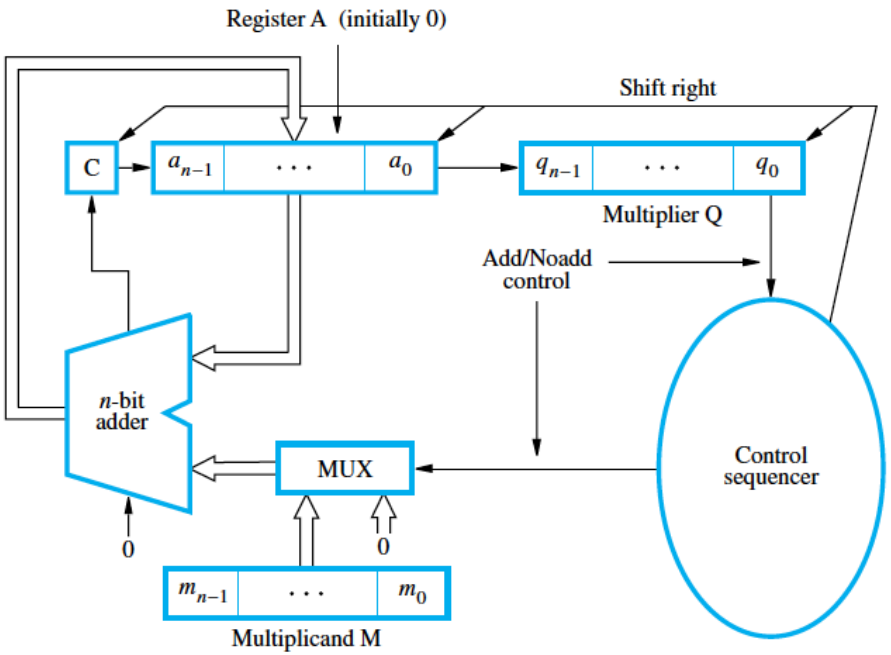
Integer Multiplication

To multiply by 2^n, just shift left n bits

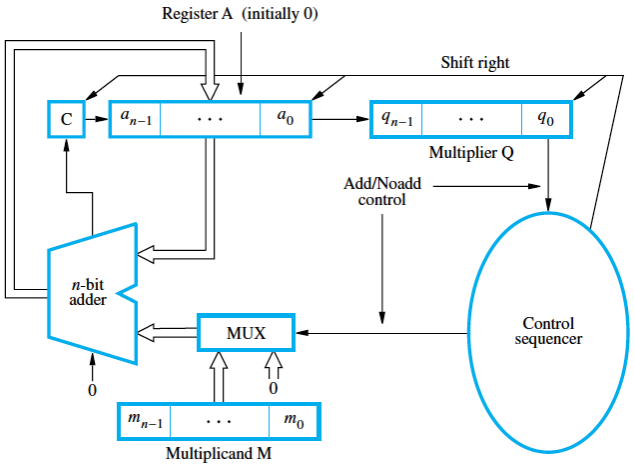
In general, multiplication involves shifting and adding



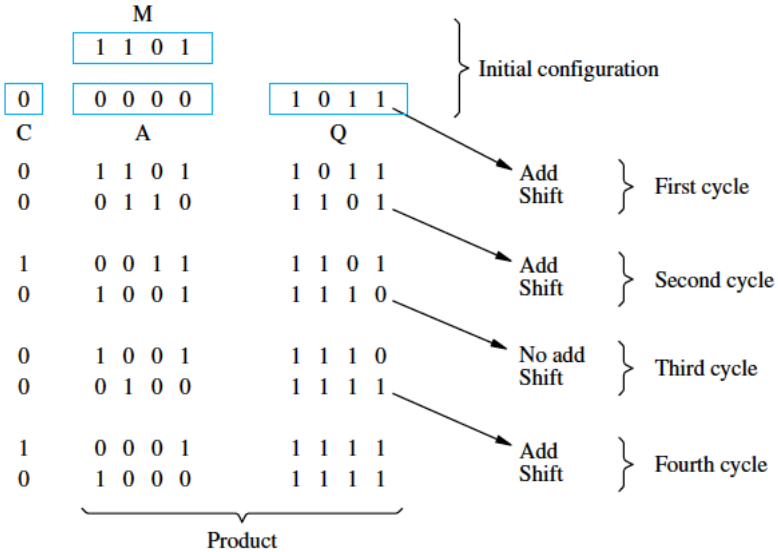
The product of two n-bit numbers requires 2n bits

Configuration for unsigned multiplication

Integer Multiplication



4-bit numbers (-13 x 11) unsigned multiply



Repeat N times (N= 4 here)

- Add multiplicand to the A register only if the LSB of Q is 1
- Shift for each cycle (C, A and Q together)

Generates an 8-bit product

Compute 13 x 11 and negate the result to get -143

Booth's Algorithm

A more general technique

- Works for both positive and negative factors
- Reduces the number of operations required

Example: suppose we want to multiply by 30 = 0011110_2

30 can be regarded as

$$\begin{array}{rll} & 0100000 & (32) \\ - & 0000010 & (2) \\ \hline & 0011110 & (30) \end{array}$$

i.e., multiply by (32 – 2)

Add 32 x multiplicand plus -2 x multiplicand

Booth's Algorithm

Multiplier		Version of multiplicand selected by bit i
Bit i	Bit $i-1$	
0	0	$0 \times M$
0	1	$+1 \times M$
1	0	$-1 \times M$
1	1	$0 \times M$

Scan multiplier bits from right to left
(assume initial 0 bit to right of LSB)
- subtract multiplicand when moving from 0 to 1
- add multiplicand when moving from 1 to 0
- neither add nor subtract when moving from 0 to 0 or 1 to 1
- shift multiplicand left one bit for each cycle

The original multiply is "*recoded*"

30 = 0011110_2 using 7 bits

0 0 1 1 1 1 1 0

⇓

0 +1 0 0 0 0 -1 0

Subtract multiplicand for each -1 in recoded multiplier
Add multiplicand for each +1 in recoded multiplier
Neither add nor subtract for each 0 in recoded multiplier

Example: 45 x 30 Using Booth's Algorithm

45 → 0101101 -45 → 1010011

							0	1	0	1	1	0	1	← multiplicand
							0	+1	0	0	0	−1	0	← Recoded multiplier
0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	1	1	1	1	1	1	0	1	0	0	1	1		← 2's complement of the multiplicand (Sign extended)
0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	0	0	0	0	0	0	0	0				
0	0	0	0	0	0	0	0	0	0					
0	0	0	1	0	1	1	0	1						
0	0	0	0	0	0	0	0							
0	0	0	1	0	1	0	1	0	0	0	1	1	0	→ 0x0546 = 1350 = 45 x 30

One addition and one subtraction is required to generate the product
Using 00111110 would have required 5 additions

Another Booth's Recoding Example

Assume a 17-bit multiplier:

0	0	1	0	1	1	0	0	1	1	1	0	1	0	1	1	0	0
								⇓									
0	+1	−1	+1	0	−1	0	+1	0	0	−1	+1	−1	+1	0	−1	0	0