Booth's Multiplication Algorithm

Worked Example: Multiply -3 × 2
(4-bit)

Intuition Behind Booth's Algorithm

- • Handles signed binary multiplication efficiently using 2's complement.
- • Reduces operations for runs of 1s in multiplier.
- • Uses a combination of add/subtract and arithmetic right shift.
- • Based on detecting transitions in Q_0 and Q_{-1}.

Booth's Algorithm Steps

- 1. Initialize A = 0, Q = multiplier, Q-1 = 0, M = multiplicand.
- 2. For n bits:
- a. If $Q_0Q_{-1} = 10$: A = A - M
- If $Q_0Q_{-1} = 01$: A = A + M
- b. Perform arithmetic right shift on [A, Q, Q-1]
- 3. Final result is in [A, Q] (concatenated).

Initial Setup (4-bit Example)

- M = -3 $\longrightarrow$ 1101
- Q = 2 $\longrightarrow$ 0010
- A = 0000, Q-1 = 0
- -M (2's complement of 1101) = 0011

Step 1

- $Q_0Q_{-1} = 00 \longrightarrow$ No Operation
- Right shift [A|Q|Q-1] = 0000 0010 0
- Result: A = 0000, Q = 0001, Q-1 = 0

Step 2

- $Q_0Q_{-1} = 10 \longrightarrow A = A - M \longrightarrow 0000 - 1101 = 0011$
- Right shift [A|Q|Q-1] = 0011 0001 0
- Result: A = 0001, Q = 1000, Q-1 = 1

Step 3

- $Q_0Q_{-1} = 01 \longrightarrow A = A + M \longrightarrow 0001 + 1101 = 1110$
- Right shift [A|Q|Q-1] = 1110 1000 1
- Result: A = 1111, Q = 0100, Q-1 = 0

Step 4

- $Q_0Q_{-1} = 00 \longrightarrow$ No Operation
- Right shift [A|Q|Q-1] = 1111 0100 0
- Result: A = 1111, Q = 1010, Q-1 = 0

Final Result

- Final [A|Q] = 11111010 (8-bit 2's complement)
- ⟶ 2's complement of 11111010 = 00000110
  ⟶ 6
- ⟶ Final result = -6
- ✅ Verified: -3 × 2 = -6