Lecture #7: Intro to Synchronous Sequential State Machine Design

Paul Hartke
Phartke@stanford.edu
Stanford EE121
January 29, 2002

Administrivia

- Midterm #1 is next Tuesday (February 5th) in class.
  - Will not include state machines.
- Lab 3 Design Post-Mortem
  - Comments/Issues?
- Lab 4 handout
  - Due next week as normal.
- HW3 handout
  - Due Next Thursday February 7th
  - Read it over and I’ll answer any questions on Thursday.
Two Types of Logic Circuits

• Combinational
  – A circuit whose outputs depend only on its current inputs

• Sequential
  – A circuit whose outputs depend not only on its current inputs, but also on the past sequence of inputs, possibly arbitrarily far back in time.

Readings

• In DDPP, Chapter 7 Intro, 7.3 (not 7.3.5), 7.4 (not 7.4.5), 7.5, 7.7
• We’ll do 7.8 next time
States and State Variables

• “The state of a sequential circuit is a collection of state variables whose values at any one time contain all the information about the past necessary to account for the circuit’s future behavior.”

• The states are normally encoded as binary numbers so for \( n \) state variables, there are \( 2^n \) possible states.
  – Since there is a finite number of states, these circuits are also called finite-state machines (FSM).

Basic Sequential Element

• Need an element that remembers: D Flip Flop (DFF)

• Lots of way of building this element (or an analogous one)—we’ll talk about ways later.
Simultaneous Input Changes

• Q: What if D and CLK change at the same time?
  – A: Bad things so do not change the input near the clock transition
• Setup Time: the amount of time the synchronous input (D) must be stable before the active edge of the clock.
• Hold Time: the amount of time the synchronous input (D) must be stable after the active edge of the clock.

Setup and Hold Time Diagram

• If changes on D input violate either setup or hold time, then correct FF operation is not guaranteed.
• If they are violated, metastability results.
Mealy State Machine

• A Mealy state machine’s output depends on both the state variables and the current input.

Moore State Machine

• A Moore state machine’s outputs only depend on the state variables.
Mealy vs. Moore

- Moore machine guarantees the outputs are steady for a full clock cycle.
- However, a change at the input takes at least one clock cycle to affect the output.
- Moore machine might require more states since not dependent on the input.
- Most of the time, I use a Moore machine.

State Machine Design Process

1. Determination of inputs and outputs.
2. Determination of machine states.
3. Create State/Bubble Diagram—should this be a Mealy or Moore machine?
4. State Assignment—assign each state a particular value.
5. Create Transition/Output Table
6. Derive Next State Logic for each state element—using K-maps as necessary.
7. Derive Output logic.
8. Implement in Xilinx.
1’s Counting Machine

• Design a clocked synchronous state machine with two inputs, X and Y, and one output, Z. The output should be 1 if the number of 1 inputs on X and Y since reset is a multiple of 4, and 0 otherwise.
  – Section 7.4.6 in DDPP

Machine States

• S0 \rightarrow \text{Got zero 1s (modulo 4)}
• S1 \rightarrow \text{Got one 1 (modulo 4)}
• S2 \rightarrow \text{Got two 1s (modulo 4)}
• S3 \rightarrow \text{Got three 1s (modulo 4)}
Bubble Diagram

State and Output Table

- $S^*$ is the next state given the current state and the inputs.

<table>
<thead>
<tr>
<th>XY</th>
<th>Meaning</th>
<th>$S$</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Got zero 1s (modulo 4)</td>
<td>S0</td>
<td>S0</td>
<td>S1</td>
<td>S2</td>
<td>S1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Got one 1 (modulo 4)</td>
<td>S1</td>
<td>S1</td>
<td>S2</td>
<td>S3</td>
<td>S2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Got two 1s (modulo 4)</td>
<td>S2</td>
<td>S2</td>
<td>S3</td>
<td>S0</td>
<td>S3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Got three 1s (modulo 4)</td>
<td>S3</td>
<td>S3</td>
<td>S0</td>
<td>S1</td>
<td>S0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$S^*$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
State Assignment

- Use Binary Encoding but in K-map order.
  - Requires two state elements/DFFs.
  - Semi-Arbitrary decision.
- \( S_0 \rightarrow 00 \)
- \( S_1 \rightarrow 01 \)
- \( S_2 \rightarrow 11 \)
- \( S_3 \rightarrow 10 \)

Transition/Excitation Table

- Since we will only use D-flip flops, the transition and excitation tables are the same.
  - For a DFF, \( Q^* = D \)

<table>
<thead>
<tr>
<th>( XY )</th>
<th>( Q1 )</th>
<th>( Q2 )</th>
<th>( 00 )</th>
<th>( 01 )</th>
<th>( 11 )</th>
<th>( 10 )</th>
<th>( Z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 00 )</td>
<td>( 00 )</td>
<td>( 01 )</td>
<td>( 11 )</td>
<td>( 10 )</td>
<td>( 1 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 01 )</td>
<td>( 01 )</td>
<td>( 11 )</td>
<td>( 10 )</td>
<td>( 11 )</td>
<td>( 0 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 11 )</td>
<td>( 11 )</td>
<td>( 10 )</td>
<td>( 00 )</td>
<td>( 10 )</td>
<td>( 0 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 10 )</td>
<td>( 10 )</td>
<td>( 00 )</td>
<td>( 01 )</td>
<td>( 00 )</td>
<td>( 0 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( Q1^* \) \( Q2^* \) or \( D1 \) \( D2 \)
Derive Next State Logic

- Use a K-Map for each state register input

Logic Equations

- AKA “Next-State” Logic
- \[ D_1 = Q_2 \cdot X' \cdot Y + Q_1' \cdot X \cdot Y + Q_1 \cdot X' \cdot Y' + Q_2 \cdot X \cdot Y' \]
- \[ D_2 = Q_1' \cdot X' \cdot Y + Q_1' \cdot X \cdot Y' + Q_2 \cdot X' \cdot Y' + Q_2' \cdot X \cdot Y \]
- \[ Z = Q_1' \cdot Q_2' \]
One-Hot Encoding

- Alternative encoding of state variables.
- Use one state element for each state variable
- $S_0 \rightarrow 0001$
- $S_1 \rightarrow 0010$
- $S_2 \rightarrow 0100$
- $S_3 \rightarrow 1000$

One-Hot Transition Table

- Same as before…

<table>
<thead>
<tr>
<th>Q</th>
<th>$XY=00$</th>
<th>$XY=01$</th>
<th>$XY=11$</th>
<th>$XY=10$</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>0001</td>
<td>0010</td>
<td>0100</td>
<td>0010</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>0010</td>
<td>0100</td>
<td>1000</td>
<td>0100</td>
<td>0</td>
</tr>
<tr>
<td>0100</td>
<td>0100</td>
<td>1000</td>
<td>0001</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
<td>0001</td>
<td>0010</td>
<td>0001</td>
<td>0</td>
</tr>
</tbody>
</table>
Logic Equations

- $D_1 = Q_1X''Y' + Q_2X''Y + Q_2X'Y' + Q_3X'Y$
- $D_2 = Q_2X''Y' + Q_3X''Y + Q_3X'Y' + Q_4X'Y$
- $D_3 = Q_3X''Y' + Q_4X''Y + Q_4X'Y' + Q_1X'Y$
- $D_4 = Q_4X''Y' + Q_1X''Y + Q_1X'Y' + Q_2X'Y$
- $Z = Q_4$

One-Hot Value?

- One-Hot decreases the decode depth required for next state logic at the expense of more DFFs and logic.
  - Decode depth of next state logic largely determines speed of state machine.
  - Why didn’t this help on this example?
- Lab 4 will have a state machine that should show the tradeoffs better.
Modified One-Hot

- In modified one-hot, the reset sequence is all zeros which transitions to the first state at the first clock edge.
- Needed for FPGAs whose FF’s powerup as zeros.

Don’t Forget *Synchronous* Resets!!

- The Xilinx FPGAs are designed so that on powerup, the DFFs initialize to logic 0.
  - We do not want to depend on that!!
- If your library supports it, use one that has a synchronous reset and tie it to the global reset pin.
- Else, explicitly design reset signal into your FSM.
- Can gate the reset signal to parts of the design on other events.