# CSE4210 Architecture and Hardware for DSP

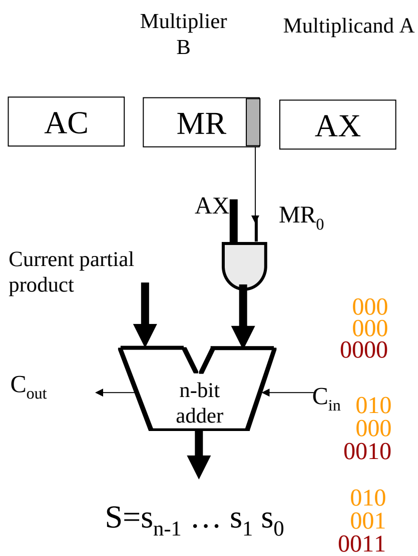
Lecture 3

Multiplication



## Multiplication

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




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

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

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

| AC | MR | AX |
|---|---|---|
| 0 0 0 | 1 1 **0** | 0 1 0 |
| 0 0 0 | 0 1 **1** | 0 1 0 |
| 0 0 1 | 0 0 **1** | 0 1 0 |
| 0 0 1 | 1 0 0 | 0 1 0 |



# 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
```

**York University** **CSE**

# Multiplication

- Right shift

Multiplication by $2^k$ 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

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

**York University** **CSE**

| | |
|---|---|
| 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 CSE

```
a          1 0 1 0
x          1 0 1 1
===============
p(0)       0 0 0 0
+x0a       1 0 1 0
---------------------
2p(1)  0   1 0 1 0
p(1)       0 1 0 1   0
+x1a       1 0 1 0
---------------------------
2p(2)  0   1 1 1 1    0
P(2)       0 1 1 1    1 0
+x2a       0 0 0 0
--------------------------------
2p(3)  0   0 1 1 1    1 0
P(3)       0 0 1 1    1 1 0
+x3a       1 0 1 0
--------------------------------------
2p(4)  0   1 1 0 1    1 1 0
P(4)       0 1 1 0    1 1 1 0
```

$$p^{(j+1)} = \left(p^{(j)} + x_j a 2^k\right) 2^{-1}$$

with $p^{(0)} = 0$ and $p^{(k)} = p$

$$\boxed{\begin{array}{ll} a & 1\ 0\ 1\ 0 \\ x & 1\ 0\ 1\ 1 \end{array}}$$

```
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
```

York University CSE

```
a                    1 0 1 0
x                    1 0 1 1
===============
p(0)                 0 0 0 0
2p(0)            0   0 0 0 0
+x3a                 1 0 1 0
----------------------------------
p(1)             0   1 0 1 0
2p(1)          0 1   0 1 0 0
+x2a                 0 0 0 0
--------------------------------
p(2)           0 1   0 1 0 0
2P(2)        0 1 0   1 0 0 0
+x1a                 1 0 1 0
----------------------------------
p(3)         0 1 1   0 0 1 0
2P(3)      0 1 1 0   0 1 0 0
+x0a                 1 0 1 0
--------------------------------------
p(4)       0 1 1 0   1 1 1 0
```

$$p^{(j+1)} = 2p^{(j)} + x_{k-j-1}a$$

with $p^{(0)} = 0$ and $p^{(k)} = p$

$$\boxed{\begin{array}{ll} a & 1\ 0\ 1\ 0 \\ x & 1\ 0\ 1\ 1 \end{array}}$$

```
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 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



# Example

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 |

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |

# 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.



# Example

```
10              1 0 1 0              3M
13              1 1 0 1
              ---------
                1 0 1 0        M
        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)

# 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
```



# 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 (the sequence is replaced by 0's).
- 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.

York University CSE

# Booth Multiplier

| $a_i$ | $a_{i-1}$ | |
|---|---|---|
| 0 | 0 | Shift |
| 1 | 1 | Shift |
| 1 | 0 | -M |
| 0 | 1 | +M |

York University CSE

# Modified Booth Encoding

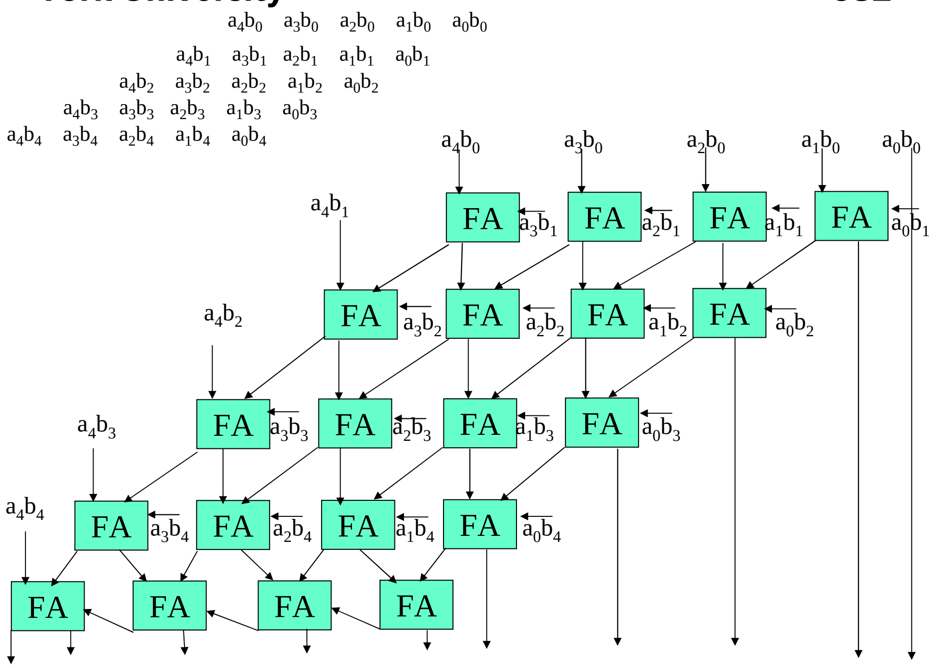
- Can look at 3 bits with overlap.
- Eliminates the need to have 3M (only M,2M).
- 2M is M with left shift.

| 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 |

# Example



# Example




York University CSE

# Implementing Large Multipliers using Smaller ones