Overview

- **Computer Arithmetic's**
- **Addition and Subtraction**
- Multiplication Algorithms
- Division Algorithms
- Decimal Arithmetic Unit
- BCD Operation
- Decimal Arithmetic operations

Computer Arithmetic's

➢ **Arithmetic Instruction** Manipulate data to produce results necessary for the solution of computational problem

➢ Four Basic Arithmetic Operations are – **Addition, Subtraction, Multiplication and Division**

➢An **arithmetic processor** is the part of a processor unit that executes arithmetic operations

➢Arithmetic operations can be performed for following data types

- Fixed point binary data in signed magnitude representation
- Fixed point binary data in signed 2's complement representation
- Floating Point binary data
- Binary coded decimal data

Addition and Subtraction

Representation of both *positive* and *negative* numbers

- Following 3 representations

Signed magnitude representation
Signed 1's complement representation
Signed 2's complement representation

Example: Represent +9 and -9 in 7 bit-binary number

Only one way to represent + 9 ==> 0 001001

Three different ways to represent - 9:

In signed-magnitude: 1 001001
In signed-1's complement: 1 110110
In signed-2's complement: 1 110111

Fixed point numbers are represented either integer part only or fractional part only.

Addition and Subtraction

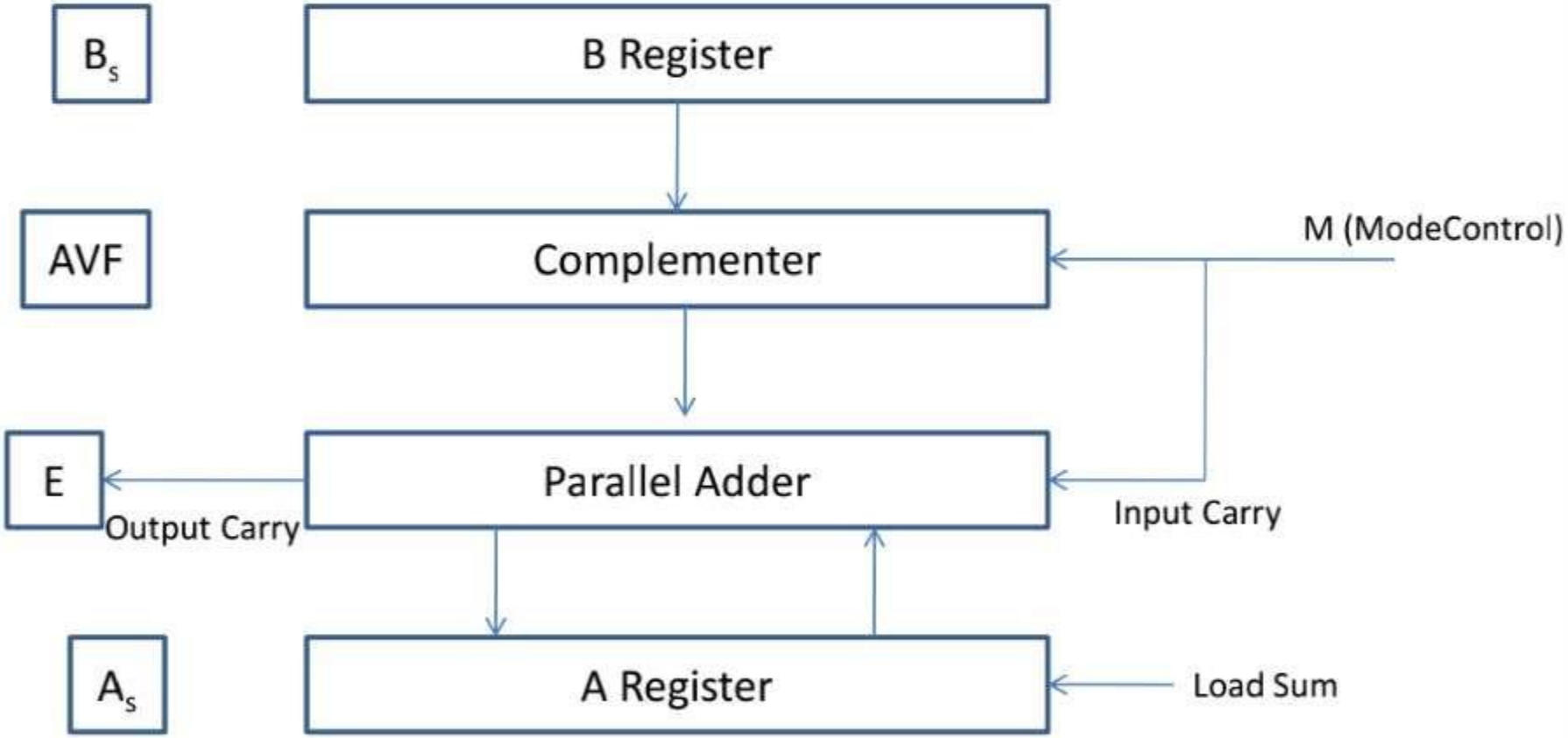
Addition and Subtraction with Signed Magnitude Data

Operation	Add Magnitudes	Subtract Magnitudes		
		A>B	A<B	A=B
(+ A) + (+ B)	+ (A + B)			
(+ A) + (- B)		+ (A - B)	- (B - A)	+ (A - B)
(- A) + (+ B)		- (A - B)	+ (B - A)	+ (A - B)
(- A) + (- B)	- (A + B)			
(+ A) - (+ B)		+ (A - B)	- (B - A)	+ (A - B)
(+ A) - (- B)	+ (A + B)			
(- A) - (+ B)	- (A + B)			
(- A) - (- B)		- (A - B)	+ (B - A)	+ (A - B)

Computer Arithmetic's

- **Addition (subtraction) Algorithm**
  - When the sign of A and B are identical (different) , add the magnitudes and attach the sign of A to the result.
  - When the signs of A and B are different (identical), compare the magnitudes and subtract the smaller number from the larger.
    - Choose the sign of result to be same as A if A>V
    - or the complement of sign of A if A<B
    - if A=B subtract B from A and make the sign of result positive

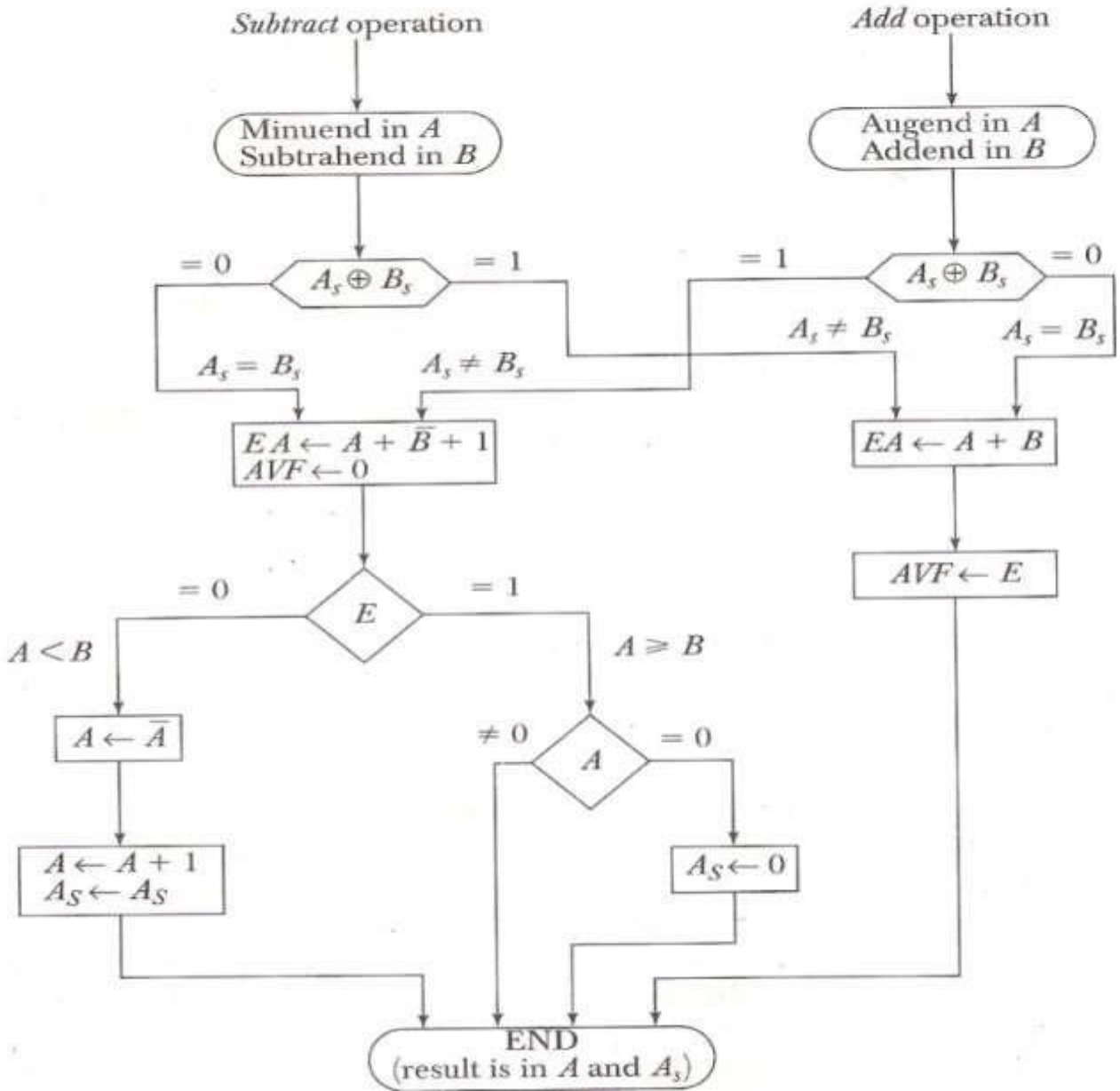
Hardware Implementation



Simple procedure require magnitude comparator, an adder, two subtractor however alternative reveals that using 2's complement for operation requires only an adder and a complementor

M=0 output = **A+B** **M=1** output = A+B'+1= **A-B**

Flow Chart for Add and Subtract Operation



Signed 2's Complement Representation

- **Addition :**

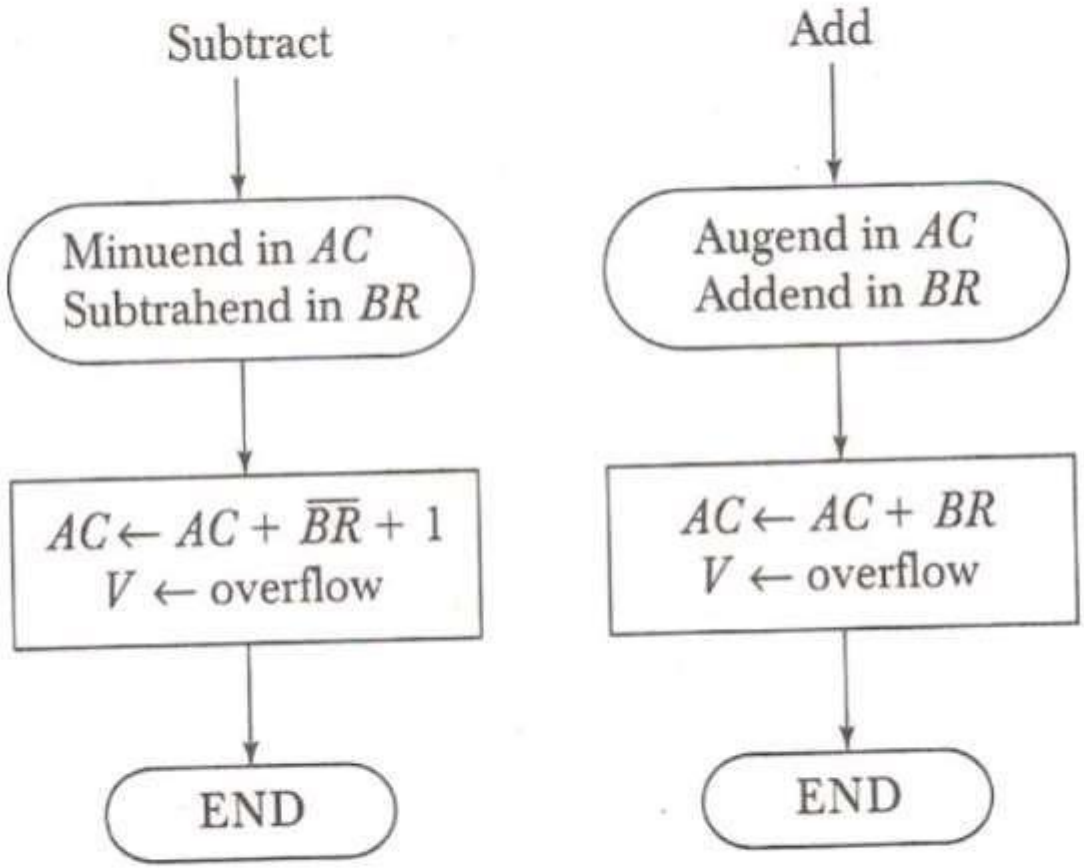
  - Addition of two numbers in signed 2's complement form consists of adding the numbers with the sign bits treated the same as the other bits of the number. Carry out of sign bit is discarded

  - Sum is obtained by adding the content of AC and BR (including the sign bit). Overflow bit is set to 1 if EX-OR of last two carries if 1

- **Subtraction :**
  - Here Subtraction consists of first taking the 2's complement of the subtrahend and then adding it to minuend

  - Subtraction done by adding the content of AC to 2's Complement of BR.

Signed 2's Complement Representation



Overview

- Computer Arithmetic's
- Addition and Subtraction
- **Multiplication Algorithms**
- **Division Algorithms**
- Decimal Arithmetic Unit
- BCD Operation
- Decimal Arithmetic operations

Binary Multiplication

23	10111		Multiplicand
19	× 10011		Multiplier
	10111		
	10111		
	00000	+	
	00000		
	10111		
437	110110101		Product

Sign of product determined from sign of Multiplicand and Multiplier , if same +ve else -ve

H/W implementation -Multiplication

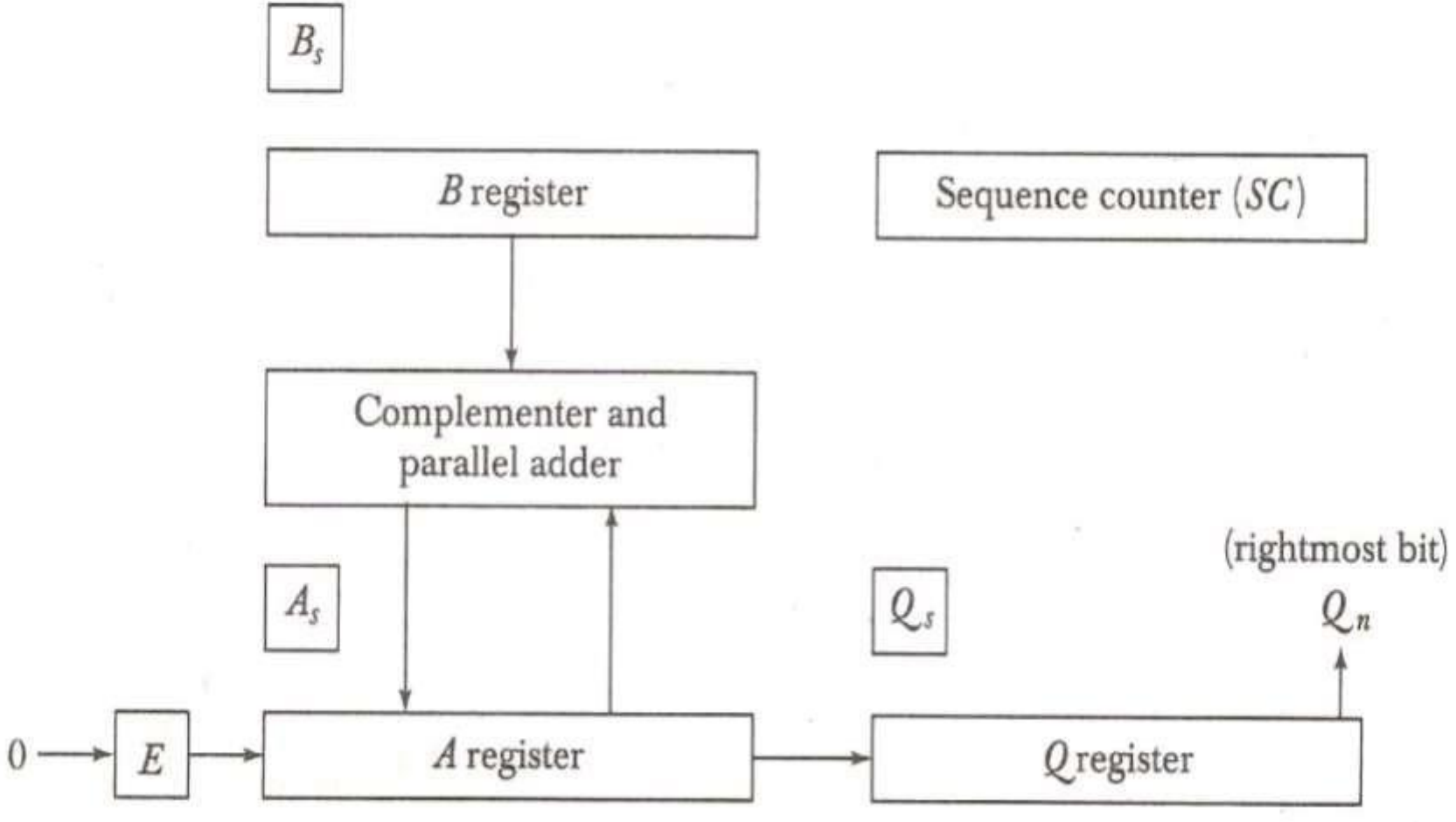
For multiplication in hardware, process is slightly changed

1. In place of as Register equal to no. of multiplier bits, use one adder and one Register

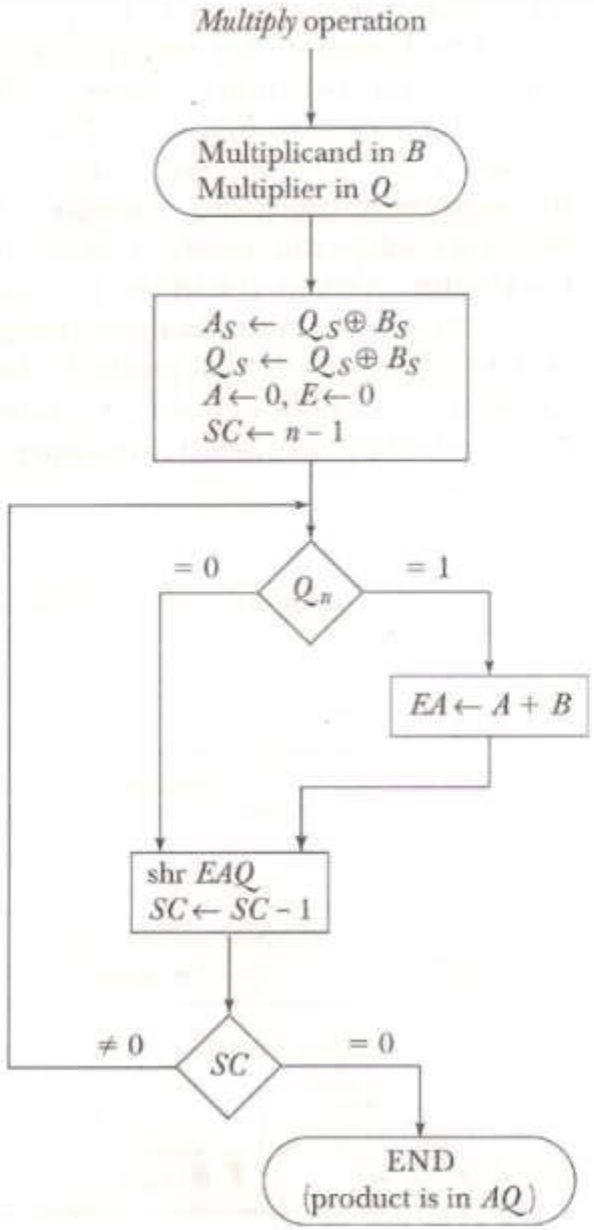
2. Instead of shifting multiplicand to left , partial product is shifted right

3. When corresponding bit of multiplier is 0, no need to add zero to partial product

H/W implementation -Multiplication



Flow Chart Binary Multiplication



Booth's Multiplication Algorithm

- **Used for multiplication in Signed 2's complement representation**

- It operates on the fact that strings of 0's in the multiplier require no addition but just shifting

- And a string of 1's in the multiplier from bit weight 2^k to 2^m can be treated as $2^{k+1} - 2^m$

- E.g. +14 (001110) here k=3, m=1 represented as $2^4-2^1 = 16-2 = 14$

  Thus $M \times 14 = M \times 2^4 - M \times 2^1$

Booth's Multiplication Algorithm

➢**Used for multiplication in Signed 2's complement representation**

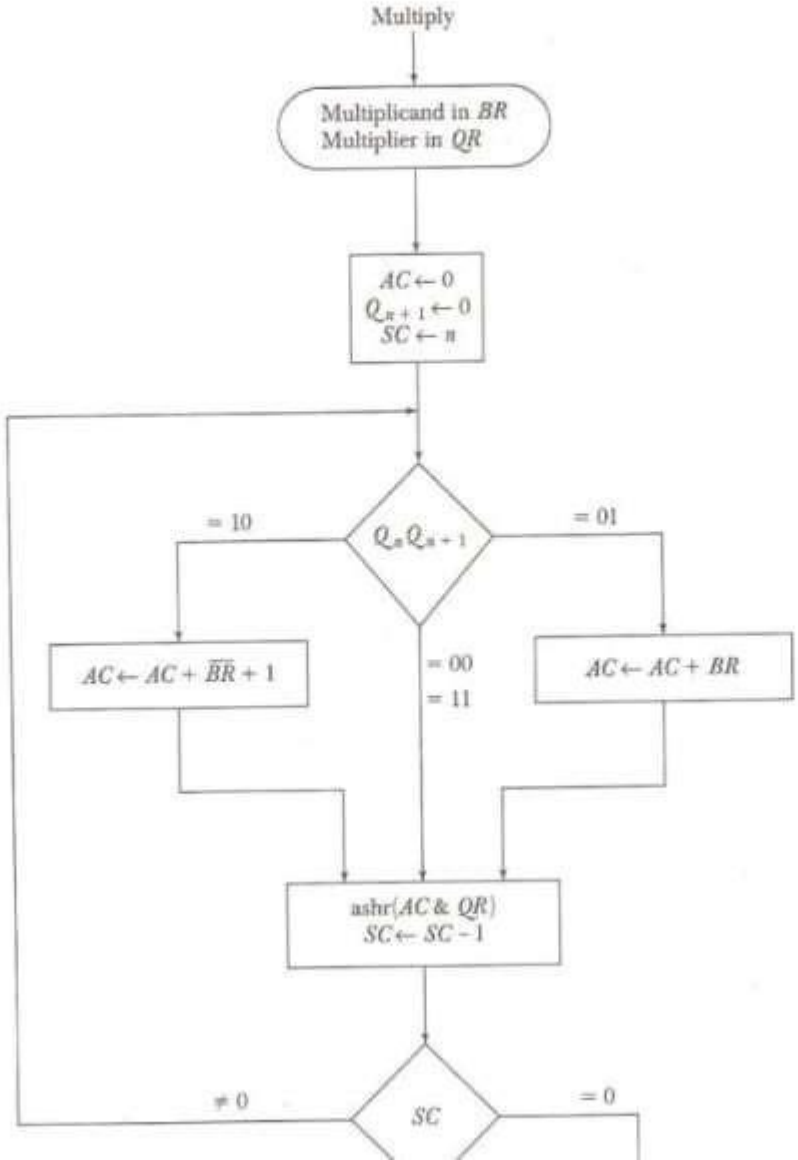
➢Prior to shifting the multiplicand can be added to the partial product , subtracted from the partial product or left unchanged according to the following rules :

1. The multiplicand is subtracted from partial product upon encountering the first least significant 1 in the string of 1's in the multiplier

2. The multiplicand is added to the partial product upon encountering the first 0 (provided that there was a previous 1) in the string of 0's in the multiplier.

3. The partial product does not change when the multiplier bit is identical to the previous multiplier bit

Booth's Multiplication Algorithm



Example - Booth's Multiplication Algorithm

Q_nQ_{n+1}	$BR = 10111$ $\overline{BR} + 1 = 01001$	AC	QR	Q_{n+1}	SC
	Initial	00000	10011	0	101
1 0	Subtract BR	01001			
		01001			
	ashr	00100	11001	1	100
1 1	ashr	00010	01100	1	011
0 1	Add BR	10111			
		11001			
	ashr	11100	10110	0	010
0 0	ashr	11110	01011	0	001
1 0	Subtract BR	01001			
		00111			
	ashr	00011	10101	1	000