



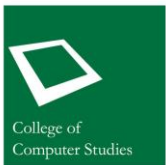
# CSARCH Lecture Series: Binary Integer Multiplication

Sensei RL Uy

College of Computer Studies

De La Salle University

Manila, Philippines



# Copyright Notice

This lecture contains copyrighted materials and is use solely for instructional purposes only, and not for redistribution.

Do not edit, alter, transform, republish or distribute the contents without obtaining express written permission from the author.

# Overview

Reflect on the following question:

- Are there other ways to perform integer binary multiplication besides the usual pencil-and-paper method?

$$\begin{array}{r} 0100 \\ \times 0101 \\ \hline \end{array}$$


Do I know how to  
multiply  
manually?

# Overview

- This sub-module introduces the concepts of integer binary multiplication
- The objectives are as follows:
  - ✓ Describe the process of performing integer binary multiplication using pencil-and-paper method
  - ✓ Describe the process of performing integer binary multiplication using Booth's algorithm
  - ✓ Describe the process of performing integer binary multiplication using extended Booth's algorithm

# ALU Multiplication

- Some methods in performing integer binary multiplication:
  - Pencil-and-paper method
  - Booth's algorithm
  - Extended Booth's algorithm

# Pencil-and-Paper Method

Steps:

- 1.) represent both multiplicand ( $m$ ) and multiplier ( $n$ ) using 2's complement format
- 2.) the first intermediate product has a length of  $m+n$  bits. Sign extend as needed
- 3.) As with pencil-and-paper multiplication, succeeding intermediate products are shifted right
- 4.) if the multiplier is negative, add the 2's complement of the multiplicand, sign-extended and shifted to sign-bit position

# Pencil-and-Paper Method

$$\begin{array}{r} \phantom{x} \phantom{000000} 0100 \\ x \phantom{000000} 0101 \\ \hline 000000100 \\ 00000000 \\ 000100 \\ 00000 \\ \hline 00010100 \end{array}$$

Diagram illustrating the Pencil-and-Paper Method for binary multiplication. The multiplicand (0100) and multiplier (0101) are shown. The partial products are calculated and shifted right. The final result is 00010100.

Annotations:

- step 1: Multiplicand (0100) and Multiplier (0101) are identified.
- step 2: The first intermediate product (000000100) is shown.
- step 3: Subsequent intermediate products (00000000, 000100, 00000) are shown, shifted right.
- step 4 not needed: The final result (00010100) is reached.

Example:  $+4 * +5$

Steps:

- 1.) represent both multiplicand (m) and multiplier (n) using 2's complement format
- 2.) the first intermediate product has a length of  $m+n$  bits. Sign extend as needed
- 3.) As with pencil-and-paper multiplication, succeeding intermediate products are shifted right
- 4.) if the multiplier is negative, add the 2's complement of the multiplicand, sign-extended and shifted to sign-bit position

# Pencil-and-Paper Method

	1100	}	step 1
x	1011		
-----			
11111100		←	step 2
1111100		}	step 3
000000			
11100			
0100		←	step 4
-----			
00010100			

Example:  $-4 * -5$

Steps:

- 1.) represent both multiplicand (m) and multiplier (n) using 2's complement format
- 2.) the first intermediate product has a length of  $m+n$  bits. Sign extend as needed
- 3.) As with pencil-and-paper multiplication, succeeding intermediate products are shifted right
- 4.) if the multiplier is negative, add the 2's complement of the multiplicand, sign-extended and shifted to sign-bit position



# Pencil-and-Paper Method

	0100	}	step 1
x	1011		
-----			
00000100		←	step 2
0000100		}	step 3
000000			
00100			
1100		←	step 4
-----			
11101100			

Example:  $+4 * -5$

Steps:

- 1.) represent both multiplicand (m) and multiplier (n) using 2's complement format
- 2.) the first intermediate product has a length of  $m+n$  bits. Sign extend as needed
- 3.) As with pencil-and-paper multiplication, succeeding intermediate products are shifted right
- 4.) if the multiplier is negative, add the 2's complement of the multiplicand, sign-extended and shifted to sign-bit position

# Pencil-and-Paper Method

$$\begin{array}{r} \phantom{x} \phantom{000} 1100 \\ x \phantom{00} 0101 \\ \hline 11111100 \\ 00000000 \\ 111100 \\ 00000 \\ \hline 11101100 \end{array}$$

Diagram illustrating the Pencil-and-Paper Method for multiplying two 4-bit numbers in 2's complement format. The multiplicand is 1100 (representing -4) and the multiplier is 0101 (representing +5). The result is 11101100 (representing -20).

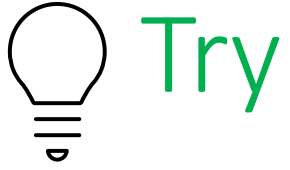
Annotations:

- step 1: Multiplicand (1100) and Multiplier (0101) are shown.
- step 2: The first intermediate product (11111100) is shown, which is the multiplicand shifted left by one position.
- step 3: Subsequent intermediate products (00000000, 111100, 00000) are shown, which are the multiplicand shifted left by two, three, and four positions respectively.
- step 4 not needed: The final result (11101100) is shown.

Example:  $-4 * +5$

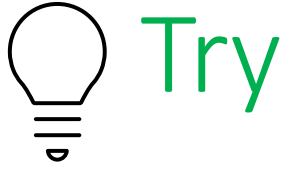
Steps:

- 1.) represent both multiplicand (m) and multiplier (n) using 2's complement format
- 2.) the first intermediate product has a length of  $m+n$  bits. Sign extend as needed
- 3.) As with pencil-and-paper multiplication, succeeding intermediate products are shifted right
- 4.) if the multiplier is negative, add the 2's complement of the multiplicand, sign-extended and shifted to sign-bit position



Try:  $+13 * -6$  (using Pencil-and-paper)

$$\begin{array}{r} 01101 \\ \times 11010 \\ \hline \end{array}$$



Try:  $+13 * -6$  (using Pencil-and-paper)

```
  01101
x 11010
-----
```

```

      01101
    x 11010
    -----
000000000000
000001101
000000000
0001101
001101
10011
-----
1110110010
```

step 1

step 2

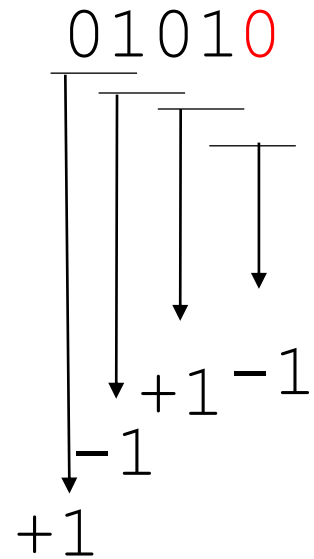
step 3

step 4

# Booth's Algorithm

- Treats positive and negative multipliers uniformly.
- Rewrites multiplier in terms of sums and differences.
- Convert code according to next bit at right
- $0 \text{ to } 1 \Rightarrow +1$
- $1 \text{ to } 0 \Rightarrow -1$
- Otherwise, 0
- Right of lsb is “nothing”, *i.e.*, equal to 0

# Booth's Algorithm



Steps in obtaining Booth's equivalent of a binary number

1. append 0 at the LSb side
2. pair 2 bits starting at LSb
3.  $00 \rightarrow 0$ ;  $01 \rightarrow +1$ ;  $10 \rightarrow -1$ ;  $11 \rightarrow 0$

$$0 * m = 0$$

$$1 * m = m$$

$$-1 * M = 2's \text{ complement}(m)$$

# Booth's Algorithm

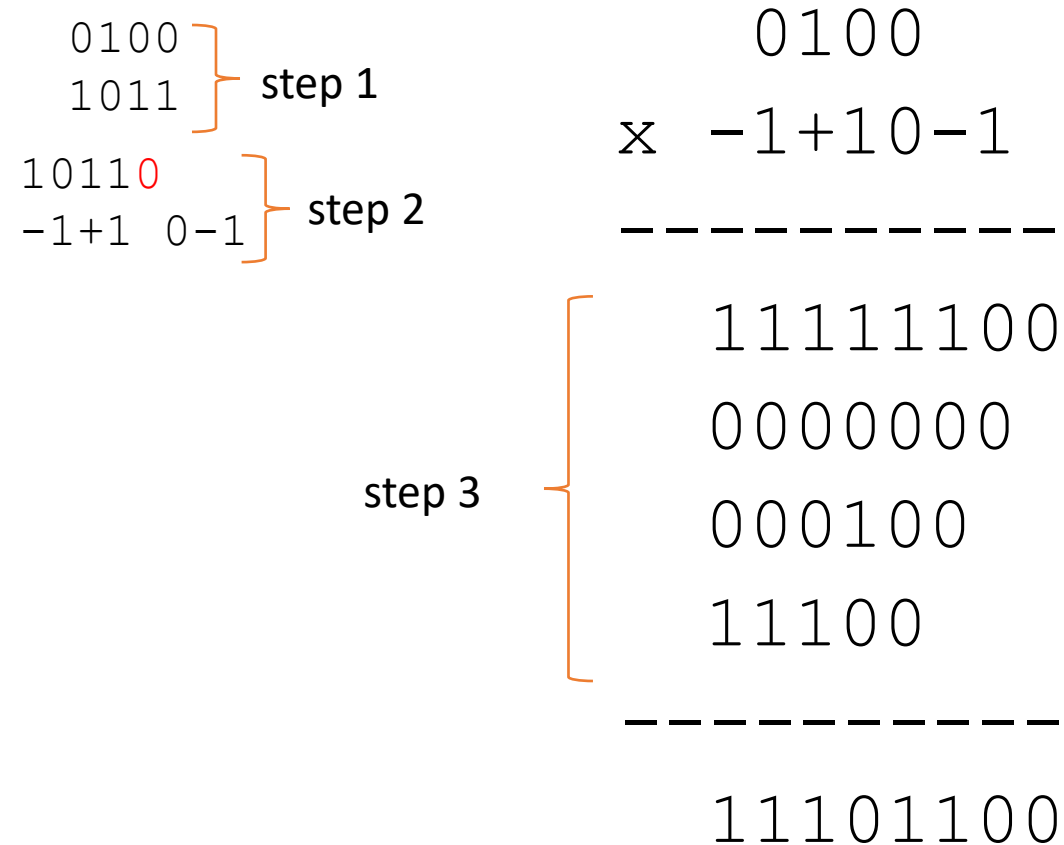
$$\begin{array}{r}
 \begin{array}{l}
 0100 \\
 0101
 \end{array} \left. \vphantom{\begin{array}{l} 0100 \\ 0101 \end{array}} \right\} \text{step 1} \\
 \begin{array}{l}
 01010 \\
 +1-1+1-1
 \end{array} \left. \vphantom{\begin{array}{l} 01010 \\ +1-1+1-1 \end{array}} \right\} \text{step 2} \\
 \begin{array}{r}
 \begin{array}{l}
 11111100 \\
 0000100 \\
 111100 \\
 00100
 \end{array} \left. \vphantom{\begin{array}{l} 11111100 \\ 0000100 \\ 111100 \\ 00100 \end{array}} \right\} \text{step 3} \\
 \hline
 00010100
 \end{array}$$

Example:  $+4 * +5$

Steps:

- 1.) represent both multiplicand (m) and multiplier (n) using 2's complement format
- 2.) convert multiplier to its Booth's equivalent
  - append 0 at the LSb side
  - pair 2 bits starting at LSb
  - $00 \rightarrow 0$ ;  $01 \rightarrow +1$ ;  $10 \rightarrow -1$ ;  $11 \rightarrow 0$
- 3.) Multiply using pencil-and-paper method ignoring extra steps if multiplier is negative

# Booth's Algorithm

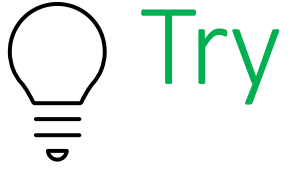


Example: +4 \* -5

Steps:

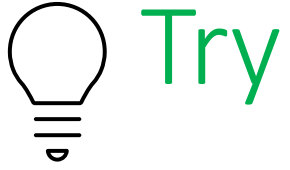
- 1.) represent both multiplicand (m) and multiplier (n) using 2's complement format
- 2.) convert multiplier to its Booth's equivalent
  - append 0 at the LSb side
  - pair 2 bits starting at LSb
  - 00 → 0; 01 → +1; 10 → -1; 11 → 0
- 3.) Multiply using pencil-and-paper method ignoring extra steps if multiplier is negative





Try:  $+13 \times -6$  (using Booth's algorithm)

$$\begin{array}{r} 01101 \\ \times 11010 \\ \hline \end{array}$$



Try:  $+13 \times -6$  (using Booth's algorithm)

```
  01101
x 11010
-----
```

```
  01101
  11010 } step 1
110100 } step 2
0-1+1-10
```

```
      01101
x 0-1+1-10
-----
0000000000
111110011
00001101
1110011
000000
-----
1110110010
```

## Extended Booth's Algorithm

- Also known as fast multiplication or bit-pair recording
- Bit-Pair Recording – reduces to half the number of summands The number of summands is reduced by pairing multiplier bits

Bit-pair recording:

0 0 0  $\Rightarrow$  0

0 0 1  $\Rightarrow$  +1

0 1 0  $\Rightarrow$  +1

0 1 1  $\Rightarrow$  +2

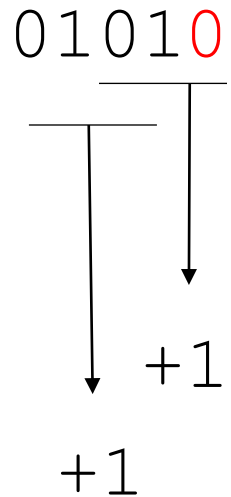
1 0 0  $\Rightarrow$  -2

1 0 1  $\Rightarrow$  -1

1 1 0  $\Rightarrow$  -1

1 1 1  $\Rightarrow$  0

# Extended Booth's Algorithm



Steps in obtaining Booth's equivalent of a binary number

1. append 0 at the LSb side
2. if odd number of bits, sign-extend
3. bit-pair starting at LSb

$$0 * m = 0$$

$$1 * m = m$$

$$-1 * m = 2's \text{ complement}(m)$$

$$+2 * m = m0$$

$$-2 * m = [2's \text{ complement}(m)]0$$

Bit-pair recording:

$$0 \ 0 \ 0 \Rightarrow 0$$

$$0 \ 0 \ 1 \Rightarrow +1$$

$$0 \ 1 \ 0 \Rightarrow +1$$

$$0 \ 1 \ 1 \Rightarrow +2$$

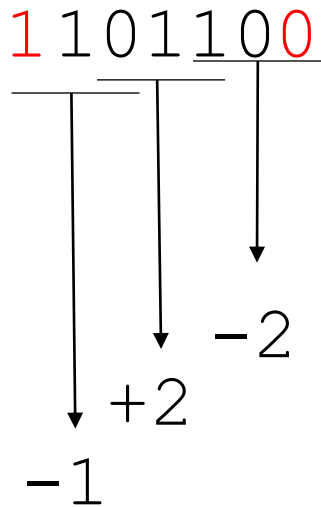
$$1 \ 0 \ 0 \Rightarrow -2$$

$$1 \ 0 \ 1 \Rightarrow -1$$

$$1 \ 1 \ 0 \Rightarrow -1$$

$$1 \ 1 \ 1 \Rightarrow 0$$

# Extended Booth's Algorithm



Steps in obtaining Extended Booth's equivalent of a binary number

1. append 0 at the LSb side
2. if odd number of bits, sign-extend
3. bit-pair starting at LSb

$$0 * m = 0$$

$$1 * m = m$$

$$-1 * m = 2's \text{ complement}(m)$$

$$+2 * m = m0$$

$$-2 * m = [2's \text{ complement}(m)]0$$

Bit-pair recording:

$$0 \ 0 \ 0 \Rightarrow 0$$

$$0 \ 0 \ 1 \Rightarrow +1$$

$$0 \ 1 \ 0 \Rightarrow +1$$

$$0 \ 1 \ 1 \Rightarrow +2$$

$$1 \ 0 \ 0 \Rightarrow -2$$

$$1 \ 0 \ 1 \Rightarrow -1$$

$$1 \ 1 \ 0 \Rightarrow -1$$

$$1 \ 1 \ 1 \Rightarrow 0$$

# Extended Booth's Algorithm

$$\begin{array}{r}
 0100 \\
 0101 \\
 \hline
 01010 \\
 +1+1 \\
 \hline
 \end{array}
 \begin{array}{l}
 \text{step 1} \\
 \text{step 2}
 \end{array}$$
  

$$\begin{array}{r}
 0100 \\
 \times \quad +1+1 \\
 \hline
 00000100 \\
 000100 \\
 \hline
 00010100
 \end{array}
 \begin{array}{l}
 \text{step 3}
 \end{array}$$

Example:  $+4 * +5$

Steps:

- 1.) represent both multiplicand (m) and multiplier (n) using 2's complement format
- 2.) convert multiplier to Extended Booth's equivalent
- 3.) Multiply using pencil-and-paper method ignoring extra steps if multiplier is negative. Since each bit-pair is equivalent to 2 bits, skip two after the initial intermediate product

# Extended Booth's Algorithm

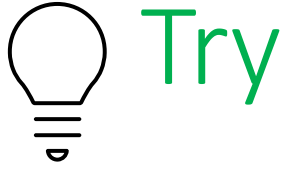
$$\begin{array}{r}
 0100 \\
 1011 \\
 \hline
 10110 \\
 -1-1 \\
 \hline
 \end{array}
 \begin{array}{l}
 \text{step 1} \\
 \text{step 2}
 \end{array}$$
  

$$\begin{array}{r}
 0100 \\
 \times \quad -1-1 \\
 \hline
 11111100 \\
 111100 \\
 \hline
 11101100
 \end{array}
 \begin{array}{l}
 \text{step 3}
 \end{array}$$

Example:  $+4 * -5$

Steps:

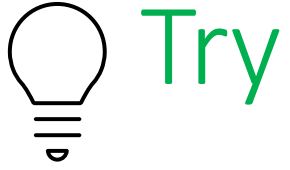
- 1.) represent both multiplicand (m) and multiplier (n) using 2's complement format
- 2.) convert multiplier to Extended Booth's equivalent
- 3.) Multiply using pencil-and-paper method ignoring extra steps if multiplier is negative. Since each bit-pair is equivalent to 2 bits, skip two after the initial intermediate product



Try:  $+13 \times -6$  (using Extended Booth's algorithm)

$$\begin{array}{r} 01101 \\ \times 11010 \\ \hline \end{array}$$





Try:  $+13 \times -6$  (using Extended Booth's algorithm)

```
  01101
x 11010
-----
```

```
  01101
  11010 } step 1
1110100 } step 2
0-1-2
```

```
      01101
x    0-1-2
-----
1111100110
11110011
000000
-----
1110110010
```

# To recall ...

- What have we learned:
  - ✓ Describe the process of performing integer binary multiplication using pencil-and-paper method
  - ✓ Describe the process of performing integer binary multiplication using Booth's algorithm
  - ✓ Describe the process of performing integer binary multiplication using extended Booth's algorithm