

# lecture 7

## sequential circuits 3:

- integer multiplication & division,
- floating point  $+$ ,  $-$ ,  $*$ ,  $/$

## Integer Multiplication

$$\begin{array}{r}
 352 \\
 \times 964 \\
 \hline
 1408 \\
 2112 \\
 3168 \\
 \hline
 339328
 \end{array}$$

## Unsigned Integer Multiplication (how to do it in binary?)

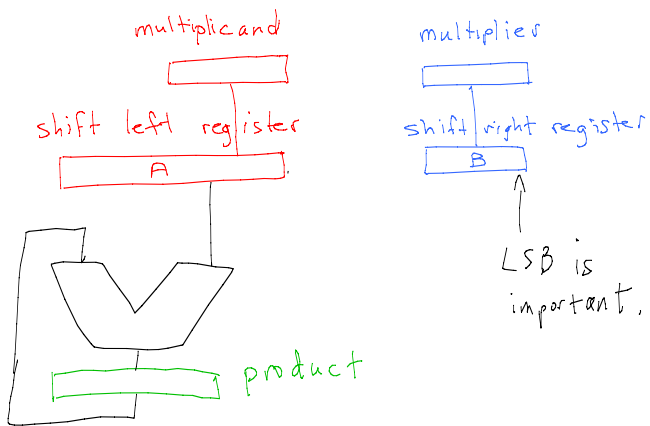
$$\begin{array}{r}
 A_{n-1} \dots A_2 A_1 A_0 \quad \text{multiplicand} \\
 \times B_{n-1} \dots B_2 B_1 B_0 \quad \text{multiplier} \\
 \hline
 P_{2n-1} \dots P_n P_{n-1} P_2 P_1 P_0 \quad \text{product}
 \end{array}$$

Note:  $(2^n - 1)(2^n - 1) < 2^{2n} - 1$

$$\begin{array}{r}
 01001101 \quad \text{multiplicand} \\
 * 00010111 \quad \text{multiplier} \\
 \hline
 01001101 \\
 01001101 \\
 01001101 \\
 00000000 \\
 01001101 \\
 00000000 \\
 00000000 \\
 \hline
 00011011101011 \quad \text{product}
 \end{array}$$

Alternative approach?

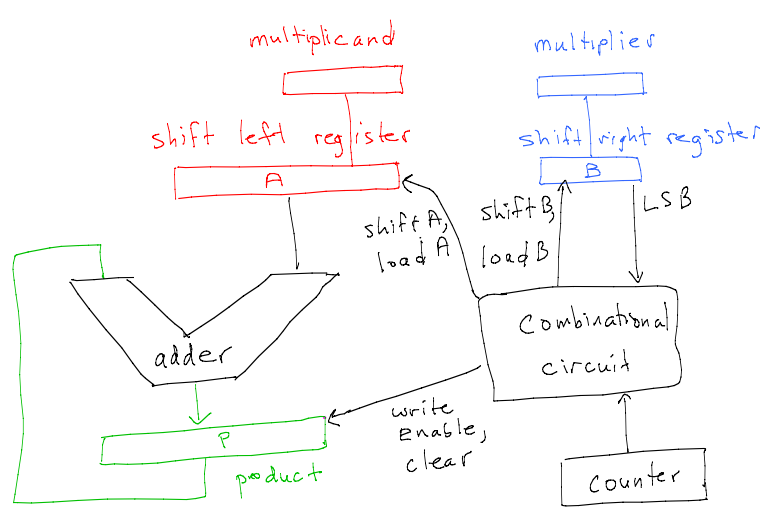
$$\begin{array}{r}
 01001101 \\
 * 00010111 \\
 \hline
 \end{array}$$



## Multiplication Algorithm

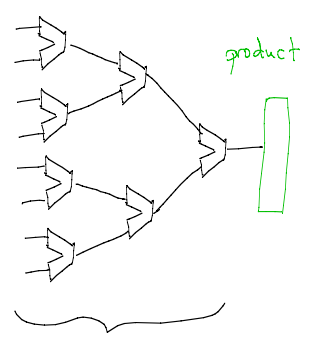
load **multiplicand** into lower  $n$  bits of  $2n$  bit register **A**  
 load **multiplier** into  $n$  bit register **B**  
 clear  $64$  bit **product** register **P**  
 for counter = 1 to  $n$  {  
   if LSB of **B** is 1  
     **P** = **P** + **A**  
   Shift **A** left (one bit)  
   Shift **B** right (one bit)  
 }

} in parallel

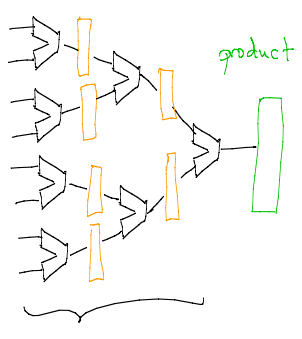


### Fast integer multiplication (sketch)

Use big and fast adders.



One clock cycle.



Use registers.  
Take several clock cycles.  
Why?

### Long Division

$$\begin{array}{r}
 785 \leftarrow \text{quotient} \\
 53 \uparrow \text{divisor} \quad | \quad 41627 \leftarrow \text{dividend} \\
 \underline{371} \\
 452 \\
 \underline{424} \\
 287 \\
 \underline{265} \\
 22 \leftarrow \text{remainder}
 \end{array}$$

How would you write out the algorithm?

$$\begin{array}{r}
 700 \\
 53 \overline{) 41627} \\
 \underline{37100} \\
 4527
 \end{array}
 \qquad
 \begin{array}{r}
 80 \\
 700 \\
 53 \overline{) 41627} \\
 \underline{37100} \\
 4527 \\
 \underline{4240} \\
 287
 \end{array}
 \qquad
 \begin{array}{r}
 80 \\
 80 \\
 700 \\
 53 \overline{) 41627} \\
 \underline{37100} \\
 4527 \\
 \underline{4240} \\
 287 \\
 \underline{265} \\
 22
 \end{array}$$

Revisit this grade school algorithm on your own and see if you really understand it.

$$\begin{array}{r}
 785 \\
 \text{divisor } 110101 \quad | \quad 101000101001001 \text{ dividend} \\
 \underline{110101} \\
 111000 \\
 \underline{110101} \\
 111001 \\
 \underline{110101} \\
 1001011 \\
 \underline{110101} \\
 10110 \\
 \underline{\quad\quad} \\
 22
 \end{array}$$

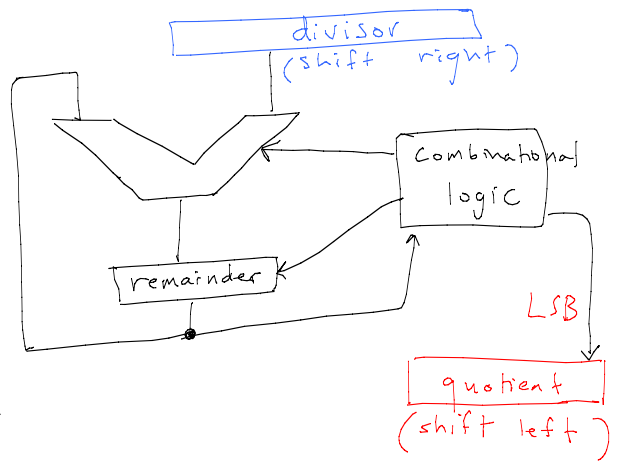
To perform subtractions, either use two's complement or use grade school subtraction (base 2).  
CAREFUL!

Algorithm for Long Division  
(note: divisor < dividend)

```

divisor = divisor * 2^n
quotient = 0
remainder = dividend
for i = 1 to n {
  shift quotient left by one bit
  if (divisor <= remainder) {
    set LSB of quotient to 1
    remainder = remainder - divisor
  }
  shift divisor right
}
  
```

Sketch (ignore register initialization)



Fast integer division ?

RST algorithm (1950s)  
gives some speedup

(details omitted - its complicated)  
but its still slower than  
multiplication

Floating Point Addition  
(assuming positive numbers)

$$\begin{aligned}
 x &= 1.101010000000000000101010 \times 2^{-3} \\
 y &= 1.00100100010000010100001 \times 2^2 \\
 & \quad x + y \quad ? \\
 x &= .0000 | 101010000000000000101010 \times 2^2
 \end{aligned}$$

$$x + y = ?$$

$$\begin{array}{r}
 x = 0.0000 | 101010000000000000101010 \times 2^2 \\
 y = 1.00100100010000010100001 \times 2^2 \\
 \hline
 1.001100010000001010000101010 \times 2^2
 \end{array}$$

↑  
28 bits significant

How to accomplish this? (Sketch only)

We need:

- compare exponents
- shift significant right (number with smaller exponent)
- big adder
- normalize (shift)
- round off

## Floating point addition ( $x > 0, y < 0$ )

Represent negative non-integer using two's complement as follows:

$$y = \overleftarrow{0001.01001} \times 2^e$$

$$-y = \overleftarrow{1110.10111} \times 2^e$$

eg.  $x = 26.5$   
 $y = -8.375$

$$\begin{array}{r} 1.1010100 \times 2^4 \\ - 0.1000011 \times 2^4 \\ \hline \end{array}$$

write using two's complement

$$\begin{array}{r} 001.1010100 \times 2^4 \\ + 111.0111101 \times 2^4 \\ \hline \end{array}$$

$$\Rightarrow x + y = 10010.001 = 18.125$$

## Floating point subtraction?

$$x - y = x + (-y)$$

## Floating point multiplication

$$\begin{array}{r} 1. \text{---} \times 2^{e_x} \\ * 1. \text{---} \times 2^{e_y} \\ \hline 1. \text{---} \times 2^{e_x + e_y} \end{array}$$

Similar to integer multiplication (but must take care of exponents too including handling overflow and underflow)

## Floating point division

$$\frac{x}{y} = \frac{5146.8954}{26.721}$$

$$= \frac{51468954}{26721} \times \frac{10^{-4}}{10^{-3}}$$

$$26721 \overline{) 51468954.0000 \dots}$$

Similar to integer division