L10 – additional State Machine examples
States Machine Design

- Other topics on state machine design
  - Examples that are Equivalent
  - More one hot examples

- Ref: text Unit 15.4, 15.5, 15.8
Equivalent State Machines

- So far have seen that equivalent states in the state table of a sequential machine are equivalent and can be removed.

- How about the equivalence between two sequential circuits?
  
  Two sequential circuits are equivalent if they are capable of doing the same work.
Equivalence of two machines

- Consider a sequential machine
  - (ref lect 6) – a machine to detect 101 results in a state graph of with state table
Now consider a second machine

- Consider a machine to detect 010, the complement of the other machine.
- Not yet developed - A is the starting state where any number of 1’s has been received.
State A and state B

- State A is starting state.
  - On a 1 stay there
  - On a 0 go to state B

- State B
  - On a 0 stay there
  - On a 1 now have 01 so go to C
State C

- In state C
  - On a 0 have 010 as last 3, output a 1 and go back to B.
  - On a 1 have a 1 as last input and return to A
Machine and state table

- Can generate a state table for this 010 sequential machine

![State Diagram]

<table>
<thead>
<tr>
<th>Pr state</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X=0</td>
<td>X=1</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td>A</td>
</tr>
</tbody>
</table>

9/2/2012 – ECE 3561 Lect
10
Copyright 2012 - Joanne DeGroat, ECE, OSU
8
Question

- Are the sequential circuits to detect 010 and 101 as the last 3 equivalent?
Look at next state tables

- The tables for the two machines are and develop an equivalence implication table.
Start on table

- Mark output incompatible states
  - S2-A,B,C   C-S0,S1,S2
Now the other blocks

- Implied next states

<table>
<thead>
<tr>
<th>Pr state</th>
<th>Next State</th>
<th>Output</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X=0  X=1</td>
<td>X=0 X=1</td>
<td></td>
</tr>
<tr>
<td>S0</td>
<td>S0  S1</td>
<td>0   0</td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>S2  S1</td>
<td>0   0</td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>S0  S1</td>
<td>0   1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pr state</th>
<th>Next State</th>
<th>Output</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X=0  X=1</td>
<td>X=0 X=1</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>B   A</td>
<td>0   0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>B   C</td>
<td>0   0</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>B   A</td>
<td>1   0</td>
<td></td>
</tr>
</tbody>
</table>

S0  S0-B  S0-B
S1-A  S1-C
S2-B  S2-B
S1-A  S1-C

A  B  C
Check implied next states

- On pass 1 remove
  - S1-C and S2-B

<table>
<thead>
<tr>
<th>Pr state</th>
<th>Next State X=0</th>
<th>X=1</th>
<th>Output X=0</th>
<th>X=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>S0</td>
<td>S1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S1</td>
<td>S2</td>
<td>S1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>S0</td>
<td>S1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pr state</th>
<th>Next State X=0</th>
<th>X=1</th>
<th>Output X=0</th>
<th>X=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>A</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>C</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td>A</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Check implied next states

- On pass 2 remove
  - S1-A
  - Now all blocks are Xed

- What does this tell us?
  - The machines are incompatible.
  - To be compatible one block in each row and one block in each column need to be un-Xed and only one.
  - If more than one – says machine incompatible or one or both not minimal
An example

- State tables and state graphs of two sequential machines. Figure 15-6 from the text.
- Equivalent?
Proving equivalence

- Again will use an implication table.
  - Only this time, it is the full square.
  - Along bottom are the states of one machine
  - Along the side are the states of the second.
- Start by removing output incompatible states.
The equivalence implication table

- X squares where the outputs are incompatible
- Enter implied equivalence pairs for remaining states.
Step 2

- Go back through and remove the implied equivalence pairs that were Xed on the first pass. Continue until no further Xs are entered.

- If there is one square not Xed in each row and each column, the state machines are equivalent. (When both are minimal)

- Consider problem 15-17 in text. Does this work if the state tables are of different size?
Problem 15.17

- The problem statement

15.17 Circuits $N$ and $M$ have the state tables that follow.

(a) Without first reducing the tables, determine whether circuits $N$ and $M$ are equivalent.

(b) Reduce each table to a minimum number of states, and then show that $N$ is equivalent to $M$ by inspecting the reduced tables.

<table>
<thead>
<tr>
<th>$M$</th>
<th>$X = 0$</th>
<th>$X = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0$</td>
<td>$S_3$</td>
<td>$S_1$</td>
</tr>
<tr>
<td>$S_1$</td>
<td>$S_0$</td>
<td>$S_1$</td>
</tr>
<tr>
<td>$S_2$</td>
<td>$S_0$</td>
<td>$S_2$</td>
</tr>
<tr>
<td>$S_3$</td>
<td>$S_0$</td>
<td>$S_3$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$N$</th>
<th>$X = 0$</th>
<th>$X = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$E$</td>
<td>$A$</td>
</tr>
<tr>
<td>$B$</td>
<td>$F$</td>
<td>$B$</td>
</tr>
<tr>
<td>$C$</td>
<td>$E$</td>
<td>$D$</td>
</tr>
<tr>
<td>$D$</td>
<td>$E$</td>
<td>$C$</td>
</tr>
<tr>
<td>$E$</td>
<td>$B$</td>
<td>$D$</td>
</tr>
<tr>
<td>$F$</td>
<td>$B$</td>
<td>$C$</td>
</tr>
</tbody>
</table>
Minimize Sx machine

☐ Can it be reduced?

☐ YES
Reduced machine

- States S2 and S3 are equivalent – in fact S2 is not reachable unless the machine comes up in that state at startup and it can never reach S2 again.
Significance

Now consider the equivalence implication table. What is the implication if S2 replaces state S2 and S3?
Incompletely Specified

- Incompletely Specified State Tables
  - State tables that contain don’t cares.
  - Results in reduced logic

- Determining the best way to fill in the don’t cares is another of the \( n-p \) complete problems.

- For this course do the most logical approach.
One Hot

- CPLDs and FPGAs have a good number of F/Fs onboard. The F/Fs are there whether they are used or not, so a circuit with the minimum number of F/Fs is not the ultimate objective.

- For these devices the objective is to reduce the total number of logic cells used and the interconnection between cells.

- One hot encoding is one approach to have shorter signal paths and reduce logic cells.
What is one hot?

- One hot is a method where a flip flop is used for each state in the state machine. A state machine with $n$ states will require $n$ flip flops in its realization.
- One hot realization is excellent for controllers that step through a set sequence of linear steps.
- Text gives example of a multiplier controller state graph which is not linear.
The full circuit

- Desire $Z=1$ when $X_{-1} \cdot X_{-2} \cdot X_{-3}$ is 101.
- Simply construct the combinational logic with inputs from the F/F outputs.
Compare the gates

- Traditional implementation for sequence detector (from text)
  - 2 F/Fs
  - 2 2-input AND gates
  - 1 INV

- One hot implementation
  - 3 F/Fs
  - 1 3-input AND gate
Another example

- Design a circuit to detect when the value represents a BCD digit
- Could also detect when it is not a valid BCD digit, i.e., 10, 11, 12, 13, 14, 15 which is
  - 1010, 1011, 1100, 1101, 1110, 1111 or
  - 101x, 11xx
- As last 4 inputs
BCD digit detector

- The logic
- In groups of 4
- Detect when
  - 101x
  - 11xx
- Z=1 valid BCD
Lecture summary

- Have again looked at state machine equivalence.
- One hot encoding and it is interesting for many implementations.