CS 61C: Great Ideas in Computer Architecture (Machine Structures)

FSMs and Logisim

Instructors:
Randy H. Katz
David A. Patterson

http://inst.eecs.Berkeley.edu/~cs61c/sp11

Spring 2011 - Lecture #17

You Are Here!

• Parallel Requests
  Assigned to computer
e.g., Search “Katz”
• Parallel Threads
  Assigned to core
e.g., Lookups, Ads
• Parallel Instructions
  >1 instruction @ one time
e.g., 5 pipelined instructions
• Parallel Data
  >1 data item @ one time
e.g., Add of 4 pairs of words

Hardware descriptions
All gates functioning in parallel at same time

Smart Phone
Warehouse Scale
Computer

Hardware descriptions
All gates functioning in parallel at same time

Logic Gates

Review

• Hardware systems are constructed from Stateless Combinational Logic and Stateful “Memory” Logic (Registers)
• Real world voltages are analog, but are quantized to represent logic 0 and logic 1
• Truth table can be mapped to gates for combinational logic design
• Boolean algebra allows minimization of gates
• State registers implemented from Flip-flops

Levels of Representation/Interpretation

High-Level Language Program (e.g., C)

Compiler

Assembly Language Program (e.g., MIPS)

Assembler

Machine Language Program (MIPS)

Machine Interpretation

Hardware Architecture Description (e.g., block diagrams)

Logic Circuit Description (Circuit Schematic Diagrams)

Review

• Hardware systems are constructed from Stateless Combinational Logic and Stateful “Memory” Logic (Registers)
• Real world voltages are analog, but are quantized to represent logic 0 and logic 1
• Truth table can be mapped to gates for combinational logic design
• Boolean algebra allows minimization of gates
• State registers implemented from Flip-flops

Agenda

• State Elements
• Finite State Machines
• Administrivia
• Introduction to Logisim
• Technology Break
• Multiplexer
• ALU Design

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)
Camera Analogy

- 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)

Timing Terms

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

Maximum Clock Frequency

- What is the maximum frequency of this circuit?

\[
\text{Max Delay} = \text{Setup Time} + \text{CLK-to-Q Delay} + \text{CL Delay}
\]

Hint:
Frequency = 1/Period

Pipelining to Improve Performance (1/2)

Extra Register are often added to help speed up the clock rate

Pipelining to Improve Performance (2/2)

Insertion of register allows higher clock frequency
More outputs per second

Another Great Idea:
Finite State Machines (FSM)

- You may have seen FSMS in other classes
- Same basic idea
- Function can be represented with a “state transition diagram”
- With combinational logic and registers, any FSM can be implemented in hardware
Example: 3 Ones FSM

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

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |
```

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

```
<table>
<thead>
<tr>
<th>PS</th>
<th>Input</th>
<th>NS</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>00</td>
<td>01</td>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>10</td>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>10</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>
```

Approximate 61C Grade So Far

- ~55% points for full course
- 25% A, 50% B, 15% C, 10% D-F
  - GPA = 2.85
  - Fall 61C 2.81
- Extra credit moves up people near borderline (e.g., B to B)

Administrivia

- Project 3, Part 2 due Sunday 4/3
  - Thread Level Parallelism and OpenMP
- Last homework due Sunday 4/10
- Project 4, Part 1 due Sunday 4/10; Part 2 4/17
  - 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 4/24 – Fastest Matrix Multiply
- Final Exam Monday 5/9 11:30-2:30PM

Midway Survey Results

- Early start by email for Fall 2011 61C?
  - 70% yes, 25% maybe
- Read textbook?
  - 33% before lectures
  - 50% before assignment
- Attend lecture?
  - 80% rarely/never miss
- Pace of lecture?
  - 45% little fast, 30% just right, 10% little slow
- Enjoyed, Learned a lot
  - MapReduce lab, project
  - MIPS emulator project
- Satisfactory, learned some (all others)
- Get to know Prof?
  - 2/3 like, do more
  - 1/3 OK
- Peer instruction?
  - 25% wish more, 55% OK
61c in the News: Paul Baran, Internet Pioneer, Dies at 84

- “The internet is really the work of a thousand people”
- In the early 1960s he outlined the fundamentals for packaging data into discrete bundles, which he called “message blocks.” The bundles are then sent on various paths around a network and reassembled at their destination. Such a plan is known as “packet switching.”

Getting to Know Your Prof

- Idea was to build a distributed communications network, less vulnerable to attack or disruption than conventional networks. He suggested that networks be designed with redundant routes so that if a particular path failed or was destroyed, messages could still be delivered through another.
- When he approached AT&T with the idea to build his proposed network, the company insisted it would not work and refused. AT&T didn’t participate in ARPAnet either.

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

Logisim

- Free schematic capture/logic simulation program in Java
  - Search and download version 2.7.0
- 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 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, and RAM memory

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

<table>
<thead>
<tr>
<th>Data Multiplexer</th>
<th>N Instances of 1-bit-Wide Mux</th>
</tr>
</thead>
<tbody>
<tr>
<td>a b</td>
<td>c</td>
</tr>
<tr>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
</tr>
<tr>
<td>0 10</td>
<td>1</td>
</tr>
</tbody>
</table>

| a b c |
| 0 0 0 |
| 0 0 1 |
| 0 10 1 |

How many rows in TT?

- $c = s_{ab} + s_{ab} + s_{ab} + s_{ab}$
- $c = s_{a(b+b)} + s((a+a)b)$
- $c = s(a(1) + s((1)b)$
- $c = sa + sb$

| a b c |
| 1 00 0 |
| 1 01 1 |
| 1 10 0 |
| 1 11 1 |
How Do We Build a 1-bit-Wide Mux (in Logisim)?

Alternative Hierarchical Approach (in Logisim)

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

Hardware for FSM: Combinational Logic

Hardware for FSM: Combinational Logic

Can look at its functional specification, truth table form

Truth table ...

<table>
<thead>
<tr>
<th>PS</th>
<th>Input</th>
<th>NS</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>00</td>
<td>1</td>
</tr>
</tbody>
</table>

NS bit 0 is 1

NS bit 1 is 1

Output is 1

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

• NS0 = PS1•PS0•Input
  • NS0 = ~PS1•PS0•Input

• NS1 = PS1•PS0•Input
  • NS1 = ~PS1•PS0•Input

• Output = PS1•PS0•Input
  • Output = ~PS1•PS0•Input
Common Mistakes in Logisim

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

Summary

- Clocks tell us when D-flip-flops change
  - Setup and Hold times important
- We pipeline long-delay CL for faster clock
- 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