Booth's Algorithm - efficient way

Multiplication of two signed binary nos. - represented in 2's complement

~~7~~ X 3

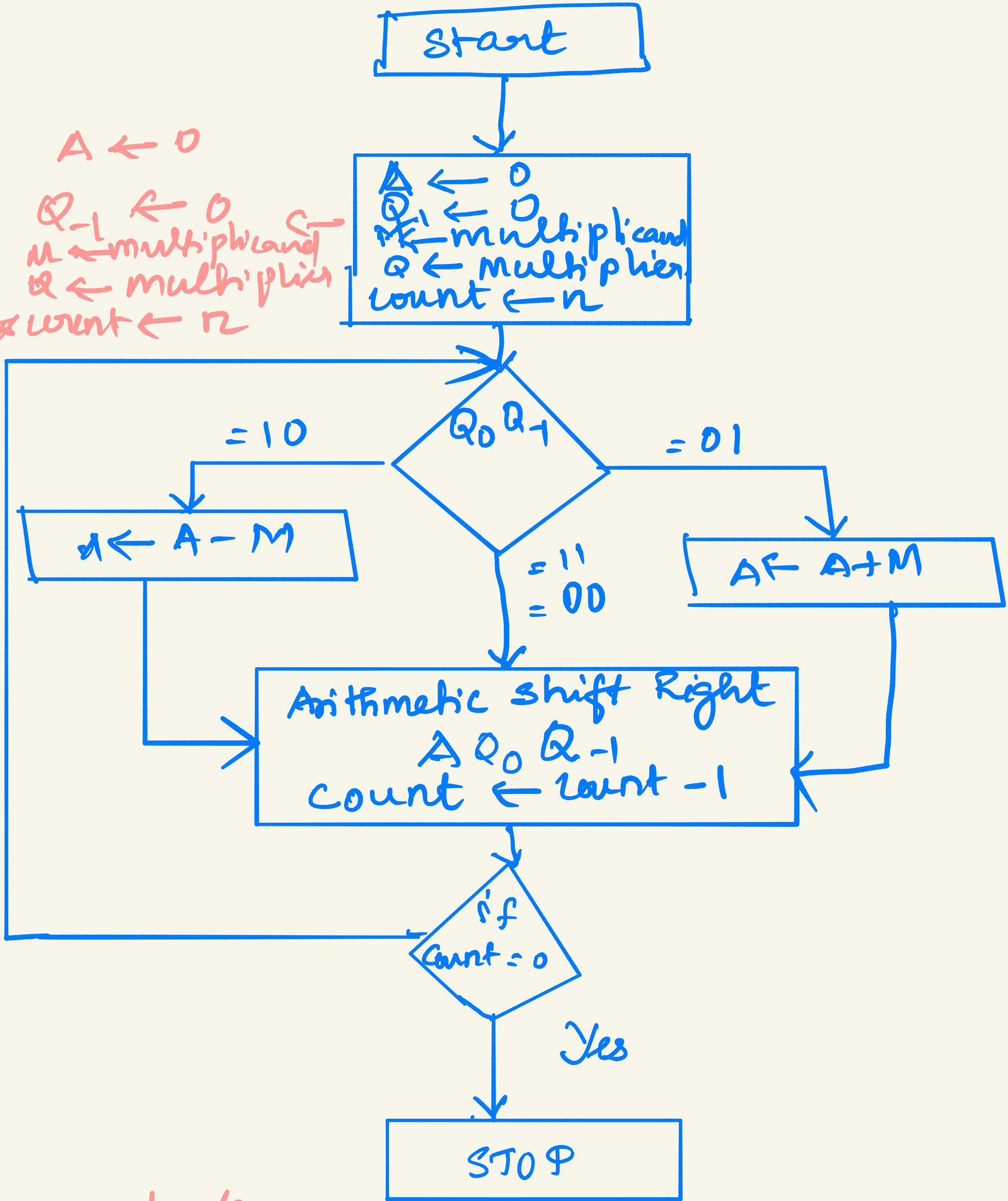
7 ⇒ multiplicand

3 ⇒ multiplier.

→ less number of additions and subtractions.

→ Fact : multiplier has strings of 0's - no addition just shifting

→ strings of 1's in the multiplier from bit weight 2^k to 2^m can be treated as 2^{k+1} to 2^m.



* n = length of multiplicand

7 - Multiplicand - M   Eg: 7×3

3 - Multiplier - Q

7 - 0111 ⇒ M

3 - 0011

-M = 2's comp (M)

1's = 1 0 0 0

+ 1

-M = 1 0 0 1

Operations n = 4

Count = 4

A	Q	Q_{-1}	
0000	0011	0	
1001			
1001	0011	0	A ← A-M A ← A+2's(M)
1100	1001	1	ARS (AQ Q_{-1}) count=3
1110	0100	1	ARS (AQ Q_{-1}) count: 2
0111			A ← A+M
1 0101	0100	1	
0010	1010	0	ARS (AQ Q_{-1}) count=1
0001	0101	0	ARS (AQ Q_{-1}) count=0

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

0 + 0 + 0 + 16 + 0 + 4 + 0 + 1 = 21

Eg2: 2×-5

$2 \Rightarrow$ multiplicand (M) = $\boxed{00010}$

$-5 \Rightarrow$ multiplier (Q) =

$-M = \begin{array}{r} 11101 \\ +\;1 \\ \hline 11110 \\ \hline \end{array}$

$5 = 0101$

$-5 = \begin{array}{r} 1010 \\ +\;1 \\ \hline 1011 \end{array}$

A	Q	Q_{-1}
00000	11011	0

Logical operations.

Logical shift left operations

0001
0010

Multiplication

LSB - filled with 0
MSB - discarded

Logical shift Right operations

1000

MSB - filled with 0
LSB - discarded

Division

1000
0100

Arithmetic Shift Left:

identical to logical shift left

MSB - discarded

LSB - filled with 0

Arithmetic Shift Right

LSB - discarded

MSB - is filled with previous value.

1011

1 101

Representation of +ve & -ve nos.

1. Unsigned representation
2. Sign & Magnitude representation

Eg: +9 using 7 bit

$\boxed{0}$ 001001

→ sign bit

-9 using 7 bit

1 001001

→ sign bit is set for -ve nos

3. Signed 1's complement representation
4. Signed 2's complement representation

Fixed point Representation

32 bit Representation

sign bit	15 bit	16 bit
1	000000001 011011	0010000000000000

$-(1011011.001)_2$

↳ radix point

Eg: $(0111.1100)_2$ to decimal

Modified Booth's algorithm.

M - multiplicand

Q - Multiplier

Eg: 13×-6

13 - Multiplicand $\Rightarrow$ 01101

+6 = 0110

-6 = 1010

1001
+ 1
1010

Note: Even no. of bits in A, M & Q

+13 = 001101

-6 $\Rightarrow$ 111010

Flag Registers

CF - Carry Flag - Carry out of MSB

ZF - Zero Flag - O/P prev. inst. Zero

OF - Overflow Flag - result of signed op. is large

SF - Sign Flag - result of comp. - sign

PF - Parity Flag - lower byte contains even no. of 1's

DF - Direction Flag - Direction of string to be proc.

TF - Trap Flag - Trapping execution

IF - Interrupt Flag - maskable Interrupt.

A	Q	Q_{-1}	Operations
000000	1110$\boxed{10}$	$\boxed{0}$	Count = 3
100110			$-2 \times M$
100110	111010	0	Arithmetic Shift Right 2 times
111001	1011$\boxed{10}$	$\boxed{1}$	counter = 2
110011			$-1 \times M$
101100			
111011	0010$\boxed{11}$	$\boxed{1}$	ASR(2 times) Counter = 1
111110	110010	1 ×	counter = 0

$\underbrace{111110\ 110010}_{-78}$

$-9 \Leftarrow\Rightarrow$

$$
\begin{array}{r}
001001 \\
110110 \\
+\ 1 \\
\hline
110111
\end{array}
$$

1010
1011
1100
1001

$$
13 \Rightarrow 001101
$$

$$
\begin{array}{rr}
-13 \Rightarrow & 110010 \\
& +\ 1 \\
\hline
& 110011
\end{array}
$$

$0 \times M \Leftrightarrow$ simply shift

$+1 \times M \Leftrightarrow$ Add M & shift

$-1 \times M \Leftrightarrow$ complement M & add to A & shift (2 times)

$2 \times M \Leftrightarrow$ left shift M & Add M to A & 2 times shift

$-2 \times M \Leftrightarrow$ left shift M complement Result Add to A 2 times shift.

Modified Booth's algorithm

Tabulation

Multiplier Bit-pair i+1	i	Multiplier bit on the right i-1	Booths representation		Multiplicand selected at position i
0	0	0	0	0	0 ×M
0	0	1	0	+1	+1× M
0	1	0	+1	−1	+1 ×M
0	1	1	+1	0	+2 ×M
1	0	0	−1	0	−2 ×M
1	0	1	−1	+1	−1 ×M
1	1	0	0	−1	−1 ×M
1	1	1	0	0	0 × M