

Lecture 8

Sequential Multipliers

Notation

a Multiplicand $a_{k-1}a_{k-2} \dots a_1 a_0$
x Multiplier $x_{k-1}x_{k-2} \dots x_1 x_0$
p Product ($a \cdot x$) $p_{2k-1}p_{2k-2} \dots p_2 p_1 p_0$

Multiplication of two 4-bit unsigned binary numbers in dot notation

| | | |
|---|-----------------|-------------|
| | • • • • | a |
| x | • • • • | x |
| | ----- | |
| | • • • • | $x_0 a 2^0$ |
| | • • • • | $x_1 a 2^1$ |
| | • • • • | $x_2 a 2^2$ |
| | • • • • | $x_3 a 2^3$ |
| | ----- | |
| | • • • • • • • • | p |

Basic Multiplication Equations

$$p = a \cdot x \quad x = \sum_{i=0}^{k-1} x_i \cdot 2^i$$

$$p = a \cdot x = \sum_{i=0}^{k-1} a \cdot x_i \cdot 2^i =$$

$$= x_0 a 2^0 + x_1 a 2^1 + x_2 a 2^2 + \dots + x_{k-1} a 2^{k-1}$$

Shift/Add Algorithms
Right-shift algorithm

$$p = a \cdot x = x_0 a 2^0 + x_1 a 2^1 + x_2 a 2^2 + \dots + x_{k-1} a 2^{k-1} =$$

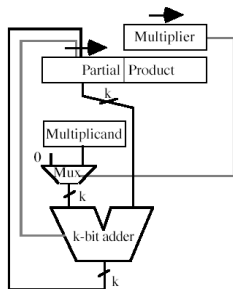
$$= \underbrace{(\dots((0 + x_0 a 2^k) / 2 + x_1 a 2^k) / 2 + \dots + x_{k-1} a 2^k) / 2}_{k \text{ times}} =$$

$$p^{(0)} = 0$$

$$p^{(j+1)} = (p^{(j)} + x_j a 2^k) / 2 \quad j=0..k-1$$

$$p = p^{(k)}$$

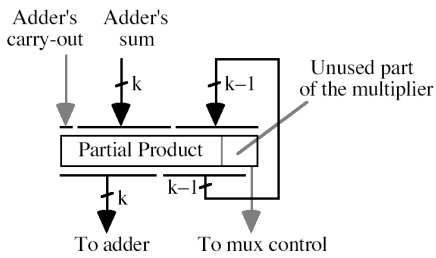
Sequential shift-and-add multiplier for right-shift algorithm



Right-shift multiplication algorithm: Example

| Right-shift algorithm | |
|-----------------------|-----------------|
| 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 |

Area optimization for the sequential shift-and-add multiplier with the right-shift algorithm



**Shift/Add Algorithms
Left-shift algorithm**

$$p = a \cdot x = x_0a2^0 + x_1a2^1 + x_2a2^2 + \dots + x_{k-1}a2^{k-1} =$$

$$= (\dots((0 \cdot 2 + x_{k-1}a) \cdot 2 + x_{k-2}a) \cdot 2 + \dots + x_1a) \cdot 2 + x_0a =$$

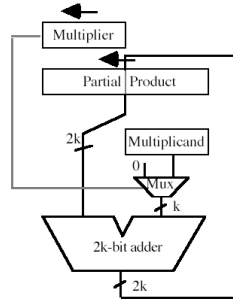
k times

$$p^{(0)} = 0$$

$$p^{(j+1)} = (p^{(j)} \cdot 2 + x_{k-1-j}a) \quad j=0..k-1$$

$$p = p^{(k)}$$

Sequential shift-and-add multiplier for left-shift algorithm



Left-shift multiplication algorithm: Example

Left-shift algorithm

| | |
|------------|-----------------|
| 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 |
| ----- | |

Shift/Add Algorithms
Right-shift algorithm: multiply-add

$$p^{(0)} = y2^k$$

$$p^{(j+1)} = (p^{(j)} + x_j a 2^k) / 2 \quad j=0..k-1$$

$$p = p^{(k)}$$

$$= \underbrace{(\dots((y2^k + x_0a2^k)/2 + x_1a2^k)/2 + \dots + x_{k-1}a2^k)/2}_{k \text{ times}}$$

$$= y + x_0a2^0 + x_1a2^1 + x_2a2^2 + \dots + x_{k-1}a2^{k-1} = y + a \cdot x$$

Shift/Add Algorithms
Left-shift algorithm: multiply-add

$$p^{(0)} = y2^{-k}$$

$$p^{(j+1)} = (p^{(j)} \cdot 2 + x_{k-(j+1)}a) \quad j=0..k-1$$

$$p = p^{(k)}$$

$$= (\dots((y2^{-k} \cdot 2 + x_{k-1}a) \cdot 2 + x_{k-2}a) \cdot 2 + \dots + x_1a) \cdot 2 + x_0a =$$

k times

$$= y + x_{k-1}a2^{k-1} + x_{k-2}a2^{k-2} + \dots + x_1a2^1 + x_0a = y + a \cdot x$$

Sequential multiplication of 2's-complement numbers with right shifts (positive multiplier)

| | |
|-------------------|---------------------|
| ===== | |
| a | 1 0 1 1 0 |
| x | 0 1 0 1 1 |
| ===== | |
| p ⁽⁰⁾ | 0 0 0 0 0 |
| +x ₀ a | 1 0 1 1 0 |
| ----- | |
| 2p ⁽¹⁾ | 1 1 0 1 1 0 |
| p ⁽¹⁾ | 1 1 0 1 1 0 |
| +x ₁ a | 1 0 1 1 0 |
| ----- | |
| 2p ⁽²⁾ | 1 1 0 0 0 1 0 |
| p ⁽²⁾ | 1 1 0 0 0 1 0 |
| +x ₂ a | 0 0 0 0 0 |
| ----- | |
| 2p ⁽³⁾ | 1 1 1 0 0 0 1 0 |
| p ⁽³⁾ | 1 1 1 0 0 0 1 0 |
| +x ₃ a | 1 0 1 1 0 |
| ----- | |
| 2p ⁽⁴⁾ | 1 1 0 0 1 0 0 1 0 |
| p ⁽⁴⁾ | 1 1 0 0 1 0 0 1 0 |
| +x ₄ a | 0 0 0 0 0 |
| ----- | |
| 2p ⁽⁵⁾ | 1 1 1 0 0 1 0 0 1 0 |
| p ⁽⁵⁾ | 1 1 1 0 0 1 0 0 1 0 |
| ===== | |

Sequential multiplication of 2's-complement numbers with right shifts (negative multiplier)

| | |
|----------------------|---------------------|
| ===== | |
| a | 1 0 1 1 0 |
| x | 1 0 1 0 1 |
| ===== | |
| p ⁽⁰⁾ | 0 0 0 0 0 |
| +x ₀ a | 1 0 1 1 0 |
| ----- | |
| 2p ⁽¹⁾ | 1 1 0 1 1 0 |
| p ⁽¹⁾ | 1 1 0 1 1 0 |
| +x ₁ a | 0 0 0 0 0 |
| ----- | |
| 2p ⁽²⁾ | 1 1 1 0 1 1 0 |
| p ⁽²⁾ | 1 1 1 0 1 1 0 |
| +x ₂ a | 1 0 1 1 0 |
| ----- | |
| 2p ⁽³⁾ | 1 1 0 0 1 1 1 0 |
| p ⁽³⁾ | 1 1 0 0 1 1 1 0 |
| +x ₃ a | 0 0 0 0 0 |
| ----- | |
| 2p ⁽⁴⁾ | 1 1 1 0 0 1 1 1 0 |
| p ⁽⁴⁾ | 1 1 1 0 0 1 1 1 0 |
| +(-x ₄ a) | 0 1 0 1 0 |
| ----- | |
| 2p ⁽⁵⁾ | 0 0 0 1 1 0 1 1 1 0 |
| p ⁽⁵⁾ | 0 0 0 1 1 0 1 1 1 0 |
| ===== | |

Basic Multiplication Equations

$$p = a \cdot x \quad x = \sum_{i=0}^{k-1} x_i \cdot r^i$$

$$p = a \cdot x = \sum_{i=0}^{k-1} a \cdot x_i \cdot r^i =$$

$$= x_0 a r^0 + x_1 a r^1 + x_2 a r^2 + \dots + x_{k-1} a r^{k-1}$$

High-Radix Shift/Add Algorithms
Right-shift high-radix algorithm

$$p = a \cdot x = x_0 a r^0 + x_1 a r^1 + x_2 a r^2 + \dots + x_{k-1} a r^{k-1} =$$

$$= (\dots((0 + \underbrace{x_0 a r^k / r + x_1 a r^k / r + \dots + x_{k-1} a r^k / r}_{k \text{ times}})) / r =$$

$$p^{(0)} = 0$$

$$p^{(j+1)} = (p^{(j)} + x_j a r^k) / r \quad j=0..k-1$$

$$p = p^{(k)}$$

High-Radix Shift/Add Algorithms
Left-shift high-radix algorithm

$$p = a \cdot x = x_0 a r^0 + x_1 a r^1 + x_2 a r^2 + \dots + x_{k-1} a r^{k-1} =$$

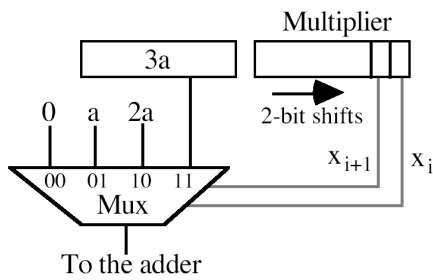
$$= (\dots((0 \cdot r + \underbrace{x_{k-1} a \cdot r + x_{k-2} a \cdot r + \dots + x_1 a \cdot r + x_0 a}_{k \text{ times}})) \cdot r =$$

$$p^{(0)} = 0$$

$$p^{(j+1)} = (p^{(j)} \cdot r + x_{k-1-j} a) \quad j=0..k-1$$

$$p = p^{(k)}$$

The multiple generation part of a radix-4 multiplier with precomputation of $3a$

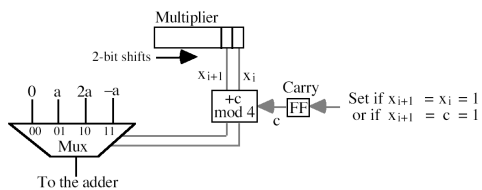


Example of radix-4 multiplication using the $3a$ multiple

```

=====
a           0 1 1 0
3a          0 1 0 0 1 0
x           1 1 1 0
=====
p(0)        0 0 0 0
+(x1x0)twoa  0 0 1 1 0 0
-----
4p(1)       0 0 1 1 0 0
p(1)        0 0 1 1 0 0
+(x3x2)twoa  0 1 0 0 1 0
-----
4p(2)       0 1 0 1 0 1 0 0
p(2)        0 1 0 1 0 1 0 0
=====
    
```

The multiple generation part of a radix-4 multiplier based on replacing $3a$ with $4a$ (carry into next higher radix-4 multiplier digit) and $-a$



Radix-4 Booth Recoding

| | | | | |
|---------------|----------|-----------|----------|---------------------------------|
| 1 0 0 1 | 1 1 0 1 | 1 0 1 0 | 1 1 1 0 | Operand x |
| (1) -1 0 1 0 | 0 -1 1 0 | -1 1 -1 1 | 0 0 -1 0 | |
| (1) $\cdot 2$ | 2 | $\cdot 1$ | 2 | $\cdot 1$ |
| | | $\cdot 1$ | | 0 |
| | | | | $\cdot 2$ |
| | | | | Recoded radix-4 version z |

Table 10.1 Radix-4 Booth's recoding yielding $(z_{k/2} \dots z_1 z_0)_{four}$

| x_{i+1} | x_i | x_{i-1} | y_{i+1} | y_i | $z_{i/2}$ | Explanation |
|-----------|-------|-----------|-----------|-----------|-----------|-----------------------------------|
| 0 | 0 | 0 | 0 | 0 | 0 | No string of 1s in sight |
| 0 | 0 | 1 | 0 | 1 | 1 | End of a string of 1s in x |
| 0 | 1 | 0 | 0 | 1 | 1 | Isolated 1 in x |
| 0 | 1 | 1 | 1 | 0 | 2 | End of a string of 1s in x |
| 1 | 0 | 0 | $\cdot 1$ | 0 | $\cdot 2$ | Beginning of a string of 1s in x |
| 1 | 0 | 1 | $\cdot 1$ | 1 | $\cdot 1$ | End one string, begin new one |
| 1 | 1 | 0 | 0 | $\cdot 1$ | $\cdot 1$ | Beginning of a string of 1s in x |
| 1 | 1 | 1 | 0 | 0 | 0 | Continuation of string of 1s in x |

| | | | | |
|---------------|---------|-----------|---------|---------------------------------|
| 1 0 0 1 | 1 1 0 1 | 1 0 1 0 | 1 1 1 0 | Operand x |
| (1) $\cdot 2$ | 2 | $\cdot 1$ | 2 | $\cdot 1$ |
| | | $\cdot 1$ | | 0 |
| | | | | $\cdot 2$ |
| | | | | Recoded radix-4 version z |

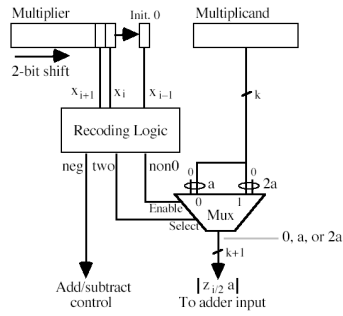
Example radix-4 multiplication with modified Booth's recoding of the 2's-complement multiplier

```

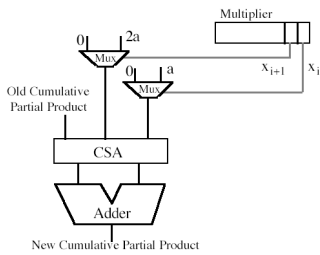
=====
a          0 1 1 0
x          1 0 1 0
z           $\cdot 1$   $\cdot 2$  Recoded version of x
=====
p(0)       0 0 0 0 0 0
+z0a     1 1 0 1 0 0
-----
4p(1)     1 1 0 1 0 0
p(1)      1 1 1 1 0 1 0 0
+z1a     1 1 1 0 1 0
-----
4p(2)     1 1 0 1 1 1 0 0
p(2)      1 1 0 1 1 1 0 0
=====

```

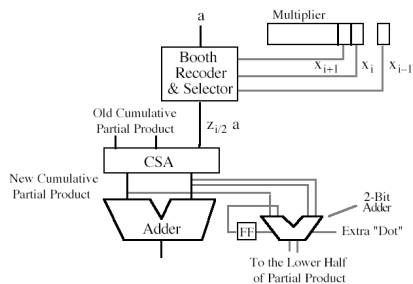
The multiple generation part of a radix-4 multiplier based on Booth's recoding



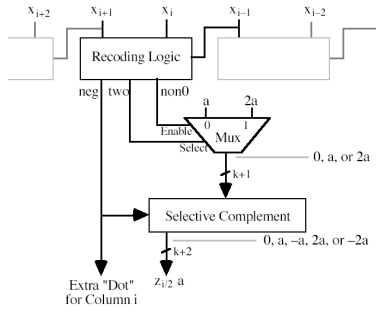
Radix-4 multiplication with a carry-save adder used to combine the cumulative partial product, $x_i a$, and $2x_{i+1} a$ into two numbers



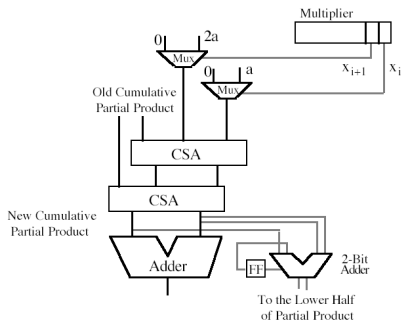
Radix-4 multiplier with a carry-save adder and Booth's recoding



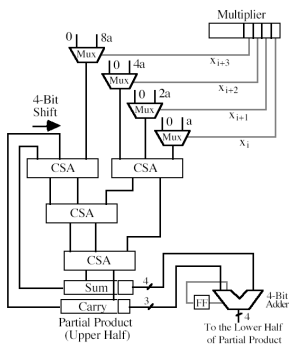
Booth recoding and multiple selection logic for high-radix multiplication



Radix-4 multiplier with two carry-save adders



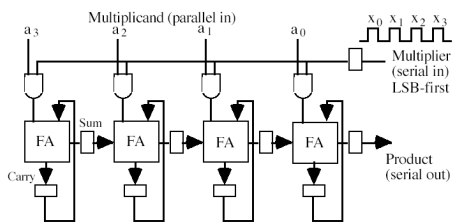
Radix-16 multiplier with carry-save adders



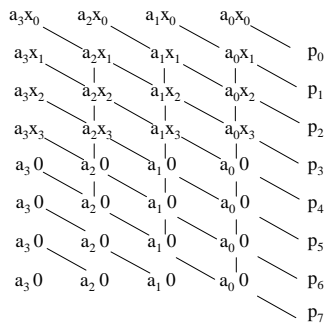
Bit Serial Multipliers Advantages

- small area
- reduced pin count
- reduced wire length
- high clock rate

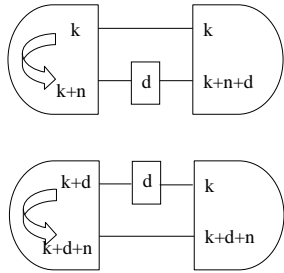
Semisystolic Bit-Serial Multiplier (1)



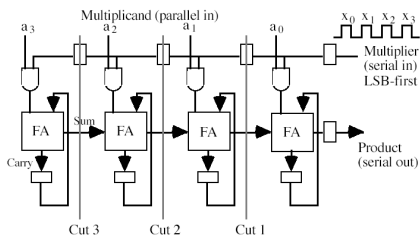
Semisystolic Bit-Serial Multiplier (2)



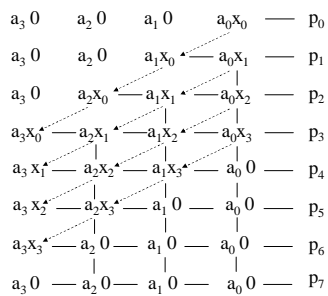
Retiming



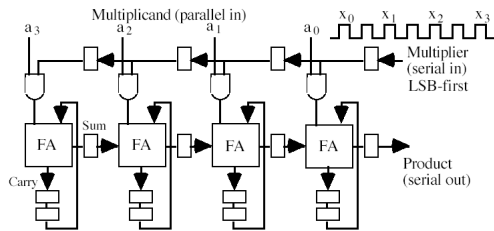
Retimed Semisystolic Bit-Serial Multiplier (1)



Retimed Semisystolic Bit-Serial Multiplier (2)

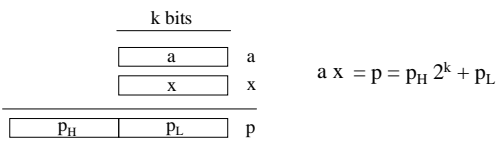


Systolic Bit-Serial Multiplier



Modular Multiplication

Special Cases



$$a x \bmod 2^k = p_L$$

$$a x \bmod 2^{k-1} = p_L + p_H + \text{carry}$$

$$a x \bmod 2^{k+1} = p_L - p_H - \text{borrow}$$

Modular Multiplication

Special Case (1)

$$\begin{aligned}
 a x \bmod 2^{k-1} &= (p_H 2^k + p_L) \bmod (2^k - 1) = \\
 &= (p_H (2^k \bmod 2^{k-1}) + p_L) \bmod (2^k - 1) = \\
 &= p_H + p_L \bmod (2^k - 1) = \\
 &= \begin{cases} p_H + p_L & \text{if } p_H + p_L < 2^k - 1 \\ p_H + p_L - (2^k - 1) & \text{if } p_H + p_L \geq 2^k - 1 \end{cases} \\
 &= p_L + p_H + \text{carry} \\
 &\quad \text{carry} = \text{carry from addition } p_L + p_H
 \end{aligned}$$

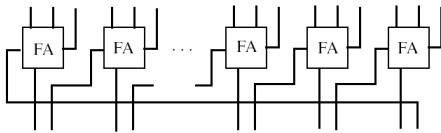
Modular Multiplication

Special Case (2)

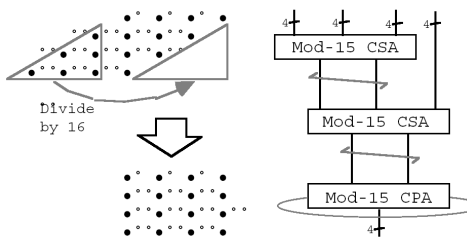
$$\begin{aligned}
 a \times \text{mod } 2^{k+1} &= (p_H 2^k + p_L) \text{ mod } (2^{k+1}) = \\
 &= (p_H(2^{k+1}-1) + p_L) \text{ mod } (2^{k+1}) = \\
 &= p_L - p_H \text{ mod } (2^{k+1}) = \\
 &= \begin{cases} p_L - p_H & \text{if } p_L - p_H \geq 0 \\ p_L - p_H + (2^{k+1}) & \text{if } p_L - p_H < 0 \end{cases} \\
 &= p_L - p_H + \text{borrow}
 \end{aligned}$$

borrow = borrow from subtraction $p_L + p_H$

Modulo (2^b-1) Carry Save Adder



4 x 4 Modulo 15 Multiplier



4 x 4 Modulo 13 Multiplier

