Computer Arithmetic And ALU Design I

Instructor: Dmitri A. Gusev

Fall 2007

CS 502: Computers and Communications

Lecture 4, September 17, 2007

How to represent a natural number?

Base of a number system: The number of digits used in the system. Example 1: Base 10 (*decimal*)

$$1760_{10} = 0*10^0 + 6*10^1 + 7*10^2 + 1*10^3$$

Numbers are written using *positional notation*.

Example 2: Base 2 (*binary*)

$$11101_2 = 1*2^0 + 0*2^1 + 1*2^2 + 1*2^3 + 1*2^4 = 29_{10}$$

Addition and Subtraction in Binary

$$
\begin{array}{lrclr}
 & 1\quad 11 & \leftarrow\text{carry} & & 1\quad \leftarrow\text{carry} \\
 & 10011 & \rightarrow & 1+2+16 = & 19 \\
+ & \underline{11001} & \rightarrow & 1+8+16 = & +\ \underline{25} \\
 & 101100 & \rightarrow & 4+8+32 = & 44
\end{array}
$$

$$
\begin{array}{lrclr}
 & 1\ \ 1\quad & \leftarrow\text{borrow} & & \\
 & 10101 & & 1+4+16 = & 21 \\
- & \underline{\ \ 1011} & & 1+2+8 = & -\ \underline{11} \\
 & 1010 & & 2+8 = & 10
\end{array}
$$

Two’s Complement Representation of Negative Numbers

- Representing

$$\mathrm{Negative}(m) = 2^k - m,$$

where *k* is the number of bits used.

Example: *k*=8,

-125	10000011	(256-125=131=1+2+128)
$+$ $\underline{\quad 3}$	$+$ $\underline{00000011}$	
-122	10000110	“134”=Negative(122))

Overflow will occur if the result of addition exceeds 127: “128” (10000000) now serves as (-128)!

Shortcuts

1. To negate a two’s complement binary number, invert every 0 to 1 and every 1 to 0, then add 1 to the result
2. To convert a binary number represented in n bits to more than n bits, replicate the most significant bit (the sign bit) to fill the new bits on the left-hand side

More Data Transfer Instructions

Instruction	Example	Meaning	Comments
load half unsigned	lhu $s1,100($s2)	$s1 = Memory[$s2+100]	Halfword memory to register
store half	sh $s1,100($s2)	Memory[$s2+100] = $s1	Halfword register to memory
load byte unsigned	lbu $s1,100($s2)	$s1 = Memory[$s2+100]	Byte from memory to register
store byte	sb $s1,100($s2)	Memory[$s2+100] = $s1	Byte from register to memory
load upper immediate	lui $s1,100	$s1 = 100*2^{16}$	Loads constant in upper 16 bits

More Conditional Branch Instructions

Instruction	**Example**	**Meaning**	**Comments**
set less than unsigned	sltu $s1,$s2,$s3	if($s2<$s3) $s1=1; else $s1=0	Compare less than; unsigned numbers
set less than immediate unsigned	sltiu $s1,$s2,100	If($s2<100) $s1=1; else $s1=0	Compare < constant; unsigned numbers

More Arithmetic Instructions

Instruction	Example	Meaning	Comments
add immediate	addi $s1,$s2,100	$s1 = $s2 + 100	+constant; overflow detected
add unsigned	addu $s1,$s2,$s3	$s1 = $s2 + $s3	overflow undetected
subtract unsigned	subu $s1,$s2,$s3	$s1 = $s2 - $s3	overflow undetected
add immediate unsigned	addiu $s1,$s2,100	$s1 = $s2 + 100	+constant; overflow undetected
move from coprocessor register	mfc0 $s1,$eps	$s1 = $eps	Used to copy Exception PC plus other special registers

Detecting Overflow

- No overflow when adding a positive and a negative number
- No overflow when signs are the same for subtraction
- Overflow occurs when the value affects the sign:
  - overflow when adding two positives yields a negative
  - or, adding two negatives gives a positive
  - or, subtract a negative from a positive and get a negative
  - or, subtract a positive from a negative and get a positive
- Consider the operations A + B, and A – B
  - Can overflow occur if B is 0 ?
  - Can overflow occur if A is 0 ?

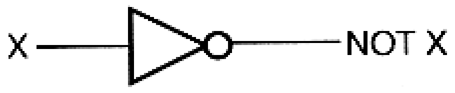
Effects of Overflow

- An exception (interrupt) occurs
  - Control jumps to predefined address for exception
  - Interrupted address is saved for possible resumption
- Don't always want to detect overflow

  — new MIPS instructions: `addu, addiu, subu`

note: `addiu` *still sign-extends!*

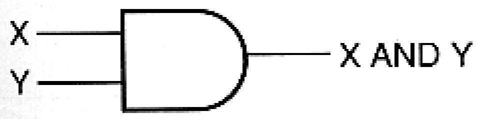
NOT Gate

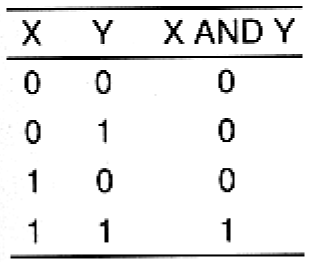


X	NOT X
0	1
1	0

A triangle with an *inversion bubble*.

AND Gate



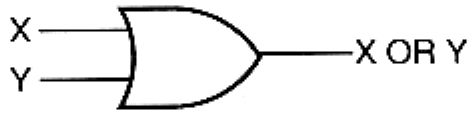


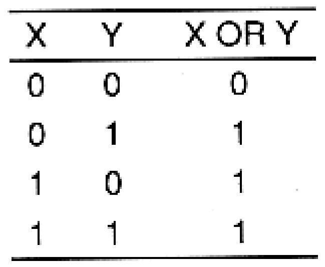
X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Boolean expression:

$A = X \cdot Y$

OR Gate





X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

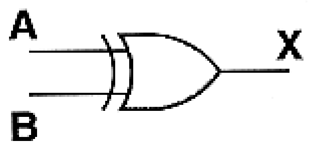
Boolean expression: $A = X + Y$

XOR Gate

Boolean Expression

$X = A \oplus B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

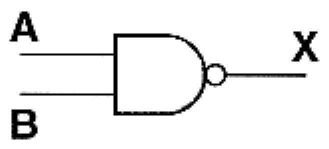
Exclusive OR.

NAND Gate

Boolean Expression

X = (A · B)'

Logic Diagram Symbol



Truth Table

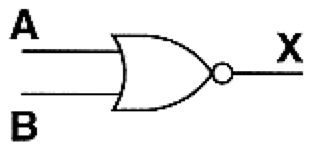
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate

Boolean Expression

X = (A + B)'

Logic Diagram Symbol



Truth Table

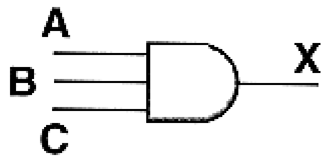
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Three-Input AND Gate

Boolean Expression

$X = A \cdot B \cdot C$

Logic Diagram Symbol



Truth Table

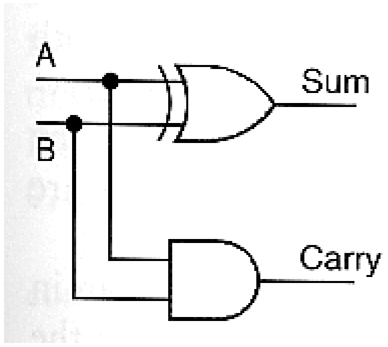
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

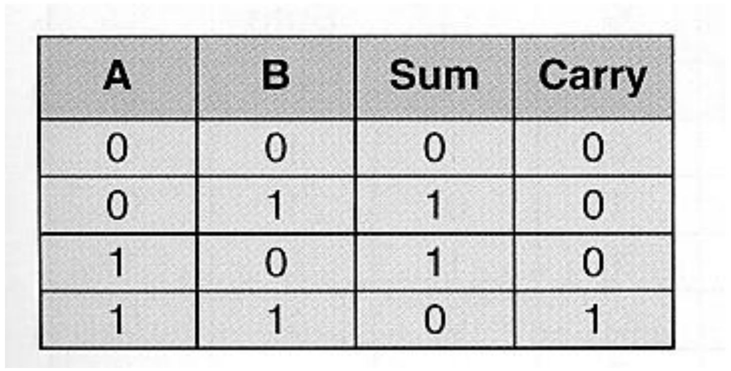
Combinatorial and Sequential Circuits: Definitions

A *combinatorial circuit* is a circuit whose output is solely determined by its input values.

A *sequential circuit* is a circuit whose output is a function of input values **and** the current state of the circuit.

Half Adder

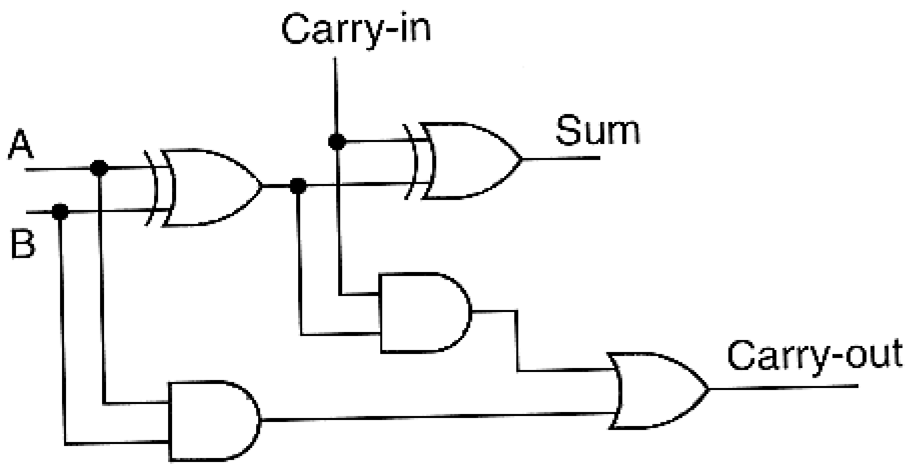


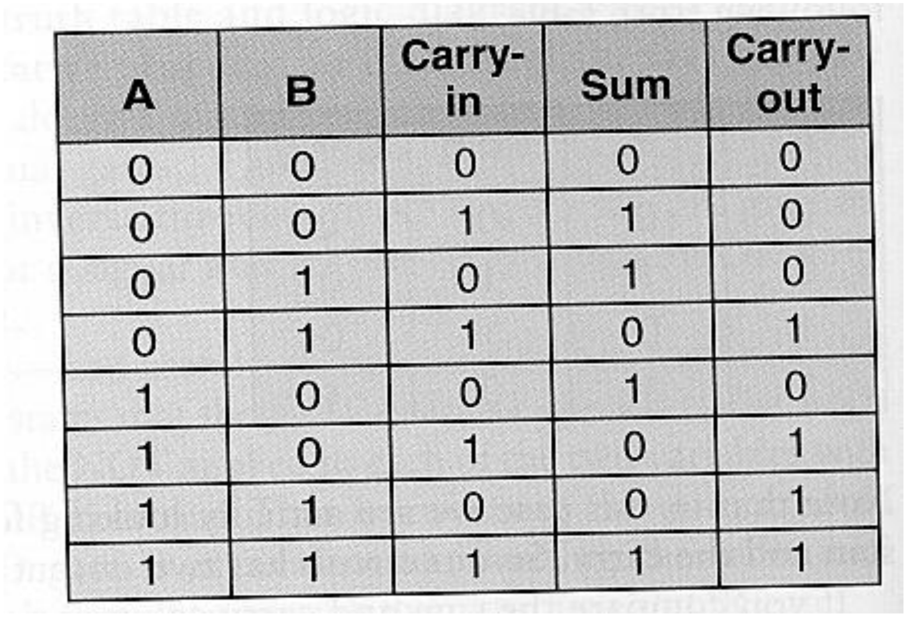


A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Problem with the Half Adder: No Carry-In

Full Adder





A	B	Carry-in	Sum	Carry-out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

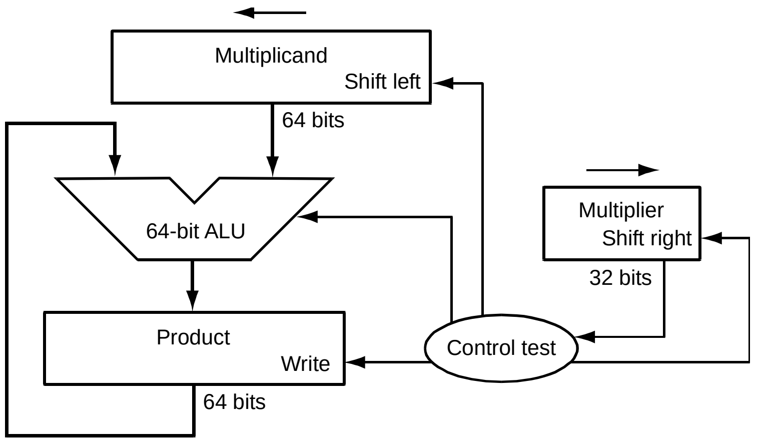
Multiplication

- More complicated than addition
  - accomplished via shifting and addition
- More time and more area
- Let's look at 3 versions based on a gradeschool algorithm

```
      0010    (multiplicand)
    x 1011    (multiplier)
  ________
```

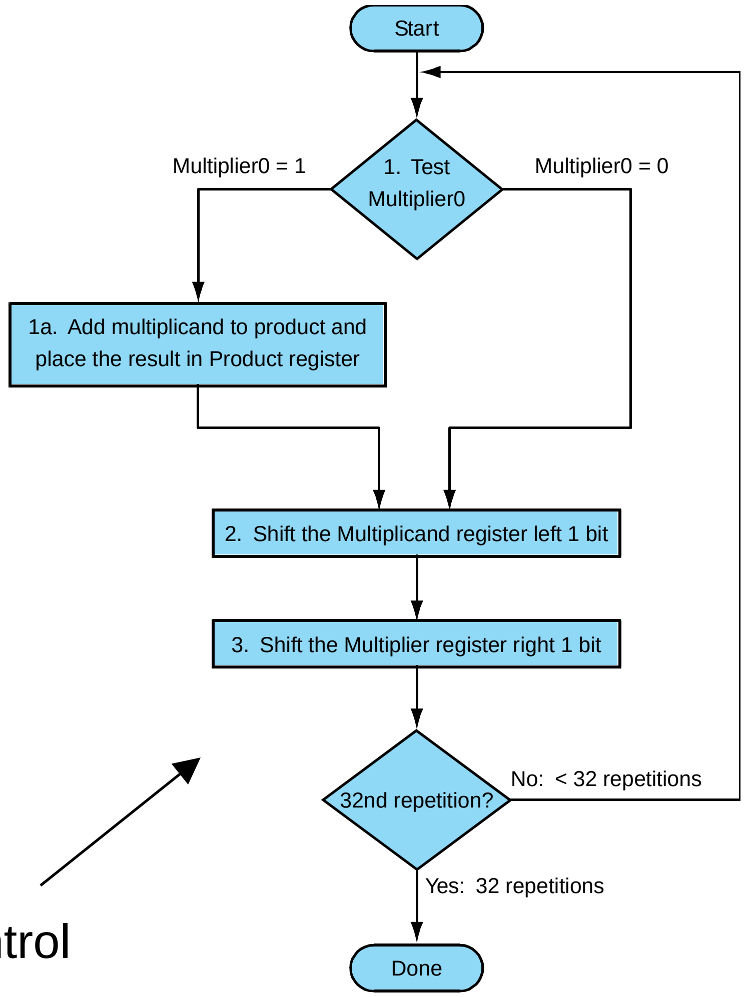
- Negative numbers: convert and multiply
  - there are better techniques, we won’t look at them

Multiplication: Implementation



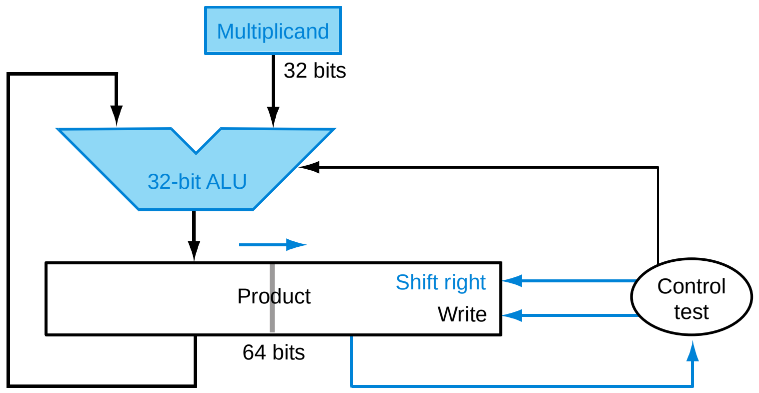
Datapath

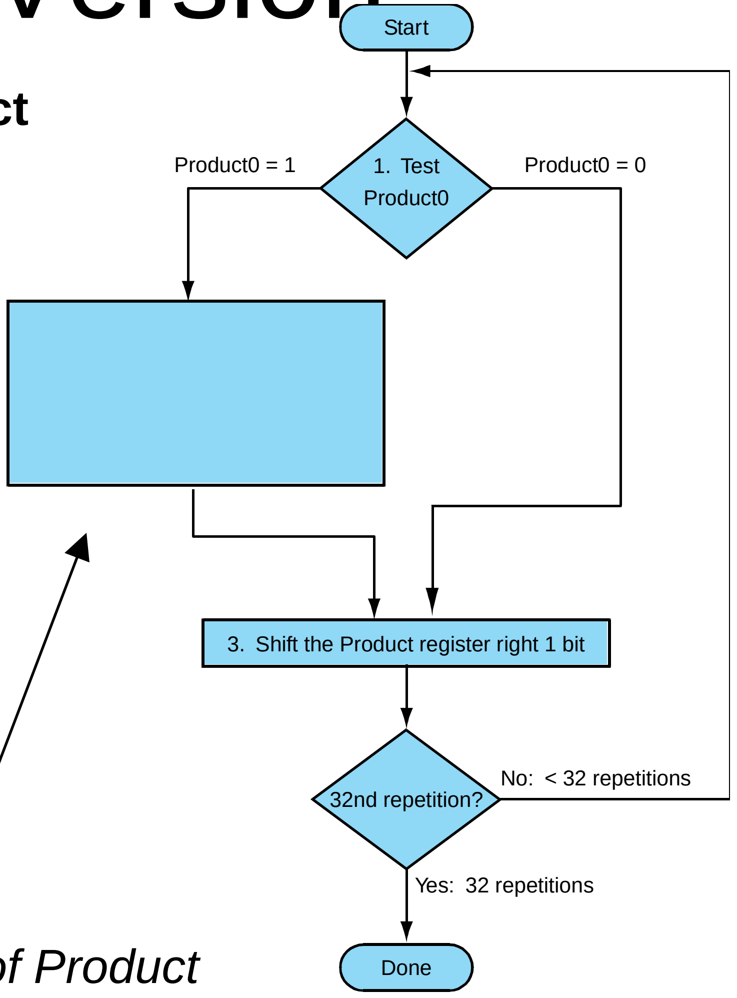
Control


Start
1. Test Multiplier0
Multiplier0 = 1
Multiplier0 = 0
1a. Add multiplicand to product and place the result in Product register
2. Shift the Multiplicand register left 1 bit
3. Shift the Multiplier register right 1 bit
32nd repetition?
No: < 32 repetitions
Yes: 32 repetitions
Done

Improved Version

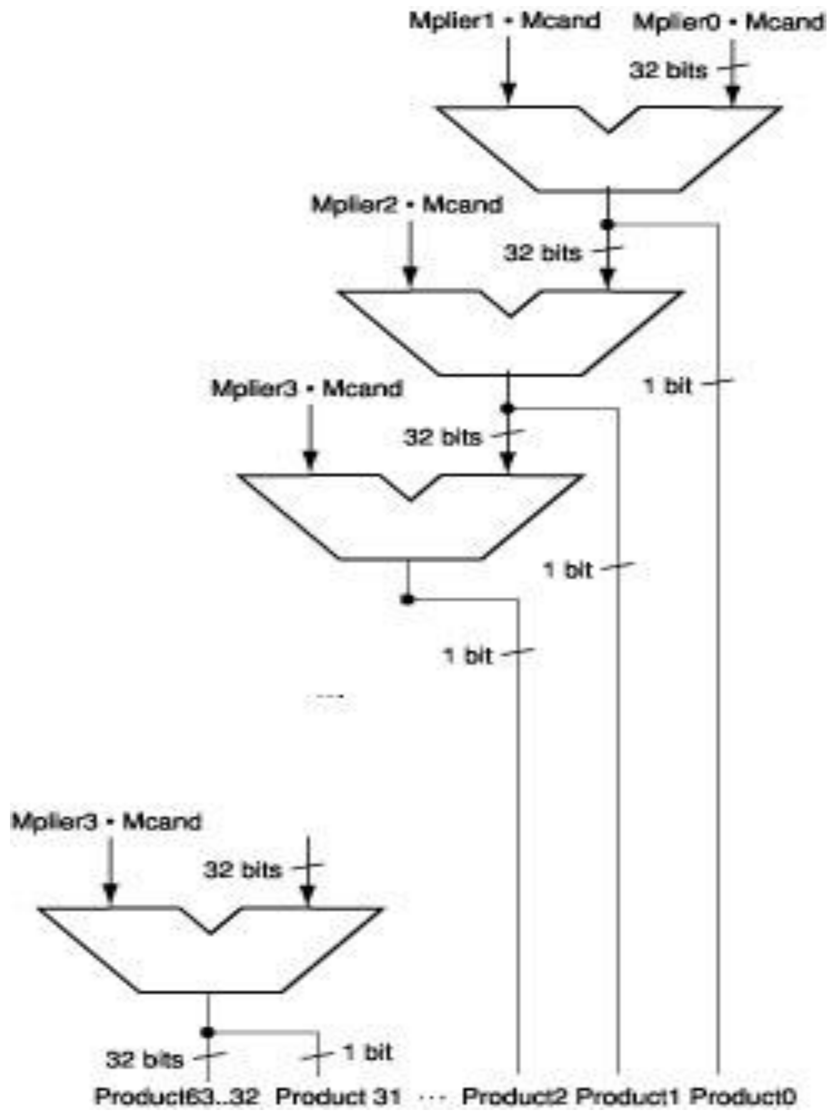
- **Multiplier starts in right half of product**





Add multiplicand to
The left-hand half of Product
(need a 65th bit for overflow)

Final Version: Fast Multiplication


Mplier1 • Mcand
Mplier0 • Mcand
32 bits
Mplier2 • Mcand
32 bits
1 bit
Mplier3 • Mcand
32 bits
1 bit
1 bit
Mplier3 • Mcand
32 bits
32 bits
1 bit
Product63..32
Product 31
Product2
Product1
Product0