Controller Design

- Five step controller design process

1. Capture the FSM
   - Create an FSM that describes the desired behavior of the controller.

2. Create the architecture
   - Create the standard architecture by using a state register of appropriate width, and combinational logic with inputs being the state register bits and the FSM outputs.

3. Resolve the states
   - Resolve the states such that each state has a unique encoding.

4. Create the state table
   - Generate a truth table for the combinational logic such that the logic will generate the correct FSM outputs and next state signals. Reducing the inputs to this state table results in a state table that describes the next state behavior, so the table is in a state table.

5. Implement the combinational logic
   - Implement the combinational logic using any method.

Controller Design: Laser Timer Example

- Step 1: Capture the FSM
  - Already done

- Step 2: Create architecture
  - Next state register (for 4 states)
  - Input b, output x
  - Next state signals n1, n0

- Step 3: Encode the states
  - Any encoding with each state unique will work
Controller Design: Laser Timer Example (cont)

- Step 5: Implement combinational logic (cont)

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Want generate sequence 0001, 0011, 1100, 1000, (repeat)
  - Want simple sequential circuit that converts button press to single cycle duration, regardless of length of time that button actually pressed
    - We assumed such an ideal button press signal in earlier example, like the button in the laser timer controller

Controller Example: Button Press Synchronizer

- Step 1: FSM
  - Create FSM
  - Inputs: bi; FSM outputs: bo
  - FSM inputs: bi; FSM outputs: bo

- Step 2: Create architecture
  - Create architecture
  - FSM inputs: bi; FSM outputs: bo

- Step 3: Encode states
  - Encode states
  - States: s0, s1
  - States: s0, s1

- Step 4: State table
  - State table
  - States: s0, s1
  - States: s0, s1

Controller Example: Sequence Generator

- Want generate sequence 0001, 0011, 1100, 1000, (repeat)
  - Each value for one clock cycle
  - Common, e.g., to create pattern in 4 lights, or control magnets of a "stepper motor"

Controller Example: Secure Car Key

- From earlier example

Understanding the Controller's Behavior

- Step 1: FSM
  - FSM inputs: bi; FSM outputs: bo
  - FSM inputs: bi; FSM outputs: bo

- Step 2: Create architecture
  - Create architecture
  - FSM inputs: bi; FSM outputs: bo

- Step 3: Encode states
  - Encode states
  - States: s0, s1
  - States: s0, s1

- Step 4: State table
  - State table
  - States: s0, s1
  - States: s0, s1

- Step 5: Create combinational circuit
  - Create combinational circuit
**Example: Seq. Circuit to FSM (Reverse Engineering)**

What does this circuit do?

\[
y' = S_1' \\
z = S_1S_0' \\
\text{Out} = (S_1 \oplus S_0)x
\]

What does this circuit do?

\[
y' = S_1' \\
z = S_1S_0' \\
\text{Out} = (S_1 \oplus S_0)x
\]

Pick any state names you want

---

**Design Challenge**

- **Design Challenge**
  - Draw a state diagram for a finite state machine that has an input \( x \) and an output \( y \). Whenever \( x \) changes from 0 to 1, \( y \) should be 1 for four clock cycles and then return to 0 (even if \( x \) is still 1). Using the five-step sequential logic design process, implement the controller using a state register and logic gates.

Due Next Lecture (as announced in class)
1 point extra credit (Homework)