CSE4210
Multiplication

CSE4210 Winter 2012

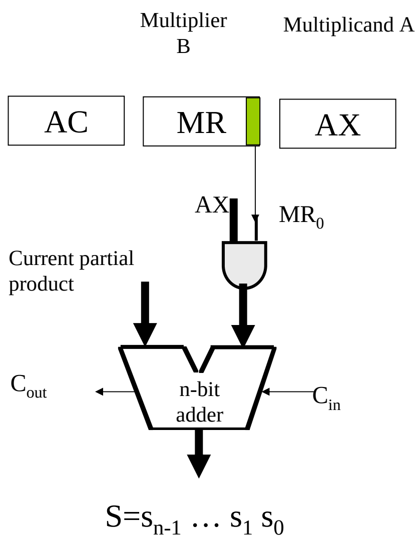
Mokhtar Aboelaze

YORK UNIVERSITY

Multiplication

- The simplest way of doing multiplication is repeated add and shift.
- Easy to understand, simple hardware, but not very fast

YORK UNIVERSITY



$$\mathbf{AX} \wedge MR_0 = \begin{cases} \mathbf{AX} & \text{if } MR_0 = 1 \\ 0 & \text{if } MR_0 = 0 \end{cases}$$

$$C_{out} \cdot \mathbf{S} \leftarrow (\mathbf{AC}) + (\mathbf{AX} \wedge MR_0)$$

$$\mathbf{AC} \cdot \mathbf{MR} \leftarrow C_{out} \cdot \mathbf{S} \cdot \mathbf{MR}_{[n-1 \cdots 1]}$$

YORK UNIVERSITY

Multiplication

```
        a   1 0 1 0
        x   1 0 1 1

x0a         1 0 1 0
x1a       1 0 1 0
x2a     0 0 0 0
x3a   1 0 1 0
      ---------------
    0 1 1 0 1 1 1 0
```

Multiplication

- Right shift

Multiplication by 2k aligns the number to the high order bits

$$p^{(j+1)} = \left(p^{(j)} + x_j a 2^k\right)2^{-1} \quad \text{with} \quad p^{(0)} = 0 \quad \text{and} \quad p^{(k)} = p$$

- Left shift

$$p^{(j+1)} = 2p^{(j)} + x_{k-j-1}a \quad \text{with} \quad p^{(0)} = 0 \quad \text{and} \quad p^{(k)} = p$$

a		1 0 1 0	
x		1 0 1 1	
==============			
$p^{(0)}$		0 0 0 0	
$+x_0a$		1 0 1 0	

$2p^{(1)}$	0	1 0 1 0	
$p^{(1)}$		0 1 0 1	0
$+x_1a$		1 0 1 0	

$2p^{(2)}$	0	1 1 1 1	0
$P^{(2)}$		0 1 1 1	1 0
$+x_2a$		0 0 0 0	

$2p^{(3)}$	0	0 1 1 1	1 0
$P^{(3)}$		0 0 1 1	1 1 0
$+x_3a$		1 0 1 0	

$2p^{(4)}$	0	1 1 0 1	1 1 0
$P^{(4)}$		0 1 1 0	1 1 1 0

a		1 0 1 0
x		1 0 1 1
==============		
$p^{(0)}$		0 0 0 0
$2p^{(0)}$	0	0 0 0 0
$+x_3a$		1 0 1 0

$p^{(1)}$	0	1 0 1 0
$2p^{(1)}$	0 1	0 1 0 0
$+x_2a$		0 0 0 0

$p^{(2)}$	0 1	0 1 0 0
$2P^{(2)}$	0 1 0	1 0 0 0
$+x_1a$		1 0 1 0

$p^{(3)}$	0 1 1	0 0 1 0
$2P^{(3)}$	0 1 1 0	0 1 0 0
$+x_0a$		1 0 1 0

$p^{(4)}$	0 1 1 0	1 1 1 0

YORK UNIVERSITY

Multiplication of Signed Numbers

- Right shift the partial sum
- If the multiplier is positive (-ve multiplicand), then the algorithm will work fine
  - Each $x_j a$ is a 2's complement number and the sum works correctly if we sign extended the partial sum
- If the multiplier is negative, then the negative-weight interpretation of the sign bit can be handled correctly if $x_{k-1}a$ is subtracted instead of added

YORK UNIVERSITY

Example

1 0 1 1 0
0 1 0 1 1

1 0 1 1 0
1 0 1 0 1

YORK UNIVERSITY

More than one-bit at a time

- What we did so far is inspecting the multiplier bit by bit and either adding the multiplicand or 0.
- We can do this by inspecting more than one bit (digit) at a time.
- If we inspect 2 bits, then we can add 0, M, 2M, or 3M at a time, and reduce the number of additions by half.
- The problem is with the 3M (could be represented as 2M+M.

YORK UNIVERSITY

Example

3= 1+2

1 0 1 0
1 0 1 0 0
1 1 1 1 0

10 — 1 0 1 0

13 — 1 1 0 1

M: 1 0 1 0

3M: 1 1 1 1 0

130 — 1 0 0 0 0 0 1 0

Can use CSA for multi operand additions (See the previous lecture)

YORK UNIVERSITY

Booth Encoding

- The basic idea is that a 1 can be represented as 2-1.
- That eliminates a sequence 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
```

YORK UNIVERSITY

Booth encoding

- Starting from right to left, if we encounter a sequence of 1's The first 1 is replaced by –1, the first 0 (after the sequence) is replaced by 1.
- Sequence of 0's means shift.
- Adding ±M (-M is the 1's complement of M with c_{in}=1).
- The number of additions varies.

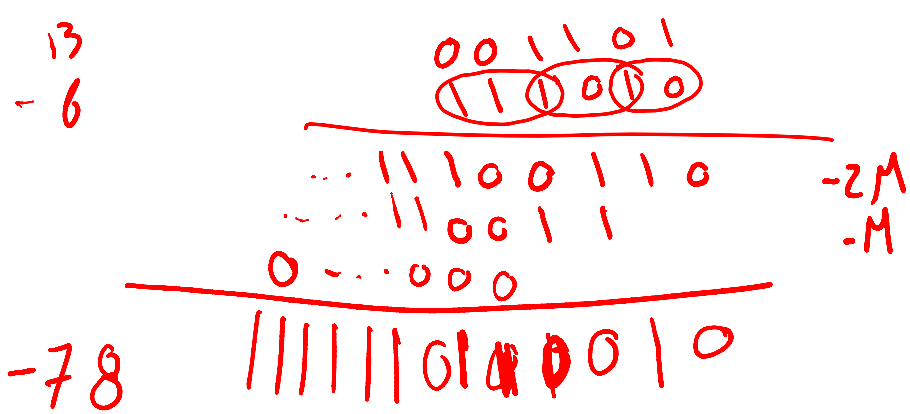
Modified Booth Encoding

- Can look at 3 bits with overlap.
- Eliminates the need to have 3M (only M,2M).
- 2M is M with left shift.

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

YORK UNIVERSITY

Example



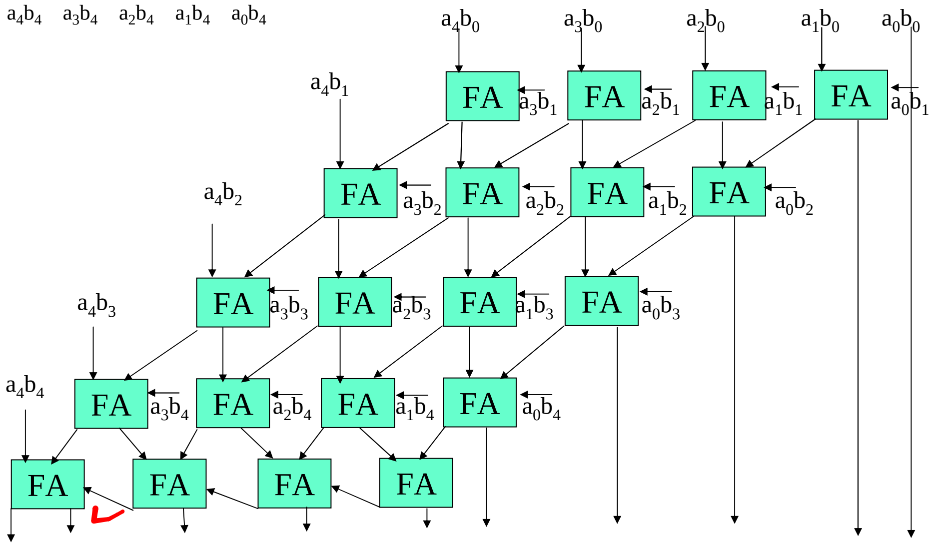
YORK UNIVERSITY

Example

YORK UNIVERSITY

Array Multiplier

				a_4b_0	a_3b_0	a_2b_0	a_1b_0	a_0b_0
			a_4b_1	a_3b_1	a_2b_1	a_1b_1	a_0b_1	
		a_4b_2	a_3b_2	a_2b_2	a_1b_2	a_0b_2		
	a_4b_3	a_3b_3	a_2b_3	a_1b_3	a_0b_3			
a_4b_4	a_3b_4	a_2b_4	a_1b_4	a_0b_4				



Implementing Large Multipliers using Smaller ones