Modified Booth Multiplier

Booth Multiplier: an Introduction

- Recode each 1 in multiplier as "+2-1"
  - Converts sequences of 1 to 10…0(-1)
  - **Might** reduce the number of 1's

0	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**
						+1	**-1**		
					+1	**-1**			
				+1	**-1**				
			+1	**-1**					
		+1	**-1**						
	+1	**-1**							
0	**1**	**0**	**0**	**0**	**0**	**0**	**-1**	**0**	**0**

Booth Recoding: Multiplication Example

Sign extension

				0	0	1	1	0	6x
				0	1	1	1	0	14
				+1	0	0	-1	0	
				0	0	0	0	0	
1	1	1	1	1	0	1	0		(-6)
		0	0	0	0	0			
	0	0	0	0	0				
0	0	1	1	0					
0	0	1	0	1	0	1	0	0	84

Booth Recoding: Advantages and Disadvantages

- Depends on the architecture
  - Potential advantage: might reduce the # of 1's in multiplier
- In the multipliers that we have seen so far:
  - Doesn't save speed
    (still have to wait for the critical path, e.g., the shift-add delay in sequential multiplier)
  - Increases area: recoding circuitry AND subtraction

Modified Booth

- Booth 2 modified to produce at most n/2+1 partial products.
- **Algorithm: (for unsigned numbers)**
  1. Pad the LSB with one zero.
  2. Pad the MSB with 2 zeros if n is even and 1 zero if n is odd.
  3. Divide the multiplier into overlapping groups of 3-bits.
  4. Determine partial product scale factor from modified booth 2 encoding table.
  5. Compute the Multiplicand Multiples
  6. Sum Partial Products

Modified Booth Multiplier: Idea (cont.)

- Can encode the digits by looking at three bits at a time
- Booth recoding table:

i+1	**i**	**i-1**	**add**
0	**0**	**0**	**0*M**
0	**0**	**1**	**1*M**
0	**1**	**0**	**1*M**
0	**1**	**1**	**2*M**
1	**0**	**0**	**–2*M**
1	**0**	**1**	**–1*M**
1	**1**	**0**	**–1*M**
1	**1**	**1**	**0*M**

- Must be able to add *multiplicand* times –2, -1, 0, 1 and 2
- Since Booth recoding got rid of 3's, generating partial products is not that hard (shifting and negating)

[©Hauck]

Modified Booth

- **Example: (unsigned)**

1. **Pad LSB with 1 zero**

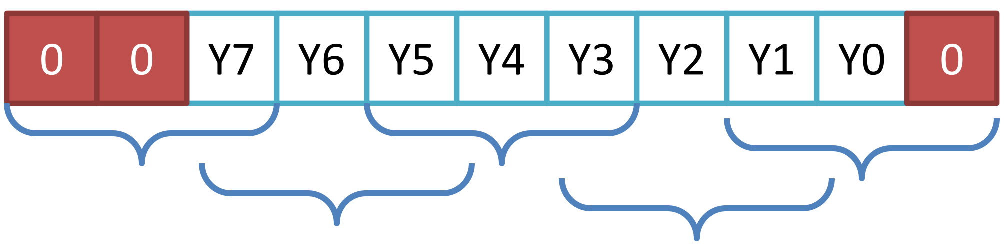
Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	0

2. **n is even then pad the MSB with two zeros**

0	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	0

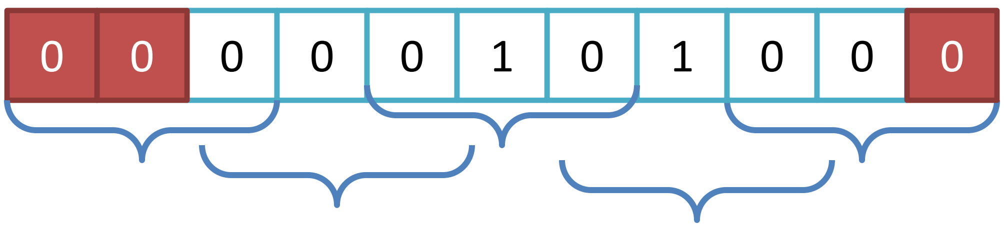
3. **Form 3-bit overlapping groups for n=8 we have 5 groups**

0	0	0	0	0	1	0	1	0	0	0



Modified Booth

4. Determine partial product scale factor from modified booth 2 encoding table.



Groups			Coding
0	0	0	$0 \times Y$
0	1	0	$1 \times Y$
0	1	0	$1 \times Y$
0	0	0	$0 \times Y$
0	0	0	$0 \times Y$

X_{i+1}	X_i	X_{i-1}	Action
0	0	0	$0 \times Y$
0	0	1	$1 \times Y$
0	1	0	$1 \times Y$
0	1	1	$2 \times Y$
1	0	0	$-2 \times Y$
1	0	1	$-1 \times Y$
1	1	0	$-1 \times Y$
1	1	1	$0 \times Y$

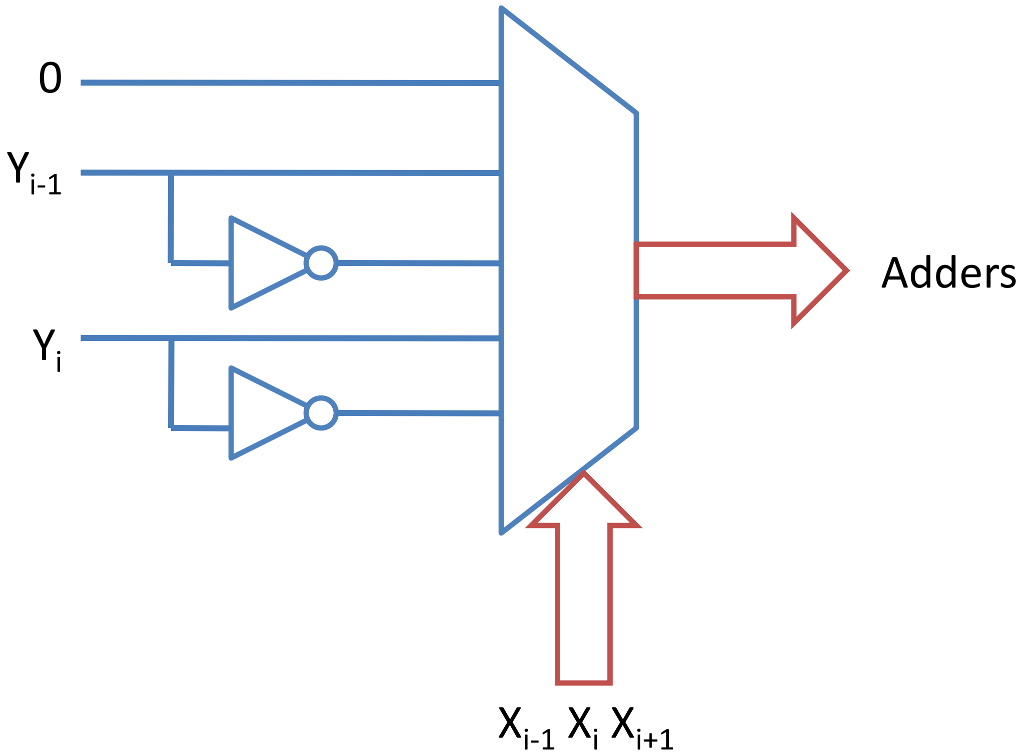
Modified Booth

5. Compute the Multiplicand Multiples

Groups			Coding
0	0	0	$0 \times Y$
0	1	0	$1 \times Y$
0	1	0	$1 \times Y$
0	0	0	$0 \times Y$
0	0	0	$0 \times Y$

```
                  0 0 0 0 0 1 0 0 0     1
                × 0 0 0 0 1 0 1 0 0    20
   ----------------------------------
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 × Y
   0 0 0 0 0 0 0 0 0 0 1 0 0 0        1 × Y
   0 0 0 0 0 0 0 0 1 0 0 0            1 × Y
   0 0 0 0 0 0 0 0 0 0                0 × Y
   0 0 0 0 0 0 0 0                    0 × Y
   ----------------------------------
```

Compute Partial Products



X_{i+1}	X_i	X_{i-1}	Action
0	0	0	$0 \times Y$
0	0	1	$1 \times Y$
0	1	0	$1 \times Y$
0	1	1	$2 \times Y$
1	0	0	$-2 \times Y$
1	0	1	$-1 \times Y$
1	1	0	$-1 \times Y$
1	1	1	$0 \times Y$

Modified Booth

6. Sum Partial Products

									0	0	0	0	0	1	0	0	0	1
								×	0	0	0	0	1	0	1	0	0	20
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0 × Y
	0	0	0	0	0	0	0	0	0	0	1	0	0	0			1 × Y
+	0	0	0	0	0	0	0	0	1	0	0	0					1 × Y
	0	0	0	0	0	0	0	0	0	0							0 × Y
	0	0	0	0	0	0	0	0									0 × Y
	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	160

Modified Booth

- Booth 2 modified to produce at most n/2+1 partial products.
- **Algorithm: (for unsigned numbers)**
  1. Pad the LSB with one zero.
  2. If n is even don't pad the MSB (n/2 PP's) and if n is odd sign extend the MSB by 1 bit (n+1/2 PP's).
  3. Divide the multiplier into overlapping groups of 3-bits.
  4. Determine partial product scale factor from modified booth 2 encoding table.
  5. Compute the Multiplicand Multiples
  6. Sum Partial Products

Modified Booth

- **Example: (n=4-bits unsigned)**

1. **Pad LSB with 1 zero**

Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	0

2. **n is even then do not pad the MSB**

Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	0

3. **Form 3-bit overlapping groups for n=8 we have 5 groups**

0	1	1	0	1	0	0	1	0

Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	0

Modified Booth

4. Determine partial product scale factor from modified booth 2 encoding table.

0	1	1	0	1	0	0	1	0

Groups			Coding
0	1	0	1 × Y
1	0	0	-2 × Y
1	0	1	-1 × Y
0	1	1	2 × Y

X_{i+1}	X_i	X_{i-1}	Action
0	0	0	0 × Y
0	0	1	1 × Y
0	1	0	1 × Y
0	1	1	2 × Y
1	0	0	-2 × Y
1	0	1	-1 × Y
1	1	0	-1 × Y
1	1	1	0 × Y

Modified Booth

5. Compute the Multiplicand Multiples

Groups			Coding
0	1	0	1 × Y
1	0	0	-2 × Y
1	0	1	-1 × Y
0	1	1	2 × Y

									1	0	0	1	0	1	0	1	-107
							×		0	1	1	0	1	0	0	1	105
	1	1	1	1	1	1	1	1	1	0	0	1	0	1	0	1	1 × Y
	0	0	0	0	0	0	1	1	0	1	0	1	1	0			-2 × Y
	0	0	0	0	0	1	1	0	1	0	1	1					-1 × Y
	0	1	0	0	1	0	1	0	1	0							2 × Y
	1	1	0	1	0	1	0	0	0	0	0	1	1	1	0	1	-11235

Modified Booth Multiplier: Idea (cont.)

- Interpretation of the Booth recoding table:

i+1	i	i-1	add	Explanation
0	0	0	0*M	No string of 1’s in sight
0	0	1	1*M	End of a string of 1’s
0	1	0	1*M	Isolated 1
0	1	1	2*M	End of a string of 1’s
1	0	0	–2*M	Beginning of a string of 1’s
1	0	1	–1*M	End one string, begin new one
1	1	0	–1*M	Beginning of a string of 1’s
1	1	1	0*M	Continuation of string of 1’s

[Par] p. 160

Modified Booth Recoding: Summary

- Grouping multiplier bits into pairs
  - Orthogonal idea to the Booth recoding
  - Reduces the num of partial products to half
  - If Booth recoding not used ➔ have to be able to multiply by 3 (hard: shift+add)
- Applying the grouping idea to Booth ➔ Modified Booth Recoding (Encoding)
  - We already got rid of sequences of 1's ➔ no mult by 3
  - Just negate, shift once or twice