Multiplication

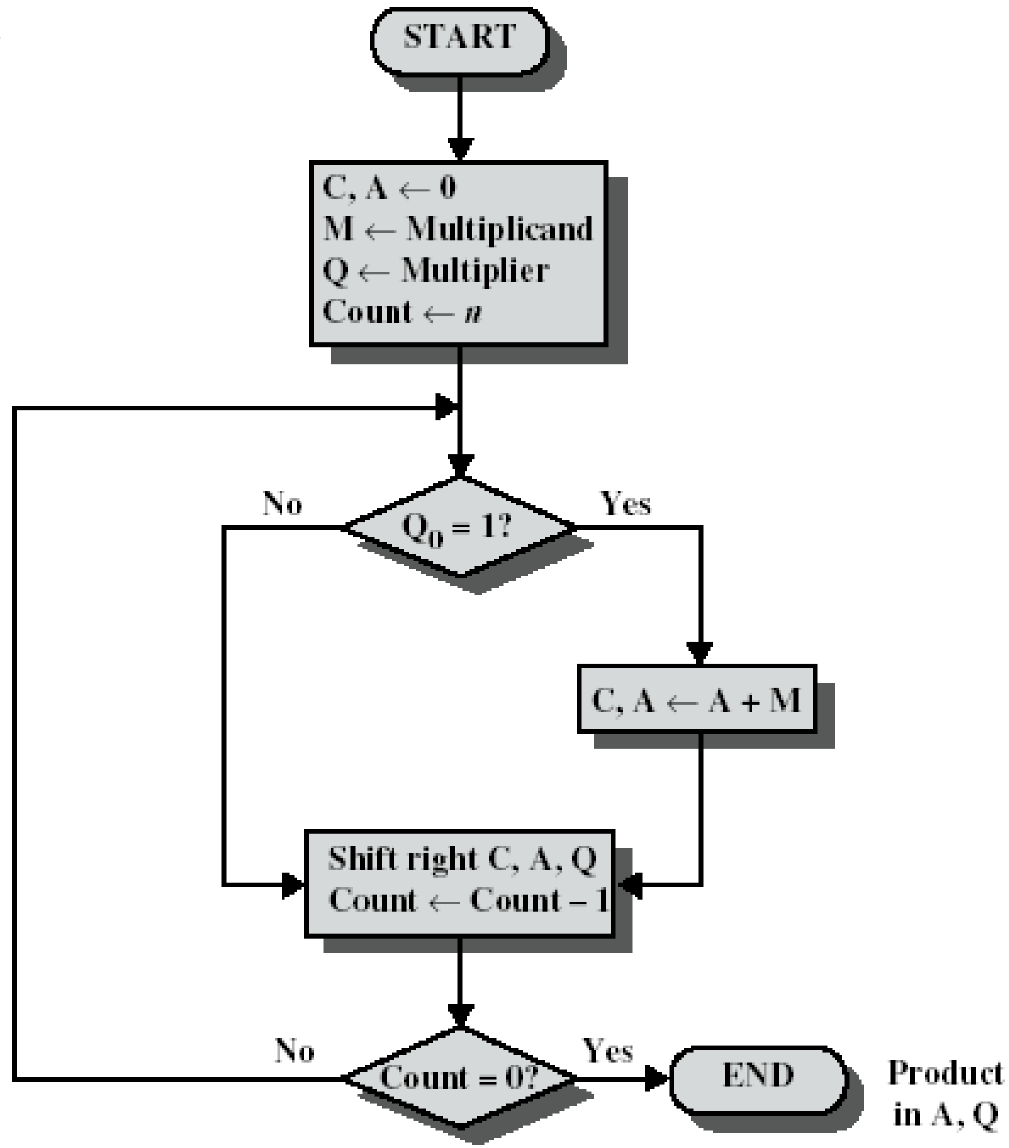
Some general observations

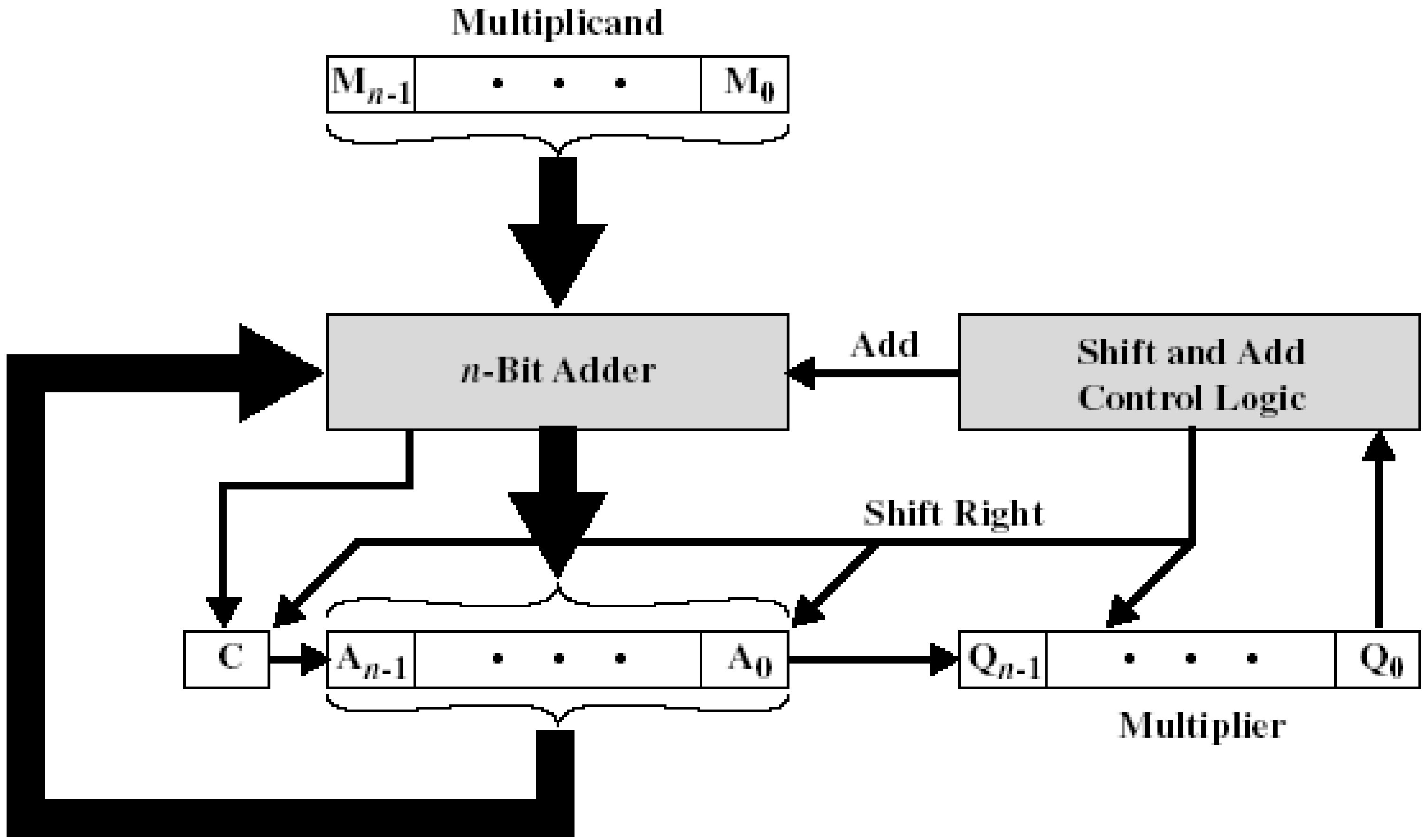
1. Multiplication involves the generation of partial products – one for each digit in the multiplier.
2. Partial products are summed to produce the final product.
3. Partial products are very simple to define for binary multiplication. If the digit is a 'one' the partial product is the multiplicand, otherwise the partial product is zero.
4. The total product is the sum of the partial products. Each successive partial product is shifted one position to the left.
5. The multiplication of two n-bit binary numbers results in a product of up to 2n bits in length.

$$
\begin{array}{r}
1011 \\
\text{X} \quad 1101 \\
\hline
1011 \\
0000 \\
1011 \\
1011 \\
\hline
10001111
\end{array}
$$

Simplifying Multiplication

1. The processor can keep a running product rather than summing at the end.
2. For each ‘1’ in the multiplier we can apply an add and a shift.
3. For each ‘0’ only a shift is needed.





1. Multiplier and multiplicand are loaded into registers Q and M.
2. A third register (A) is initially set to zero.
3. A one-bit C register (initialised to zero) holds carry bits.

C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

This Approach will not work if both or any one of the Multiplicand or Multiplier are negative.

ALTERNATIVE:

--- BOOTH ALGORITHM

- ➔ Multiplicand M unchanged

- ➔ Based upon recoding the multiplier Q to a recoded value R

- ➔ Each digit can assume a negative as well as positive and zero values

- ➔ Known as Signed Digit (SD) encoding

- Booths algorithm called skipping over one’s

- String of 1’s replaced by 0’s

- For ex: 30 = 0011110

= 32 – 2 = 0100000 - 0000010

- In the coded form = $0100 0 \bar{1} 0$

- Booth recoding Procedure

  Working from LSB to MSB retain each 0
  until a 1 is reached

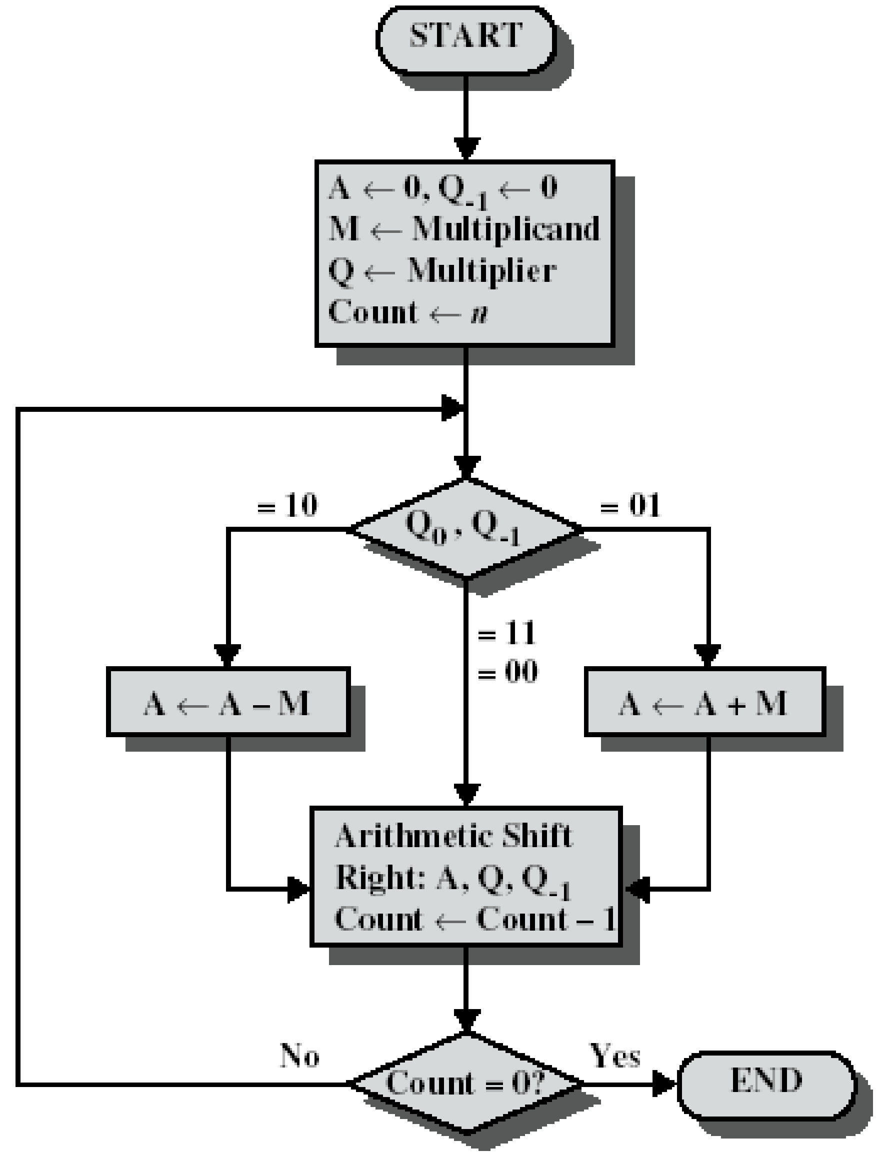
  When a 1 is encountered insert $\bar{1}$ at that position
  and complement all the succeeding 1's until a 0 is encountered

Replace that 0 with 1 and continue

While multiplying with $\bar{1}$ 2's compliment is taken

Booth's Algorithm

1. Multiplier and multiplicand are placed in Q and M registers.
2. A 1-bit register is placed to the right of the least significant bit (Q_0) and designated Q_{-1}.
3. Control logic scans the bits of the multiplier one at a time – but a bit AND its bit to the right are examined. If the bits are the same (1-1 or 0-0) then all bits of the A,Q, and Q-1 registers are shifted to the right 1 bit. If the two bits differ, then the multiplicand is added/subtracted depending on whether the bits are 0-1 or 1-0. Addition is followed by a **right arithmetic shift.**



Performing 7x3

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	$A \leftarrow A - M$	First Cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second Cycle
0101	0100	1	0111	$A \leftarrow A + M$	Third Cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth Cycle

- Booth’s algorithm generally performs fewer additions and subtractions than repeated addition.

Verify the operation of

(+7) x (-3)

Multiplicand M = 0111

Multiplier Q = 1101

A	Q	Q_{-1}	M = 0111
0000	1101	0	initial values
1001	1101	0	A $\leftarrow$ A-M
1100	1110	1	shift
0011	1110	1	A $\leftarrow$ A+M
0001	1111	0	shift
1010	1111	0	A $\leftarrow$ A-M
1101	0111	1	shift
1110	1011	1	shift

Verify the operation of

$(-7) \times (+3)$

Multiplicand M = 1001

Multiplier Q = 0011

A	Q	Q_{-1}	M = 1001
0000	0011	0	initial
0111	0011	0	A ← A-M
0011	1001	1	shift
0001	1100	1	shift
1010	1100	1	A ← A+M
1101	0110	0	shift
1110	1011	0	shift

DIVISION → Dividend/Divisor

- Bits of dividend are examined from left to right until set of bits examined represents a number greater than or equal to the divisor

- Until this 0's are placed in quotient from left to right

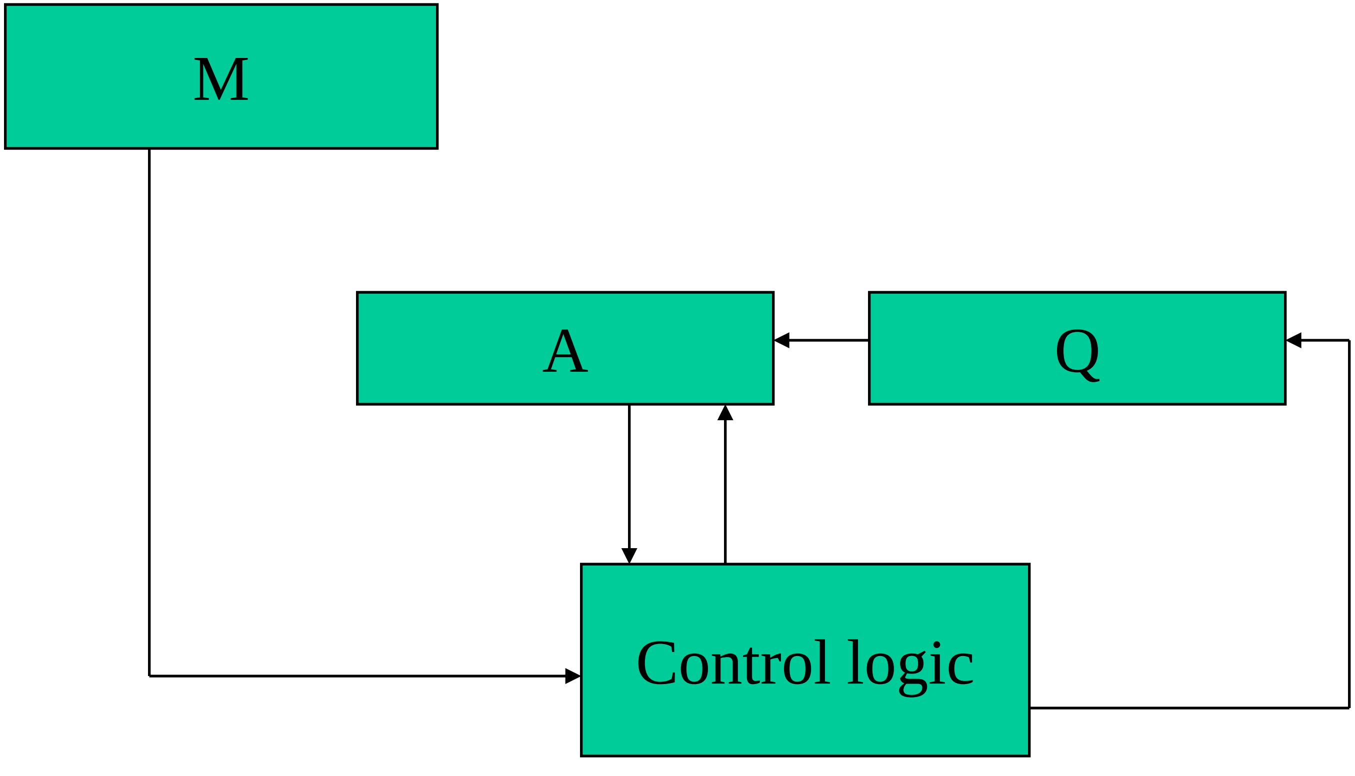
- When the event occurs a 1 is placed in the quotient and divisor subtracted from dividend

- The result is referred to as partial remainder

- Cyclic pattern followed

Example:

```
          00001101
       ___________
1011  |   10010011
      |    1011
       ___________
          001110
            1011
        ___________
            001111
                1011
         ___________
                0111
```



Block diagram of the implementation

Restoration Method

1.Divisor loaded into Register M

2.Dividend loaded into register Q

3.Initialise register A to Zero

4.Shift A and Q left by one position

5.Subtract M from A, placing the result back in A

6. If MSB of A is 1, set Q0 to Zero, add M to A (restore A)

If MSB of A is Zero, set Q0 to 1

Repeat steps 4,5,6 n times where n is
No of bits of divisor

N bit quotient obtained from Q and remainder from A

Example: Divide 7 by 3

Dividend : 0111
Divisor : 0011 M : 0011

A	Q	
0000	0111	Initial Value
0000	1110	shift left
1101		Sub M

1101	**1110**	**As MSB is 1 set Q0 =0**
0011		**Add M**
0000	**1110**	**Restore A**
0001	**1100**	**Shift left**
1101		**Sub M**
1110	**1100**	**As MSB is 1 set Q0= 0**
0011		**Add M**
0001	**1100**	**Restore A**

0011	**1000**	**Shift left**
1101		**Sub M**
0000	**1000**	**As MSB 0 set Q0 = 1**
0000	**1001**	
0001	**0010**	**Shift left**
1101		**Sub M**
1110	**0010**	**As MSB 1set Q0 = 0**
0011		**Add M**
0001	**0010**	**Restore A**

Result Q = 0010   Remainder = 0001

Divide 9 by 3

Dividend = 1001

Divisor = 0011 **M = 0011**

Non-restoration Method

Eliminates the need for restoring A after the result of subtraction is negative

1.Divisor is loaded into M

2.Dividend loaded into Q

3.Initialise A to Zero

4.If sign of A is positive shift A & Q left by One bit and subtract M from A

OR

If sign of A is negative, shift A &Q by one bit and add M to A

5. If MSB of result is 1, then Q0 is set to Zero otherwise set to one

Repeat steps 4 &5 n times

➔At the end if sign of A is negative add M to A to get the correct remainder

7 divided by 3

M 0011

A	Q	
0000	**0111**	**initialise**
0000	**1110**	**shift left, as MSB is 0**
1101		**Sub M**
1101	**1110**	**as MSB 1, Q0 = 0**
1011	**1100**	**shift left, as MSB 1**
0011		**Add M**
1110	**1100**	**as MSB 1 Q0 = 0**

1101	**1000**	**shift left , as MSB 1**
0011		**Add M**
0000	**1000**	**as MSB 0**
0000	**1001**	**set Q0 =1**
0001	**0010**	**shift left as MSB 0**
1101		**Sub M**
1110	**0010**	**as MSB 1**
0011		**Add to adjust remainder**
0001		

Q = 0010 **R = 0001**