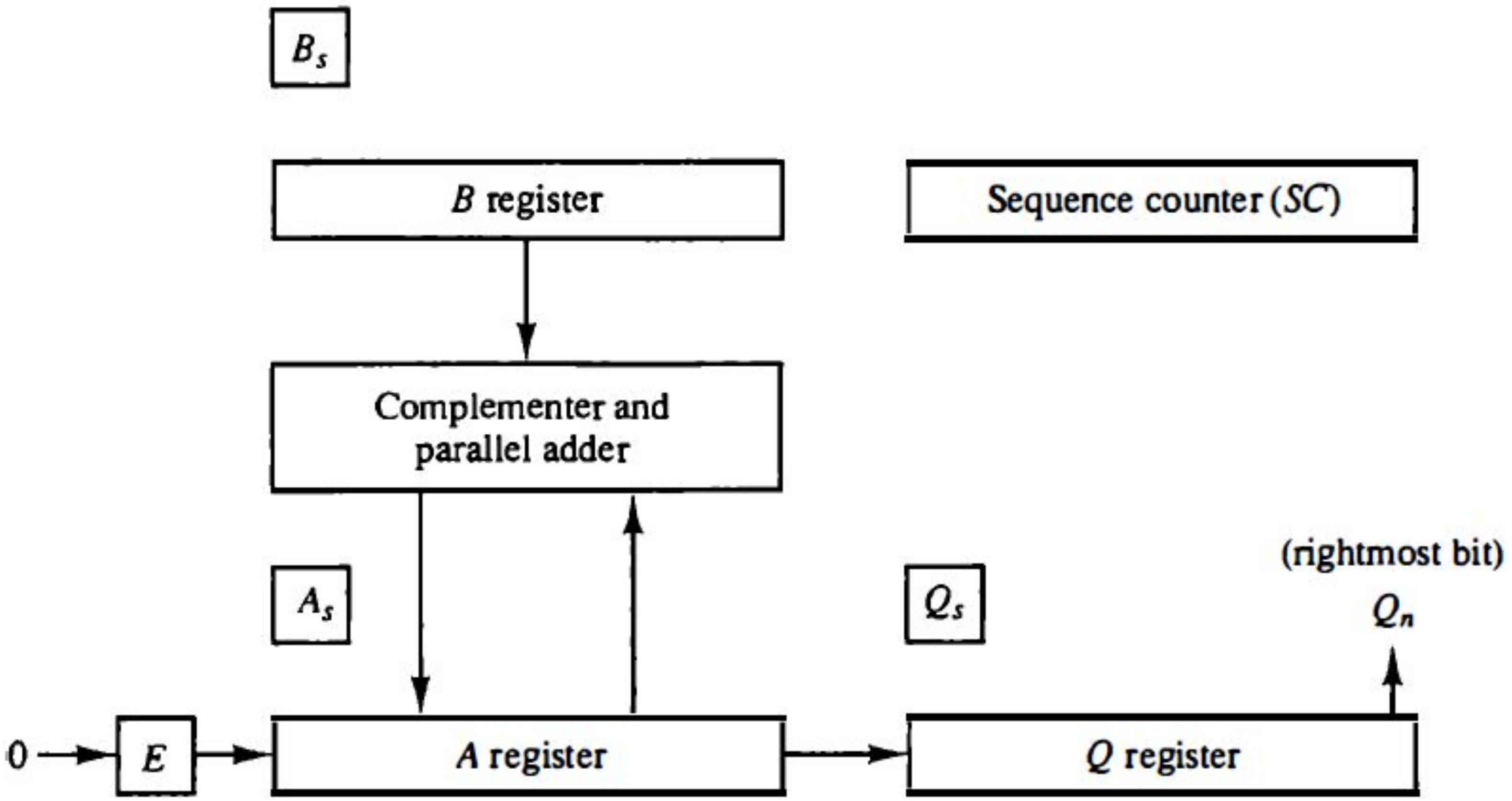
Multiplication Algorithm

- Multiplication of two fixed-point binary numbers in signed-magnitude representation is done by a process of successive shift and add operations.
- Look at successive bits of the multiplier, least significant bit first.
- If multiplier bit is 1, the multiplicand is copied down. Otherwise, zeros are copied down.
- Number copied in successive lines are shifted one position to the left from the previous number.
- Add numbers, Sum forms the product.

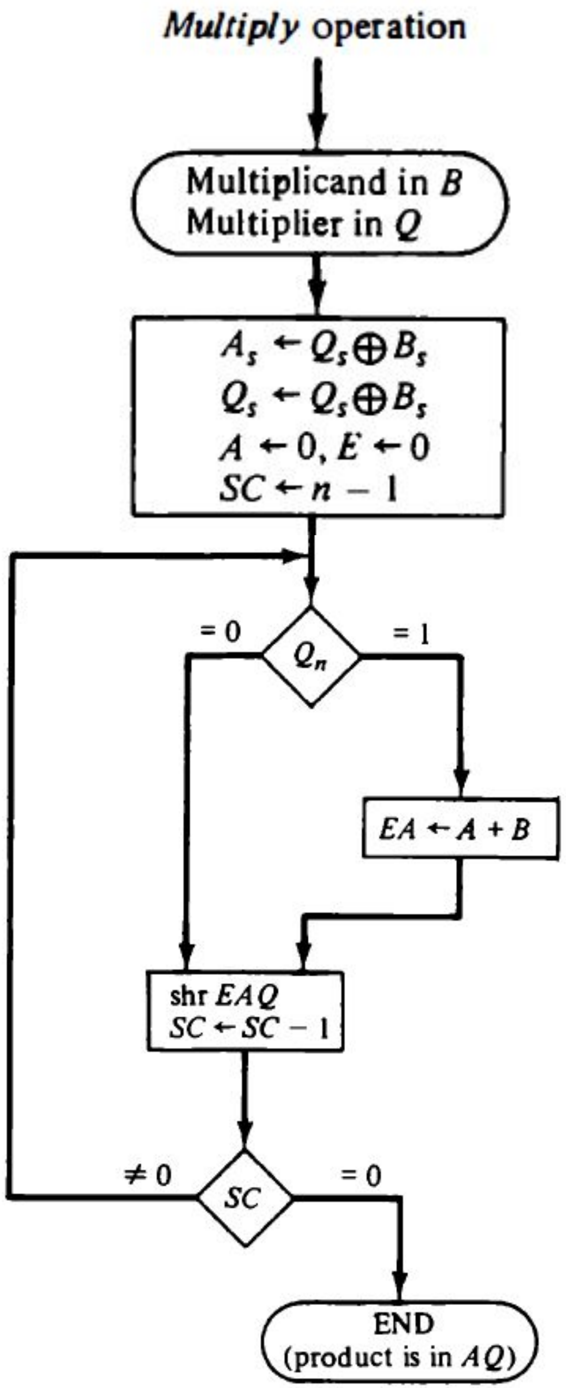
The sign of the product is determined from the signs of the multiplicand and multiplier. If they are alike, the sign of the product is positive. If they are unlike, the sign of the product is negative.

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

Hardware Implementation



- Multiplicand is stored in Register B and multiplier in Q.
- Sequence Counter SC is initially set to a number equal to number of bits in the multiplier.
- Counter is decremented by 1 after forming each partial product.
- Sum of A and B forms the partial product which is transferred to the EA register.
- Both partial product and multiplier are shifted to the right. (shr EAQ)
- LSB of A is shifted into the MSB of Q, bit from E is shifted into the MSB of A and 0 is shifted into E.
- In this manner right most bit of the multiplier will be inspected next.



Multiply operation
Multiplicand in B
Multiplier in Q
$A_s \leftarrow Q_s \oplus B_s$
$Q_s \leftarrow Q_s \oplus B_s$
$A \leftarrow 0, E \leftarrow 0$
$SC \leftarrow n - 1$
Q_n
= 0
= 1
$EA \leftarrow A + B$
shr EAQ
$SC \leftarrow SC - 1$
SC
≠ 0
= 0
END
(product is in AQ)

Numerical Example for Binary Multiplier

Multiplicand $B = 10111$	E	A	Q	SC
Multiplier in Q	0	00000	10011	101
$Q_n = 1$; add B		10111		
First partial product	0	10111		
Shift right EAQ	0	01011	11001	100
$Q_n = 1$; add B		10111		
Second partial product	1	00010		
Shift right EAQ	0	10001	01100	011
$Q_n = 0$; shift right EAQ	0	01000	10110	010
$Q_n = 0$; shift right EAQ	0	00100	01011	001
$Q_n = 1$; add B		10111		
Fifth partial product	0	11011		
Shift right EAQ	0	01101	10101	000
Final product in $AQ = 0110110101$				

Booth Multiplication Algorithm

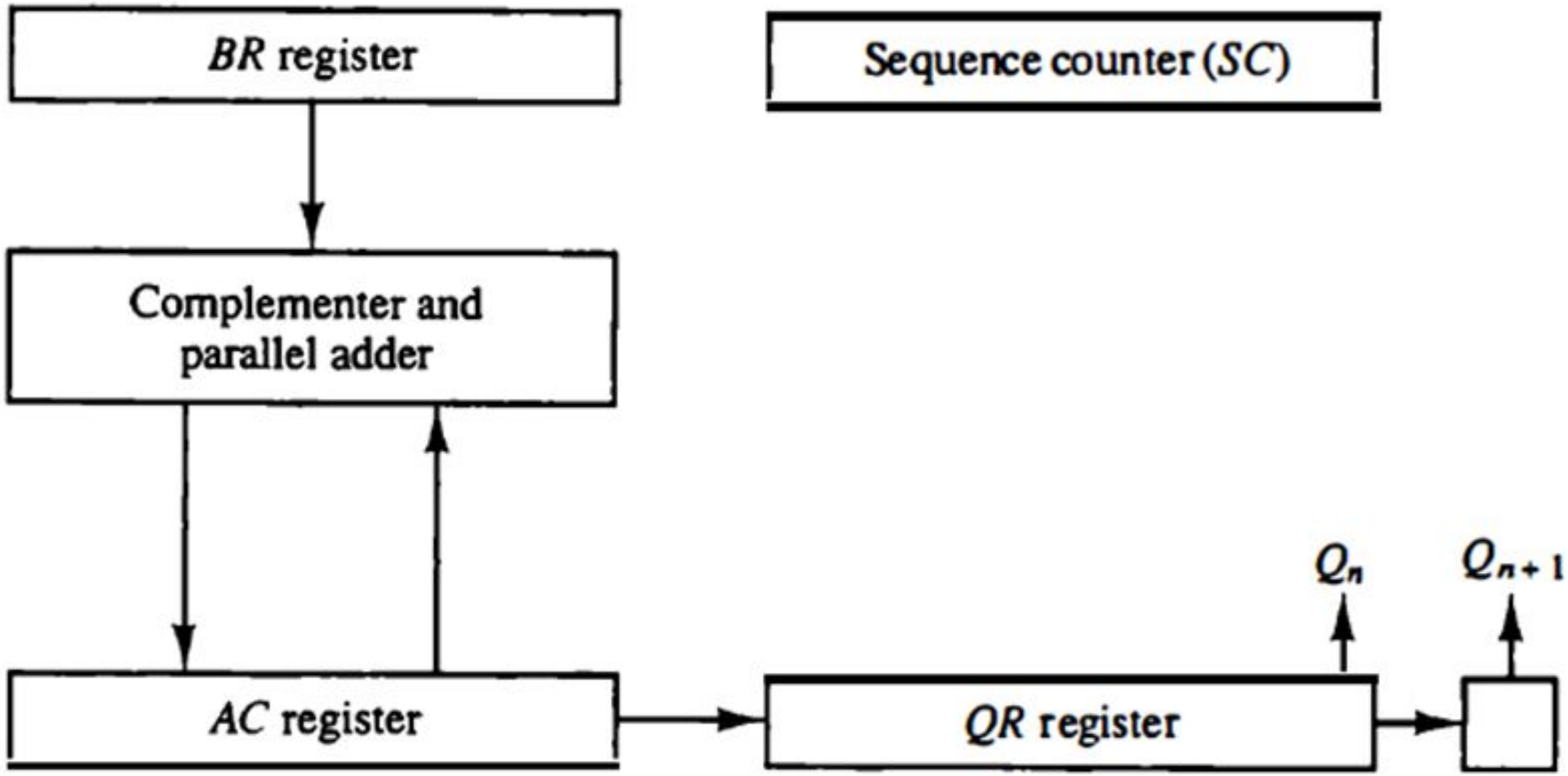
- It gives a procedure for multiplying binary integers in signed 2's complement form.
- Algorithm was invented by Andrew Donald Booth in 1950.
- Strings of 0's in multiplier -> No addition, just shifting
- Strings of 1's in multiplier from bit weight 2^k to weight 2^m can be treated as $2^{k+1} - 2^m$

  001110(+14)

  (k=3, m=1) -> $2^{k+1} - 2^m$ -> $2^4 – 2^1$ -> 16 - 2 = 14

  M x 14 -> M x 2^4 - M x 2^1

Hardware for Booth Algorithm



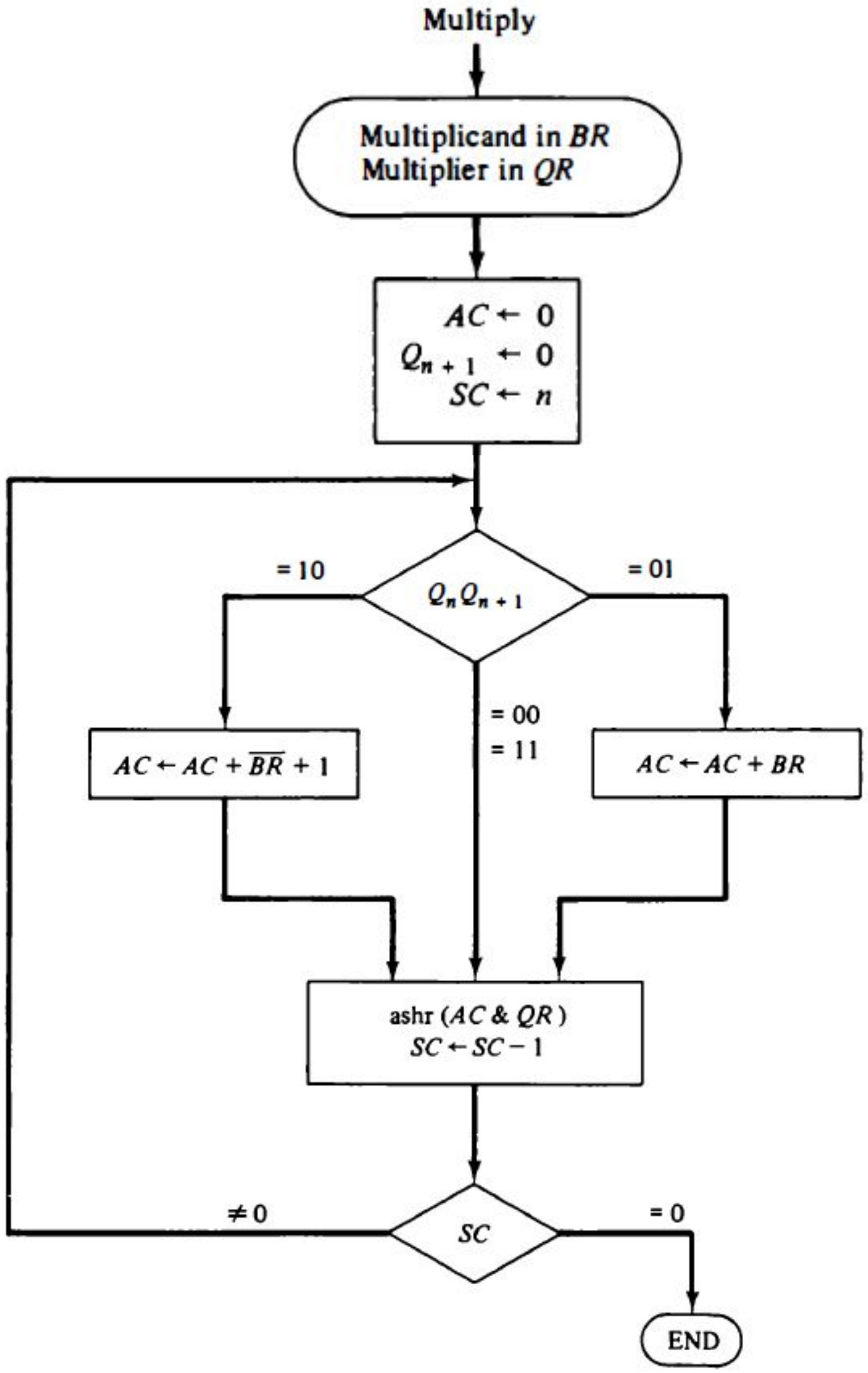
Hardware Implementation

- Sign bits are not separated from the rest of the registers.
- Q_n designates least significant bit of the multiplier in register QR.
- An extra flip-flop Q_{n+1} is appended to QR to facilitate a double bit inspection of the multiplier.

- Multiplicand is subtracted from partial product when first least significant 1 in a string of 1's in multiplier is encountered.

- Multiplicand is added to partial product upon encountering the first 0 (Provided that there was previous 1) in a string of 0's in the multiplier.

- Partial product does not change when the multiplier bit is identical to the previous multiplier bit.



Example of Multiplication with Booth Algorithm **(-9) x (-13) = +117**

Q_n Q_{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