We whomp UCLA!

After outplaying the #1-ranked team, we come back at our homecoming game and beat UCLA (ex-powerhouse) 45-28. Arrington rushes for 205 yards. 550 yards total!

At AZ next week. We’re ranked 7th!
Review…

• ISA is very important abstraction layer
  • Contract between HW and SW
• Basic building blocks are logic \textit{gates}
• Clocks control pulse of our circuits
• Voltages are analog, quantized to 0/1
• Circuit delays are fact of life
• Two types
  • Stateless Combinational Logic (\&,\|,\neg), in which \textit{output is function of input only}
  • State circuits (e.g., registers)
Accumulator Example

Want: \( S=0; \text{ for } i \text{ from } 0 \text{ to } n-1 \)

\[ S = S + X_i \]
First try… Does this work?

Feedback!

Nope!

Reason #1… What is there to control the next iteration of the ‘for’ loop?

Reason #2… How do we say: ‘s=0’?
Second try... How about this? Yep!

Rough timing...
Register Details...What’s in it anyway?

- \( n \) instances of a “Flip-Flop”, called that because the output flips and flops betw. 0,1
- \( D \) is “data”
- \( Q \) is “output”
- Also called “d-q Flip-Flop”, “d-type Flip-Flop”
What’s the timing of a Flip-flop? (1/2)

- Edge-triggered d-type flip-flop
  - This one is “positive edge-triggered”

- “On the rising edge of the clock, the input d is sampled and transferred to the output. At all other times, the input d is ignored.”
What’s the timing of a Flip-flop? (2/2)

- Edge-triggered d-type flip-flop
  - This one is “positive edge-triggered”
  - “On the rising edge of the clock, the input d is sampled and transferred to the output. At all other times, the input d is ignored.”
Administrivia

• Midterm tonight!
  • 1 Pimintel 7pm-10pm
Accumulator Revisited (proper timing 1/2)
Accumulator Revisited (proper timing 2/2)
Pipelining to improve performance (1/2)

Timing…
Pipelining to improve performance (2/2)

Timing...
Finite State Machines Introduction
Finite State Machine Example: 3 ones...

Draw the FSM...

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>
Hardware Implementation of FSM

\[ \text{reg} \leftarrow \text{CLK} \]

\[ \text{present state} (PS) \]

\[ + \]

\[ \text{INPUT} \rightarrow \text{PS} \]

\[ \text{next state} (NS) \rightarrow \text{OUTPUT} \]

\[ = \]

\[ \text{INPUT} \]

\[ \text{next state} (NS) \rightarrow \text{OUTPUT} \]

\[ \text{reg} \leftarrow \text{CLK} \]

\[ \text{OUTPUT} \]
General Model for Synchronous Systems
Peer Instruction

A. HW feedback akin to SW recursion

B. We can implement a D-Q flipflop as simple CL (And, Or, Not gates)

C. You can build a FSM to signal when an equal number of 0s and 1s has appeared in the input.

<table>
<thead>
<tr>
<th></th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FFF</td>
</tr>
<tr>
<td>2</td>
<td>FFT</td>
</tr>
<tr>
<td>3</td>
<td>FTF</td>
</tr>
<tr>
<td>4</td>
<td>FTT</td>
</tr>
<tr>
<td>5</td>
<td>TFF</td>
</tr>
<tr>
<td>6</td>
<td>TFT</td>
</tr>
<tr>
<td>7</td>
<td>TTF</td>
</tr>
<tr>
<td>8</td>
<td>TTT</td>
</tr>
</tbody>
</table>
Peer Instruction Answer

A. HW feedback akin to SW recursion

B. We can implement a D-Q flipflop as simple CL (And, Or, Not) gates

C. You can build a FSM to signal when an equal number of 0s and 1s has appeared in the input.

A. It needs 'base case' (reg reset), way to step from i to i+1 (use register + clock).

° True!

B. D-Q has state, CL never has state!

° False!

C. How many states would it have? Say it's n. How does it know when n+1 bits have been seen?

° False!
“And In conclusion...”

• We use feedback to maintain state
• Register files used to build memories
• D-FlipFlops used to build Register files
• Clocks tell us when D-FlipFlops change
  • Setup and Hold times important
• We pipeline big-delay CL for faster clock
• Finite State Machines extremely useful
  • You’ll see them in HW classes (150,152) & 164