Booths Algorithm with sign-extend method.

num1 × num2
→ multiplicand × multiplier = product

• low-order bit and previous bit (initially 0) of product

00 -> no operation –

shift product right (with sign extend)

01 -> product = left side of product + multiplicand

shift product right (with sign extend)

10 -> product = left side of product - multiplicand

shift product right (with sign extend)

11 -> no operation

shift product right (with sign extend)

- Left side of product (ls/prod) initially set to 0
- Right side of product is the multiplier
- Complete N iterations (N machine word size)

Example 1 (4-bit word size):
2 × 6 (0010 × 0110)
multiplicand = 0010; multiplier = 0110;
BB- Booth Bit
Q_n – Multiplier LSB or Product LSB ; Q_{n+1} – Booth Bit (BB)

step	**action**	**multiplicand**	**Product BB**
0	initial	0010	0000 0110 0
1	a: 00 -> no operation	0010	0000 0110 0
	b: shift right product	0010	0000 0011 0
2	a: 10 -> prod = ls/prod - Mcand	0010	1110 0011 0
	b: shift right product	0010	1111 0001 1
3	a: 11 -> no operation	0010	1111 0001 1
	b: shift right product	0010	1111 1000 1
4	a: 01 -> prod = ls/prod + Mcand	0010	0001 1000 1
	b: shift right product	0010	0000 1100 0

Example 2 (8-bit word size):
18 × -7 (0001 0010 × 1111 1001)

multiplicand = 0001 0010 multiplier = 1111 1001

step	action	multiplicand	Product	BB
	initial	0001 0010	0000 0000 1111 1001	0
1	a: 10 -> prod = ls/prod - Mcand	0001 0010	1110 1110 1111 1001	0
	b: shift right product	0001 0010	1111 0111 0111 1100	1
2	a: 01 -> prod = ls/prod + Mcand	0001 0010	0000 1001 0111 1100	1
	b: shift right product	0001 0010	0000 0100 1011 1110	0
3	a: 00 -> no operation	0001 0010	0000 0100 1011 1110	0
	b: shift right product	0001 0010	0000 0010 0101 1111	0
4	a: 10 -> prod = ls/prod - Mcand	0001 0010	1111 0000 0101 1111	0
	b: shift right product	0001 0010	1111 1000 0010 1111	1
5	a: 11 -> no operation	0001 0010	1111 1000 0010 1111	1
	b: shift right product	0001 0010	1111 1100 0001 0111	1
6	a: 11 -> no operation	0001 0010	1111 1100 0001 0111	1
	b: shift right product	0001 0010	1111 1110 0000 1011	1
7	a: 11 -> no operation	0001 0010	1111 1110 0000 1011	1
	b: shift right product	0001 0010	1111 1111 0000 0101	1
8	a: 11 -> no operation	0001 0010	1111 1111 0000 0101	1
	b: shift right product	0001 0010	1111 1111 1000 0010	1

Booth's Algorithm Flowchart