

8. Basic processor design – ALU design

EECS 370 – Introduction to Computer Organization – Winter 2009

Prof. Todd Austin & Prof. Marios Papaefthymiou

EECS Department
University of Michigan in Ann Arbor, USA

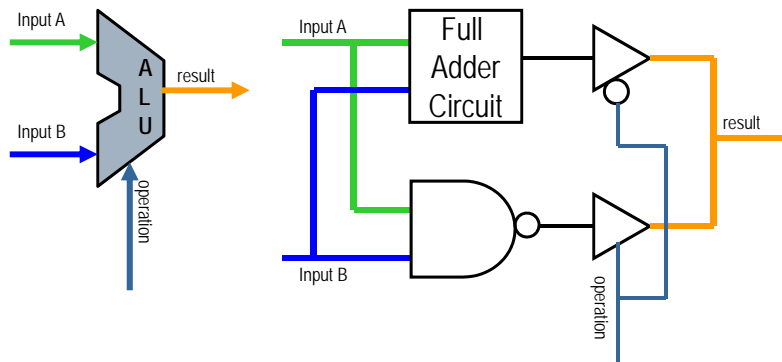
© T. Austin & M. Papaefthymiou, 2009

The material in this presentation cannot be
copied in any form without our written permission

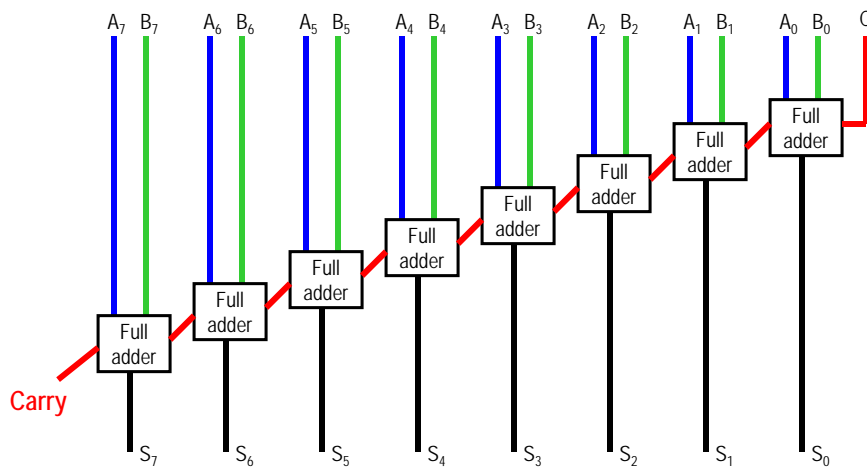
Lecture schedule

1. Combinational & Sequential Logic:
 - Basics of electronics; logic gates, muxes, decoders
 - clocks and data storage
2. **ALU design**
 - **Building an adding circuit**
3. State Machines
 - Building a simple processor

Review - ALU



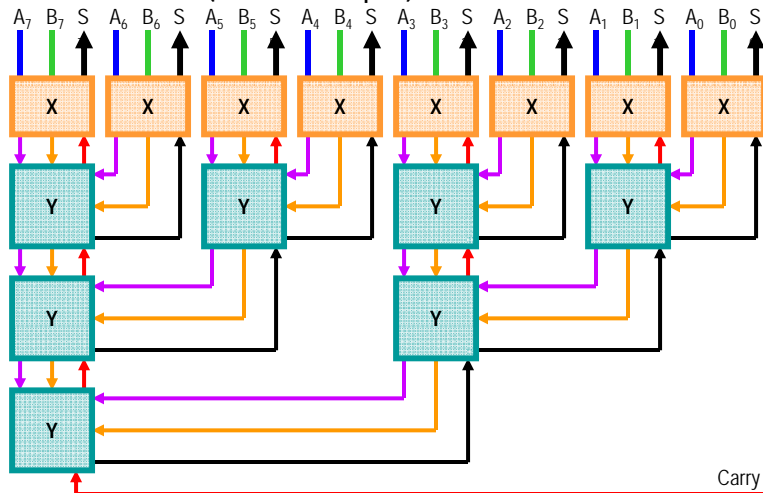
Review - 8-bit Ripple Carry Adder



Unfortunately this has a very large propagation time for 32 or 64 bit adds

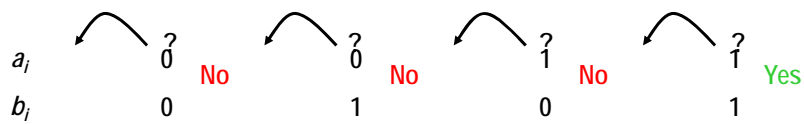
8-bit Carry Look-ahead Adder

A much faster (and more complex) adder for 32 or 64 bit adds



Single Bit Carry Generate

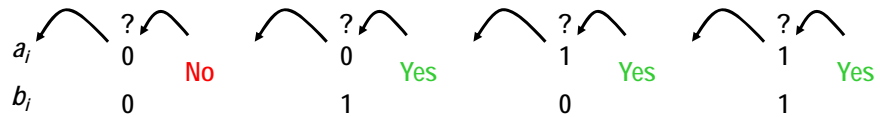
Generate: Do these two bits generate a carry?



$$g_i = a_i \& b_i$$

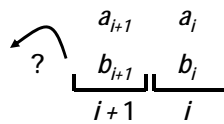
Single Bit Carry Propagate

Propagate: Do these two bits let a carry-in *propagate*?



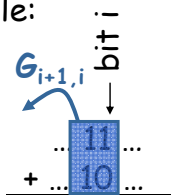
$$p_i = a_i | b_i$$

Generate for 2 Bit Range



$$G_{i+1,i} = g_{i+1} | (g_i \& p_{i+1})$$

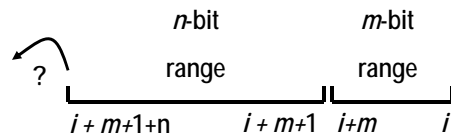
Example:



$$\begin{aligned}
 g_i &= 1 \& 0 = 0 \\
 g_{i+1} &= 1 \& 1 = 1 \\
 p_i &= 1 | 0 = 1 \\
 p_{i+1} &= 1 | 1 = 1
 \end{aligned}$$

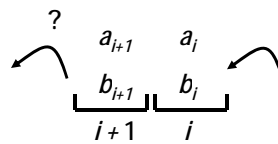
$$G_{i+1,i} = 1 | (0 \& 1) = 1$$

Generate for General Bit Ranges



$$G_{i+m+1+n,i} = G_{i+m+1+n,i+m+1} | (G_{i+m,i} \& P_{i+m+1+n,i+m+1})$$

Propagate for 2 Bit Range



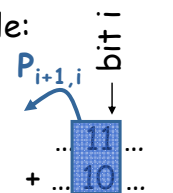
$$P_{i+1,i} = p_{i+1} \& p_i$$

$$p_i = 1 | 0 = 1$$

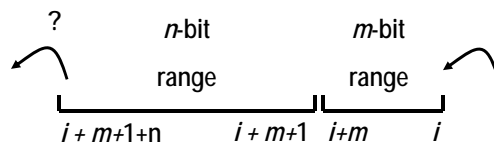
$$p_{i+1} = 1 | 1 = 1$$

$$P_{i+1,i} = 1 \& 1 = 1$$

Example:



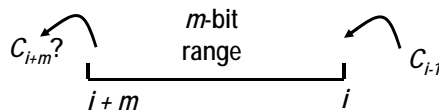
Propagate for General Bit Ranges



$$P_{i+m+1+n,i} = P_{i+m+1+n,i+m+1} \& P_{i+m,i}$$

So, What is My Carry?

Given a carry-in to a bit range and the generate and propagate signals associated with that bit range, how do we compute the carry-out?



$$C_{i+m} = G_{i+m,i} | (P_{i+m,i} \& C_{i-1})$$

Multiplier

$$\begin{array}{r} 01001101 \quad (77_{10}) \\ \times 10110011 \quad (179_{10}) \end{array}$$

$$\begin{array}{r} 01001101 \quad \times 1 \\ 010011010 \quad \times 1 \end{array}$$

$$\begin{array}{r} 010011010000 \quad \times 1 \\ 010011010000 \quad \times 1 \end{array}$$

$$\begin{array}{r} 01001101000000 \quad \times 1 \\ \hline 0110101110111 \end{array} \quad (13,783_{10})$$

Floating Point Representation

$$10.625_{10} \quad \longrightarrow \quad 1010.101_2$$

$$1.010101 \times 2^3$$

This must be a 1!
So don't store it.

+/-
1 bit

Mantissa (1010101)
23 bits

Exponent (3)
8 bits

IEEE Floating point format

- ❑ Sign bit: (0 is positive, 1 is negative)
- ❑ Mantissa: (store 23 most significant bits after the decimal point)
- ❑ Exponent: used biased base 127 encoding
 - Add 127 to the value of the exponent to encode:
 - -127 → 00000000 1 → 10000000
 - -126 → 00000001 2 → 10000001
 - ...
 - 0 → 01111111 128 → 11111111

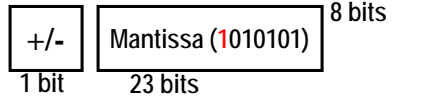
- ❑ How do you represent zero ? Special convention:
 - Exponent: -127 (all zeroes), Mantissa 0 (all zeroes), Sign + or -

Floating Point Representation

10.625_{10} \rightarrow 1010.101_2

1.010101×2^3

This must be a 1!
So don't store it.



$10.625_{10} = 0 \ 10000010 \ 010101000000000000000000$

Class Problem

- What is the value (in decimal) of the following IEEE 754 floating point encoded number?

1 1000101 010110010000000000000000

Floating Point Multiply

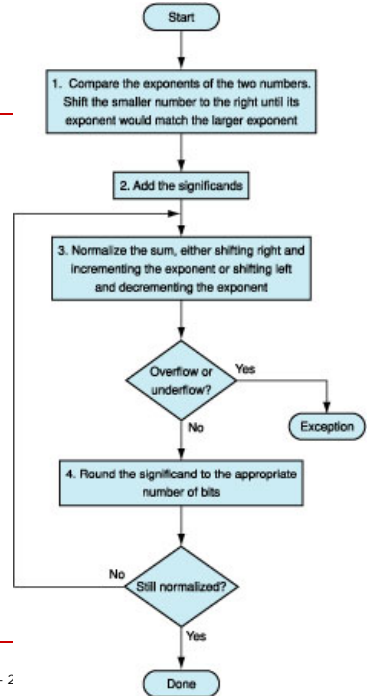
$$\begin{array}{l}
 10.625_{10} = 1010.101_2 \Rightarrow \begin{array}{|l} 0 \ 1000010 \ 0101010000000000000000 \\ + \\ 0 \ 1000010 \ 0100000000000000000000 \\ \hline 0 \ 1000101 \ 1010100100000000000000 \end{array} \\
 10_{10} = 1010_2 \Rightarrow \begin{array}{|l} 0 \ 1000010 \ 0101010000000000000000 \\ \times \\ 0 \ 1000010 \ 0100000000000000000000 \\ \hline 0 \ 1000101 \ 1010100100000000000000 \end{array} \\
 \\
 \begin{array}{r}
 1010101 \\
 \times \quad 101 \\
 \hline
 1010101 \\
 101010100 \\
 \hline
 110101001
 \end{array}
 \end{array}$$

1101010.01₂
= 106.25₁₀

Floating point Addition

1. Shift smaller exponent right to match larger.
2. Add significands
3. Normalize and update exponent
4. Check for "out of range"

See page 200 (Figure 3.16)



Class Problem

Show how to add the following 2 numbers using IEEE floating point addition: $100.125 + 13.75$

In the next episode of EECS370...

- ❑ Learn how to design a vending machine!

On a side note:

- ❑ Project 1 due this Friday (Feb 6)
- ❑ Project 2 to be released this week



Don't Panic