CS 61C:
Great Ideas in Computer Architecture
*Flip-Flops, FSMs, Logisim, Muxes*

Instructor:
David A. Patterson
http://inst.eecs.Berkeley.edu/~cs61c/sp12

**Levels of Representation/ Interpretation**

<table>
<thead>
<tr>
<th>High Level Language Program (e.g., C)</th>
<th>Assembly Language Program (e.g., MIPS)</th>
<th>Machine Language Program (MIPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiler</td>
<td>Assembly</td>
<td>Machine Interpretation</td>
</tr>
<tr>
<td>Anything can be represented as a number, i.e., data or instructions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Review**

- Real world voltages are analog, but are quantized to represent logic 0 and logic 1
- Transistors are just switches, combined to form gates: AND, OR, NOT, NAND, NOR
- Truth table can be mapped to gates for combinational logic design
- Boolean algebra allows minimization of gates

**Agenda**

- State Elements
- Administrivia
- Finite State Machines
- Introduction to Logisim
- Multiplexer
- ALU Design
- Summary

**Type of Circuits**

- *Synchronous Digital Systems* consist of two basic types of circuits:
  - Combinational Logic (CL) circuits
    - Output is a function of the inputs only, not the history of its execution
    - E.g., circuits to add A, B (ALUs)
    - Last lecture was CL
  - Sequential Logic (SL)
    - Circuits that “remember” or store information
    - Aka “State Elements”
    - E.g., memories and registers (Registers)
    - Today’s lecture is SL
Design Hierarchy

Remember This?
Conceptual MIPS Datapath

Uses for State Elements
• Place to store values for some amount of time:
  – Register files (like $1-$31 on the MIPS)
  – Memory (caches, and main memory)
• Help control flow of information between combinational logic blocks
  – State elements are used to hold up the movement of information at the inputs to combinational logic blocks and allow for orderly passage

Accumulator Example
Why do we need to control the flow of information?

First Try: Does this work?

No!
Reason #1: How to control the next iteration of the ‘for’ loop?
Reason #2: How do we say: ‘S=0’?

Second Try: How About This?
Register is used to hold up the transfer of data to adder

Rough timing...

Feedback
Model for Synchronous Systems

- Collection of Combinational Logic blocks separated by registers
- Feedback is optional
- Clock signal(s) connects only to clock input of registers
- Clock (CLK): steady square wave that synchronizes the system
- Register: several bits of state that samples on rising edge of CLK (positive edge-triggered) or falling edge (negative edge-triggered)

Register Internals

- n instances of a “Flip-Flop”
- Flip-flop name because the output flips and flops between 0 and 1
- D is “data input”, Q is “data output”
- Also called “D-type Flip-Flop”

Camera Analogy Timing Terms

- Want to take a portrait – timing right before and after taking picture
- **Set up time** – don’t move since about to take picture (open camera shutter)
- **Hold time** – need to hold still after shutter opens until camera shutter closes
- **Time click to data** – time from open shutter until can see image on output (viewfinder)

Hardware Timing Terms

- **Setup Time**: when the input must be stable before the edge of the CLK
- **Hold Time**: when the input must be stable after the edge of the CLK
- “CLK-to-Q” Delay: how long it takes the output to change, measured from the edge of the CLK

FSM Maximum Clock Frequency

- What is the maximum frequency of this circuit?

Max Delay = **Setup Time + CLK-to-Q Delay + CL Delay**

Pipelining to Improve Performance: BEFORE (1/2)

Note: delay of 1 clock cycle from input to output.
Clock period limited by propagation delay of adder/shifter
Pipelining to Improve Performance (2/2)

Timing:

- Insertion of register allows higher clock frequency
- More outputs per second

Timing diagram:

- Ready before clock edge: setup time
- Delay for Adder Combinational Logic
- Delay for Setup + Clk to Q
- Delay for Shifter Combinational Logic
- Delay for Setup + Clk to Q

Administrivia

- Project 3, Part 2: Prefer March 25 or April 1?
- A total of 131 vote(s) in 19 hours
  - 40 (31%) No! Leave it 3/25 so I can...
  - 91 (69%) Yes! Move it to 4/1 so I can...

- Due Sunday April 1
  - OK to turn it in Sunday March 25 or even Friday March 23

Administrivia

- Project 3, Part 2 due Sunday 4/1
  - Thread Level Parallelism and OpenMP
- Last homework due Sunday 4/8
- Project 4, Part 1 due Sunday 4/8; Part 2 4/15
  - Design a 16-bit pipelined computer in Logisim
  - Labs 10 and 11 prepare for Project 4
- Lab 12 – Malloc/Free in C
- Extra Credit due 4/22 – Fastest Matrix Multiply
- Final Exam Wednesday 5/9 11:30-2:30PM

Getting to Know Your Prof

- 1 son wanted to ride bike a “Century”
  - 102.3 miles in 7:04
  - Avg 15 MPH, Max 38 MPH
- 1 son wanted to Sprint Triathlon
  - Swim 0.5 miles (0.75 km)
  - Ride bike 12.5 miles (20 km)
  - Run 3.1 miles (5 km)

Another Great (Theory) Idea:
Finite State Machines (FSM)

- Function can be represented with a “state transition diagram”
- With combinational logic and registers, any FSM can be implemented in hardware

Agenda

- State Elements
- Administrivia
- Finite State Machines
- Introduction to Logisim
- Multiplexer
- ALU Design
- Summary
Example: 3 Ones FSM

FSM to detect the occurrence of 3 consecutive 1’s in the input

Draw the FSM ...

Assume state transitions are controlled by
the clock: On each clock cycle the machine checks the inputs and
moves to a new state and produces a new output ...

Hardware Implementation of FSM

Register needed to hold a representation of the machine’s state.
Unique bit pattern for each state.

Hardware for FSM: Combinational Logic

Can look at its functional specification, truth table form

Truth table ...

Logisim

• Free schematic capture/logic simulation program in Java
  – “A graphical tool for designing and simulating logic circuits”
  – Search and download version 2.7.1, online tutorial
  – ozark.hendrix.edu/~burch/logisim/
• Drawing interface based on toolbar
  – Color-coded wires aid in simulating and debugging a circuit
  – Wiring tool draws horizontal and vertical wires, automatically
    connecting to components and to other wires.
• Circuit layouts used as “subcircuits” of other circuits,
  allowing hierarchical circuit design
• Included circuit components: inputs and outputs, gates,
  multiplexers, arithmetic circuits, flip-flops, RAM memory

Logisim Wires

• Blue wires: value at that point is "unknown"
• Gray wires: not connected to anything
• OK when in process of building a circuit
• When finished => wires not be blue or gray
• If connected, all wires should be green
  – Bright green a 1
  – Dark green a 0

Agenda

• State Elements
• Administrivia
• Finite State Machines
• Introduction to Logisim
• Multiplexer
• ALU Design
• Summary
Common Mistakes in Logisim

- Connecting wires together
- Using input for output
- Connecting to edge without connecting to actual input
  - Unexpected direction of input

Data Multiplexer (e.g., 2-to-1 x n-bit-wide)

N Instances of 1-bit-Wide Mux

\[
\begin{array}{cccc}
\text{s} & \text{a} & \text{b} & \text{c} \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

How many rows in TT?

\[
c = sab + s\bar{a}b + s\bar{a}b + sab \\
= s(ab + \bar{a}b) + s(\bar{a}b + ab) \\
= s(a(\bar{b} + b)) + s((\bar{a} + a)b) \\
= s(a(1) + s((1)b) \\
= s0 + s1
\]

How Do We Build a 1-bit-Wide Mux (in Logisim)?

4-to-1 Multiplexer

\[
e = s_1s_0a + s_1s_0b + s_1s_0c + s_1s_0d
\]

Alternative Hierarchical Approach (in Logisim)
Subcircuits

- Subcircuit: Logisim equivalent of procedure or method
  - Every project is a hierarchy of subcircuits

N-bit-wide Data Multiplexer (in Logisim + tunnel)

Hardware for FSM: Combinational Logic

Can look at its functional specification, truth table form

Truth table ...

Hardware for FSM: Combinational Logic

Alternative Truth Table format: list only cases where value is a 1. Then restate as logic equations using PS1, PS0, Input

Arithmetic and Logic Unit

- Most processors contain a special logic block called “Arithmetic and Logic Unit” (ALU)
- We’ll show you an easy one that does ADD, SUB, bitwise AND, bitwise OR

Simple ALU
Adder/Subtractor: One-bit adder Least Significant Bit

\[
\begin{array}{ccc}
 a_3 & a_2 & a_1 \\
 a_0 \\
 b_3 & b_2 & b_1 \\
 b_0 \\
 s_3 & s_2 & s_1 & s_0 \\
 0 & 0 & 0 & 0 \\
 0 & 1 & 1 & 0 \\
 1 & 0 & 1 & 0 \\
 1 & 1 & 1 & 0 \\
\end{array}
\]

\[s_0 = a_0 \text{ XOR } b_0\]
\[c_1 = a_0 \text{ AND } b_0\]

Adder/Subtractor: One-bit adder (1/2)

\[
\begin{array}{cccc|cc}
 & a_3 & a_1 & c_i & s_i & c_{i+1} \\
\hline
\ldots & 0 & 0 & 0 & 0 & 0 \\
\hline
 a_3 & a_2 & a_1 & a_0 & 0 & 1 & 0 & 1 & 0 \\
 b_3 & b_2 & b_1 & b_0 & 0 & 1 & 1 & 0 & 1 \\
 s_3 & s_2 & s_1 & s_0 & 1 & 0 & 1 & 0 & 1 \\
 & & & & 1 & 1 & 0 & 0 & 1 \\
 & & & & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

Adder/Subtractor: One-bit Adder (2/2)

\[s_i = \text{ XOR}(a_i, b_i, c_i)\]
\[c_{i+1} = \text{ MAJ}(a_i, b_i, c_i) = a_i b_i + a_i c_i + b_i c_i\]

N x 1-bit Adders ⇒ 1 N-bit Adder

Connect Carry Out i-1 to Carry in i:

Twos Complement Adder/Subtractor

Critical Path

- When setting clock period in synchronous systems, must allow for worst case
- Path through combinational logic that is worst case called “critical path”
  - Can be estimated by number of “gate delays”:
    Number of gates must go through in worst case
- Idea: Doesn’t matter if speedup other paths if don’t improve the critical path
- What might critical path of ALU?
Summary

- Hardware systems made from *Stateless* Combinational Logic and *Stateful* “Memory” Logic (Registers)
- Clocks tell us when D-flip-flops change
  - Setup and Hold times important
- We pipeline long-delay CL for faster clock cycle
  - Split up the *critical path*
- Finite State Machines extremely useful
- Use muxes to select among input
  - S input bits selects $2^S$ inputs
  - Each input can be n-bits wide, indep of S
- Can implement muxes hierarchically
- Can implement FSM with register + logic