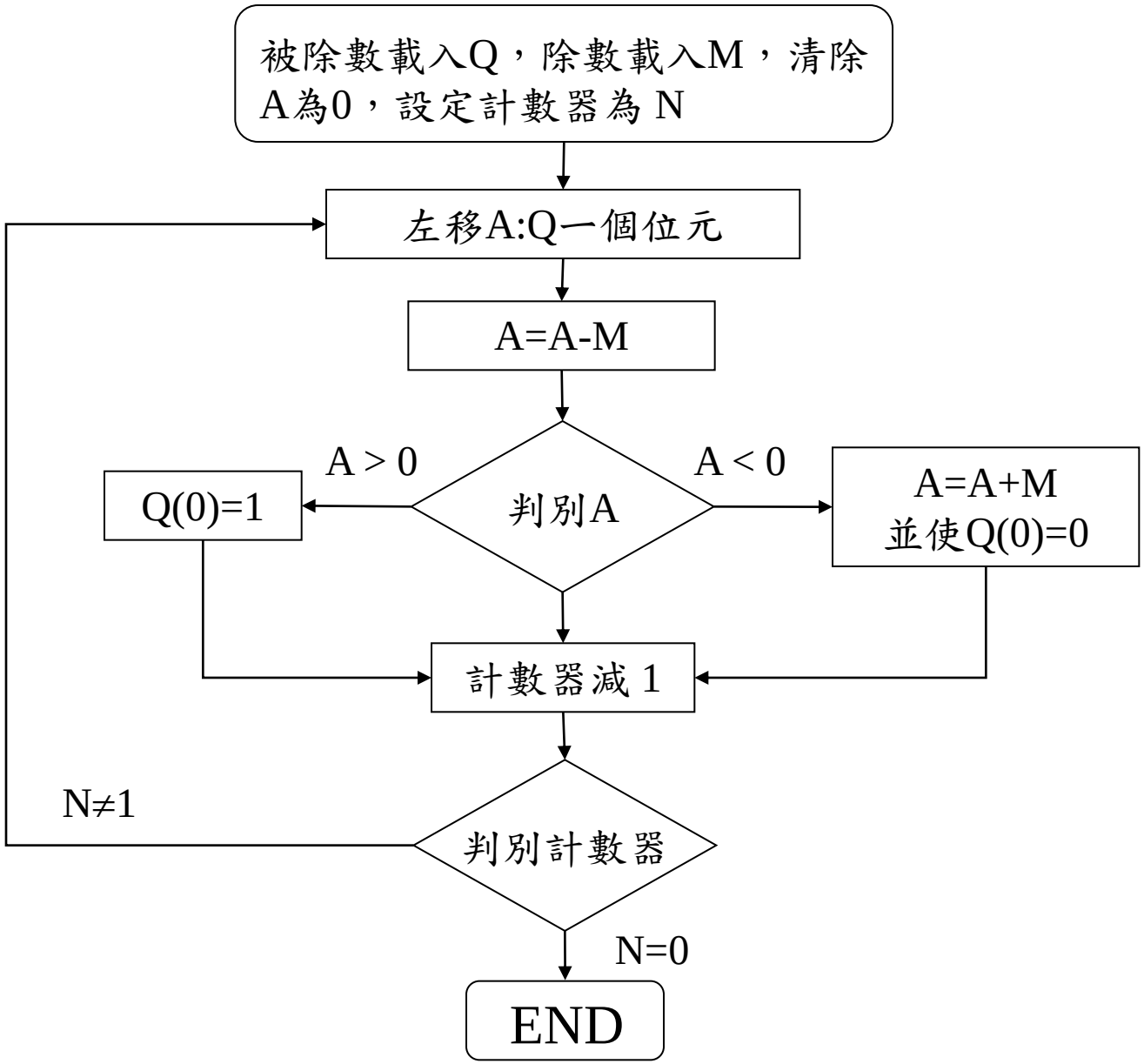
移位除法運算



除法運算暫存器配置

1.恢復式除法 (restoring division)
2.非恢復式除法 (nonrestoring division)

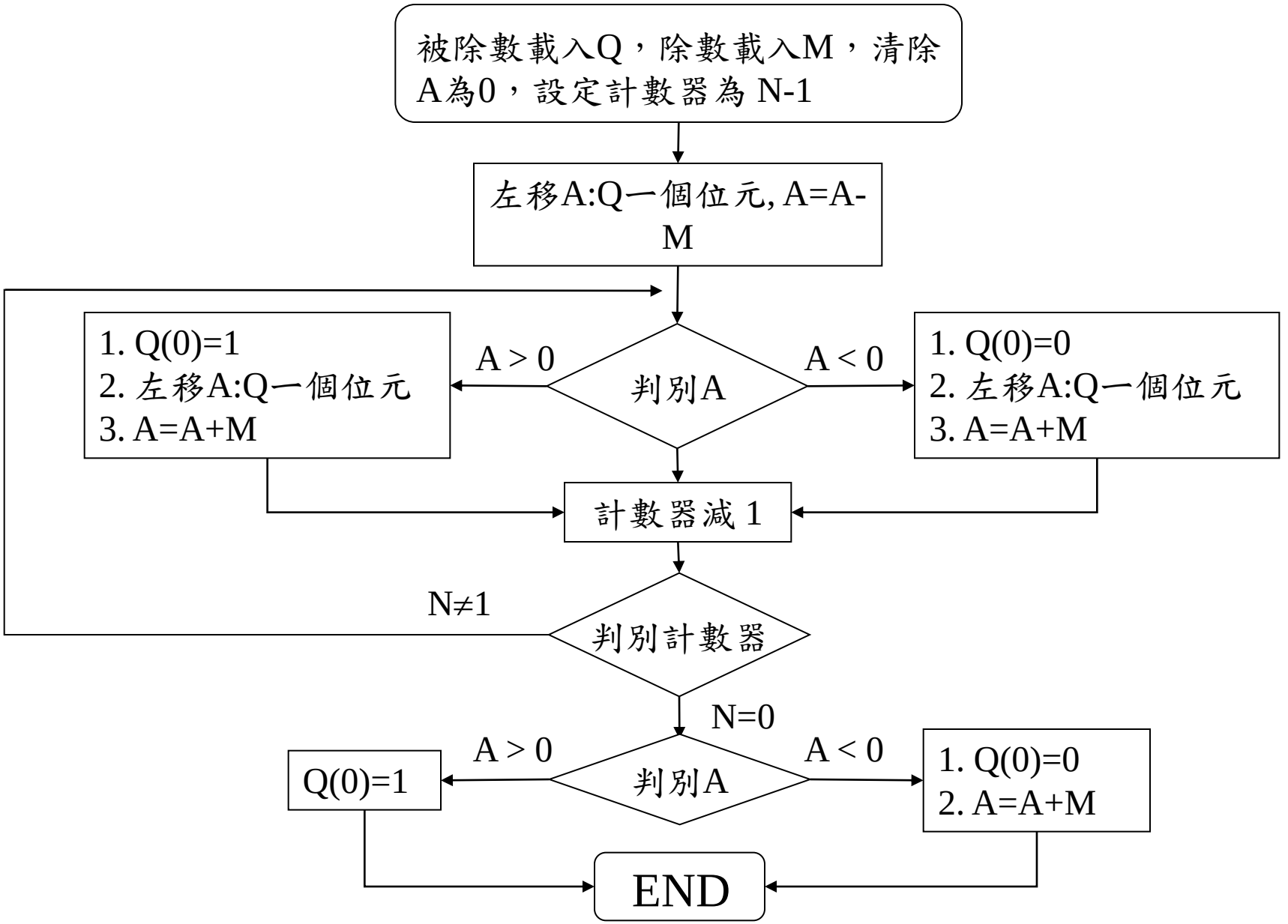
恢復式除法 (restoring division)



除數 (M)	A	被除數 (Q)	N	
00000101	00000000	00100110	8	
	00000000	01001100		左移A:Q
	- 00000101			
	11111011			A ← A-M
				A<0，恢復除數
	+ 00000101			A ← A+M
	00000000	01001100	7	
	00000000	10011000		左移A:Q
	- 00000101			
	11111011			A ← A-M
				A<0，恢復除數
	+ 00000101			A ← A+M
	00000000	10011000	6	
	00000001	00110000		左移A:Q
	- 00000101			
	11111100			A ← A-M
				A<0，恢復除數
	+ 00000101			A ← A+M
	00000001	00110000	5	
	00000010	01100000		左移A:Q
	- 00000101			
	11111101			A ← A-M

		N	
11111101			A<0，恢復除數
+ 00000101			A ← A+M
00000010	01100000	4	左移A:Q
00000100	11000000		A ← A-M
- 00000101			A<0，恢復除數
11111111			A ← A+M
+ 00000101			
00000100	11000000	3	左移A:Q
00001001	10000000		A ← A-M
- 00000101			A>0
00000100	10000001	2	左移A:Q
00001001	00000010		A ← A-M
- 00000101			A>0
00000100	00000011	1	左移A:Q
00001000	00000110		A ← A-M
- 00000101			A>0
00000011	00000111	0	
餘數	商		

非恢復式除法 (nonrestoring division)



除數 (M)	A	被除數 (Q)	N	
00000101	00000000	00100110	8	
	00000000	01001100		左移A:Q
	- 00000101			A ← A-M
	11111011	01001100	7	A<0
	11110110	10011000		左移A:Q
	+ 00000101			A ← A+M
	11111011	10011000	6	A<0
	11110111	00110000		左移A:Q
	+ 00000101			A ← A+M
	11111100	00110000	5	A<0
	11111000	01100000		左移A:Q
	+ 00000101			A ← A+M
	11111101	01100000	4	A<0
	11111010	11000000		左移A:Q
	+ 00000101			A ← A+M
	11111111	11000000	3	A<0
	11111111	10000000		左移A:Q
	+ 00000101			A ← A+M
	00000100	10000001	2	A>0

	00000100	10000001	2	A>0
	00001001	00000010		左移A:Q
-	00000101			A ← A-M
	00000100	00000011	1	A>0
	00001000	00000110		左移A:Q
-	00000101			A ← A-M
	00000011	00000111	0	A>0
	餘數	商		