## One's Complement

Negating = invert each bit from
$0 \to 1$ & $1 \to 0$

$= 2^n - X - 1$

## Two's Complement

Negating = invert each bit from
$0 \to 1$ & $1 \to 0$ and then
add 1 to result.

## Overflow

Adding two 32-bit numbers can yield
a result that needs 33-bit to
fully expressed,
Lack of 33rd bit ⇒ when overflow
occurs, sign bit is set with value
of the result instead of sign of
result

## 4 Cases

1. Add two "+" numbers
2. Add two "−" numbers
3. subtract a "−" from "+" number
4. subtract a "+" from "−" number

## Slide 6

* No overflow of adding a positive and a negative number.

(overflow term is misleading ⇒ not mean a carry "overflowed")

$$
\begin{array}{r l}
 & 0000\,0111 \quad (+7) \\
+ & 1111\,1010 \quad (-6) \\
\hline
 & 0000\,0001 = (+1)
\end{array}
$$

Not overflow

$$
\begin{array}{r l}
 & 0111\,1111 \quad (+127) \\
+ & 0000\,0010 \quad (+2) \\
\hline
 & [1000\,0001] \neq (+129)
\end{array}
$$

negative?

8-bit representation

# Multiplication

## Steps:

- Take digit of multiplier one at a time from right to left
- Multiply the multiplicand by the single digit of multiplier
- Shift intermediate product one digit to left of earlier intermediate products

n-bit mcand & m-bit multiplier ⇒ product in (n+m) bits

## Each step of multiplication

- Place a copy of mcand in the proper place if the multiplier digit is 1.
- Place 0 in the proper place if digit is 0

## First version

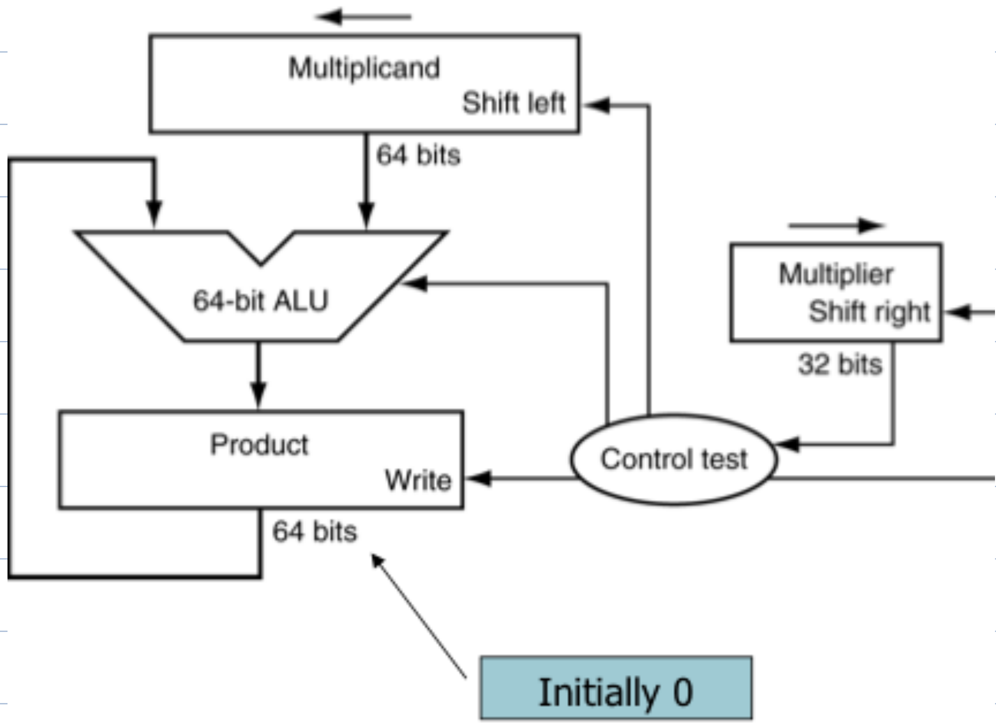
Multiplier in 32-bits
Mcand & product in 64-bits
Move multiplicand left one digit at each step to be added with intermediate products.
Multiplier is shifted right at each step.

Control decide when to shift mcand & multiplier & when to write

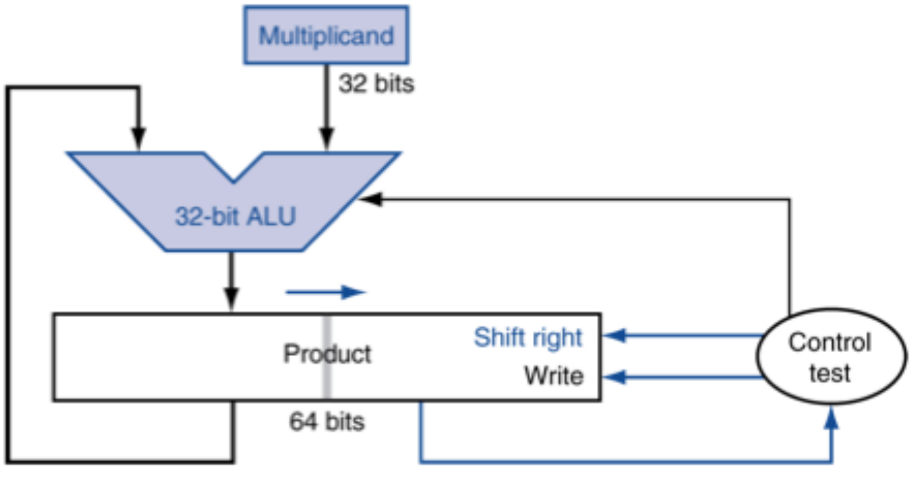
Example: mcand 0010 multiplier 0011

| I (Iteration) | Step | Multiplier | Mcand | Product |
|---|---|---|---|---|
| Ø | initial values | 0011 | 0000 0010 | 0000 0000 |
| 1 | prod = prod + Mcand | | | 0000 0010 |
| | $\overleftarrow{\text{Mcand}}$ | | 00000100 | |
| | $\overrightarrow{\text{Multiplier}}$ | 0001 | | |
| 2 | prod = prod + mcand | | | 0000 0110 |
| | $\overleftarrow{\text{mcand}}$ | | 0000 1000 | |
| | $\overrightarrow{\text{multiplier}}$ | 0000 | | |
| 3 | $\overleftarrow{\text{M Cand}}$ | | 0001 0000 | 0000 0110 |
| | $\overrightarrow{\text{Multiplier}}$ | 0000 | | |
| 4 | $\overleftarrow{\text{M cand}}$ | | | 0000 0110 |
| | $\overrightarrow{\text{multiplier}}$ | | 0010 0000 | |
| | | 0000 | | |

# Multiplication Hardware




# Optimized Multiplier



## Problems

1. 3 steps repeated 32 times (for 32-bit)
   If each 1 cycle → 3 cycles per step
   (perform operations in parallel) 3 cycle → 1 cyc

2. Half of bits in mcand always ∅
   (full 64-bit ALU wasteful) ✓
   ↳ slow for adding ∅

3. Mcand shifted left → Not affect least significant bits of product.

   Solution: Mcand is fixed relative to product & we shift product right

4. product waste space that match exactly size of multiplier

   Solution: Combine right half of product with multiplier.

In parallel: Multiplier & mcand are shifted while Mcand is added to product if multiplier bit is 1.
(Ensure that it tests right bit of multiplier & get preshifted version of mcand)

0010 x 0011 (for optimized version)

| Iteration | Step | Multiplicand | Product |
|---|---|---|---|
| 0 | Initial values | 0010 | 0000 0011 |
| 1 | 1a: 1 => Prod = Prod + Mcand | 0010 | 0010 0011 |
| | 2: Shift right Product | 0010 | 0001 0001 |
| 2 | 1a: 1 => Prod = Prod + Mcand | 0010 | 0011 0001 |
| | 2: Shift right Product | 0010 | 0001 1000 |
| 3 | 1: 0 => no operation | 0010 | 0001 1000 |
| | 2: Shift right Product | 0010 | 0000 1100 |
| 4 | 1: 0 => no operation | 0010 | 0000 1100 |
| | 2: Shift right Product | 0010 | 0000 0110 |