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

Functional Units, FSMs
Instructor: Michael Greenbaum

Review

- **Synchronous Digital Systems**
  - Synchronous - Pulse of a Clock controls flow of information
  - Digital - All signals are seen as either 0 (below a certain voltage) or 1.
- SDS constructed from two types of elements:
  - Combinational Logic - Stateless, output only a function of immediate inputs
  - State Elements (Registers)
- Truth table can be mapped to gates for combinational logic design
- Boolean algebra allows minimization of gates
- State registers implemented from Flip-flops

Accumulator Revisited

...Again

- reset signal shown
- In practice X might not arrive to the adder at the same time as \( S_{i-1} \)
- \( S_i \) temporarily is wrong, but register always captures correct value
- In good circuits, instability never happens around rising edge of clk

Agenda

- State Elements
- Finite State Machines
- Administrivia / Break
- Multiplexers
- ALU Design
- Adder/Subtractor

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)
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?

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

The Critical Path

- The Critical Path is the longest delay between any two registers in a circuit.
- The clock period must be longer than this critical path, or the signal will not propagate to that next register.

Pipelining to Improve Performance

- Insertion of register allows higher clock frequency
- More outputs per second
- However, 2 (shorter) clock cycles to produce first output. Higher latency!

Another Great Idea:
Finite State Machines (FSM)

- You may have seen FSMs in other classes (have you?)
- Function can be represented with a “state transition diagram”
- With combinational logic and registers, any FSM can be implemented in hardware
FSM Overview

- An FSM (in this class) is defined by:
  - A set of states $S$
  - A starting state $s_0$
  - A transition function that maps from Current Input and Current State to Current Output and Next State. (The arrows in the diagram)

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.

Combinational logic circuit is used to implement a function maps from present state and input to next state and output.

The register is used to break the feedback path between NS and PS, controlled by the clock

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>01</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>01</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>

Administrivia

- Midterm rubric is posted
  - Re-grade deadline extended to next Thursday since I was late in getting this up.
  - “We will re-grade the entire test”
- HW3, Project 2 Part 2 will be posted today.

Agenda

- State Elements
- Finite State Machines
- Administrivia / Break
- Multiplexers
  - ALU Design
  - Adder/Subtractor
Data Multiplexer (e.g., 2-to-1 x n-bit-wide)

\[ c = \overline{s}a\overline{s}b + \overline{s}ab + s\overline{a}b + sab \]
\[ = \overline{s}(ab + ab) + s(\overline{a}b + ab) \]
\[ = \overline{s}(a(b + b)) + s((\overline{a} + a)b) \]
\[ = s((1)b) + s((1)b) \]
\[ = s\overline{a} + sb \]

How Do We Build a 1-bit-Wide Mux?

4-to-1 Multiplexer

Alternative Hierarchical Approach

In Logisim
Common Mistakes in Logisim

- Connecting wires together
- Losing track of which input is which
- Connecting to edge without connecting to actual input
  - Unexpected direction of 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

Adder/Subtractor: One-bit adder for Least Significant Bit

\[
\begin{array}{c c c c c c}
 & a_3 & a_2 & a_1 & a_0 \\ + & b_3 & b_2 & b_1 & b_0 \\ \hline
 s_3 & s_2 & s_1 & s_0 & c_1 \\
\end{array}
\]

<table>
<thead>
<tr>
<th>a_0</th>
<th>b_0</th>
<th>s_0</th>
<th>c_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
s_0 =
\]

\[
c_1 =
\]
Adder/Subtractor: One-bit adder (1/2)

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

N x 1-bit Adders ⇒ 1 N-bit Adder

Twos Complement Adder/Subtractor

Summary

- D-flip-flops update at rising edge of clock
  - Setup and Hold times important
- Critical path constrains clock rate
- 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
- Build n-bit adder out of chained 1-bit adders.