L10 – State Machine Design Topics
States Machine Design

- Other topics on state machine design
  - Equivalent sequential machines
  - Incompletely specified machines
  - One Hot State Machines

- 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.
Formally

- **Definition 15.2**
  - Sequential circuit $N_1$ is equivalent to sequential circuit $N_2$ if for each state $p$ in $N_1$, there is a state $q$ in $N_2$ such that $p \equiv q$, and conversely, for each state $s$ in $N_2$ there is a state $t$ in $N_1$ such that $s \equiv t$.
  - Simply said they have the same states which can be seen if the circuit is small enough.

- An implication table can be used to prove this.
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>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>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>
Problem 15.17

- Can be Worked on board
- Or here in the slides

Start with an equivalence implication table

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Output compatible

Go through and X output incompatible states

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>S3</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Next State

- Fill in the next state pairs on the table

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>×</td>
<td>×</td>
<td>S3 – E</td>
<td>S3 – E</td>
<td>S3 – B</td>
<td>S3 – B</td>
</tr>
<tr>
<td></td>
<td>S1 – D</td>
<td>S1 – C</td>
<td>S1 – D</td>
<td>S1 – C</td>
<td>S1 – C</td>
<td>S1 – C</td>
</tr>
<tr>
<td>S1</td>
<td>×</td>
<td>×</td>
<td>S0 – E</td>
<td>S0 – E</td>
<td>S0 – B</td>
<td>S0 – B</td>
</tr>
<tr>
<td></td>
<td>S1 – D</td>
<td>S1 – C</td>
<td>S1 – D</td>
<td>S1 – C</td>
<td>S1 – C</td>
<td>S1 – C</td>
</tr>
<tr>
<td>S2</td>
<td>S0 – E</td>
<td>S0 – F</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>S2 – A</td>
<td>S2 – B</td>
<td>S2 – A</td>
<td>S2 – B</td>
<td>S2 – A</td>
<td>S2 – B</td>
</tr>
<tr>
<td>S3</td>
<td>S0 – E</td>
<td>S0 – F</td>
<td>S3 – E</td>
<td>S3 – E</td>
<td>S3 – B</td>
<td>S3 – B</td>
</tr>
<tr>
<td></td>
<td>S3 – A</td>
<td>S3 – B</td>
<td>S3 – A</td>
<td>S3 – B</td>
<td>S3 – A</td>
<td>S3 – B</td>
</tr>
</tbody>
</table>
1st Pass through table

- Check implied pairs (S3-E)x (S0-B)x
  - Remainder are compatible. It seems machine has redundant states
Minimize both machines?

- Start with the Sx machine – can it be minimized? If so, what are implications?

```
S1
S2
S3
S0 S1 S2
```

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></th>
<th>$X = 0$</th>
<th>$X = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0$</td>
<td>$S_1$</td>
<td>$S_2$</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>
<tr>
<td>$S_4$</td>
<td>$S_0$</td>
<td></td>
</tr>
<tr>
<td>$S_5$</td>
<td>$S_0$</td>
<td></td>
</tr>
</tbody>
</table>
Minimize both machines

- **Implied Next States**

---

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.
Minimize both machines

☐ Can it be reduced?
Reduced machine

- Can be seen from state graph
- 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.
Now for the A,B,..,E machine

- Start with incompatible outputs
And then implied next state

- And run though algorithm
Result is reduced state table

- Find that
  - \(D \equiv C\)
  - \(E \equiv F\)
  - \(A \equiv B\)

- So state table reduces to AND for \(Sx\) version
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.
Linear one hot

- Linear one hot sequential controllers requires no next state logic.
- On Reset the output of the 4 F/F is 1000
  - On clocks 0100, then 0010, then 0001, then 0000
One hot use

- Have been use in such things as
  - Successive approximation A-D converters
  - Various automotive control systems
  - Automated machinery control systems

- Also commonly used in processor controllers
  - Process controller states
    - F1,F2,F3,F4,F5,F6,F7,F8 always followed by
    - E0,E1,E2,E3,E4,E5,E6,E7 if direct addressing
    - E20,E21,E22,E23,E24,E25,E26,E27 if indirect
One hot application

- One hot could have been used in the sequence detector problems
  - Detect an input sequence ending in 101.
  - Construct a shift register that holds the last 3 inputs of an input X.
The full circuit

- Desire $Z=1$ when $X_{-1} X_{-2} X_{-3}$ 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 sequence detector that detects input sequences ending in 010 or 1001. \( Z = 1 \) when a sequence is detected.
- Start with a 4 bit shift register to hold the last 4 inputs.
Now add $Z$ generation logic

- Construct the combinational logic for $Z$
Implementation comparison

- **Traditional**
  - 3 F/Fs
  - Need to work problem
  - more than 1 hot

- **One hot**
  - 4 F/Fs
  - 2 AND gates (1-3 inp, 1-4 inp)
  - 1 OR gate (2 inp)
The state table

- For the Mealy Machine

![State Machine Diagram]

<table>
<thead>
<tr>
<th>Present State</th>
<th>X=0</th>
<th>X=1</th>
<th>X=0</th>
<th>X=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>S1</td>
<td>S4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S1</td>
<td>S1</td>
<td>S2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>S3</td>
<td>S4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>S5</td>
<td>S2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>S3</td>
<td>S4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S5</td>
<td>S1</td>
<td>S2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Comparison

- This was worked to gates in lect 9.
  - 3 D F/Fs
  - 2 – 3 input AND gates
  - 3 – 2 input AND gates
  - 2 – 2 input OR gates
  - 1 – 3 input OR gate

- Versus one hot
  - 4 F/Fs
  - 1 – 3 input AND gate
  - 1 – 4 input AND gate
  - 1 – 2 input OR

- Comments from class??????
Lecture summary

- Have looked at state machine equivalence.
- Incompletely specified machine implication.
- One hot encoding and how it may not be all that bad an alternative.