ECE 2300
Digital Logic & Computer Organization
Spring 2024

More Finite State Machines
Announcements

• Lab 2a due tomorrow

• Prelim 1
  – Tuesday March 5th, 1:25-2:40pm in class
    • closed book, closed notes, closed Internet
  – Coverage: Lectures 1~7, first part of Lecture 8
    • Binary number, Boolean algebra, CMOS, combinational logic, sequential logic, and Verilog
  – A sample prelim exam will be posted tomorrow
  – TA-led review session will be scheduled (& recorded)
Recap: Procedural Assignments

```
reg Y, Z;
always @(posedge clk)
begins
    Y = A & B;
    Z = Y;
end

Blocking assignments

reg Y, Z;
always @(posedge clk)
begins
    Y <= A & B;
    Z <= Y;
end

Nonblocking assignments
```

**Simulator Interpretation**

\[
\begin{align*}
Y_{\text{next}} & \leftarrow A \& B \\
Z_{\text{next}} & \leftarrow Y_{\text{next}} = A \& B // \text{use “new”} \ Y
\end{align*}
\]

- RHS evaluated \textit{sequentially}
- Assignment to LHS is \textit{immediate}

\[
\begin{align*}
Z_{\text{next}} & \leftarrow Y // \text{use “old”} \ Y \\
Y_{\text{next}} & \leftarrow A \& B
\end{align*}
\]

- RHS evaluated \textit{in parallel} (order doesn’t matter)
- Assignment to LHS is \textit{delayed} until the end of the always block
Recap: Procedural Assignments

reg Y, Z;
always @ (posedge clk)
begin
    Y = A & B;
    Z = Y;
end

**Blocking assignments**

reg Y, Z;
always @ (posedge clk)
begin
    Y <= A & B;
    Z <= Y;
end

**Nonblocking assignments**

Actual Circuit
(Synthesizer Interpretation)

When a reg ("Y" here) is assigned in a blocking assignment ("Y=A&B"), employ its input ("A&B") for connection in RHS of a subsequent assignment ("Z=Y")

When a reg ("Y" here) is assigned in a nonblocking assignment ("Y=A&B"), employ its output for connection in RHS of another assignment ("Z=Y")
Review: Finite State Machine (FSM)

- An FSM is an *abstract representation* of a sequential circuit
  1. A finite number of inputs;
  2. A finite number of outputs;
  3. A finite number of states;
  4. A specification of all state transitions.

Can be described by a state diagram

Moore FSM Example

Mealy FSM Example
Review:
Moore Machine

Outputs only depend on current state value
Review: Mealy Machine

Outputs only depend on input and current state value
FSM Design Procedure

(1) Understand the problem statement and determine inputs and outputs

(2) Identify states and create a state diagram

(3) Determine the number of required FFs

(4) Implement combinational logic for outputs and next state

(5) Simulate the circuit to test its operation
Example FSM: Pattern Detector

- Monitors the input, and outputs a 1 whenever a specified input pattern is detected.

- Example: Output a 1 whenever 111 is detected on the input over 3 consecutive clock cycles.
  - Overlapping patterns also detected (1111...)

- Input \textit{In} (one bit)
- Output \textit{Out} (one bit)
- \textit{Reset} causes FSM to start in initial state
- \textit{Clock} input not shown (always present)
Transition/Output Table

• Shows the next state ($S^*$) and output values for each combination of current state ($S$) and inputs

• Used to derive the minimized state transition ($S^*$) and output Boolean equations
Moore Transition/Output Table 1

Current State (S) | Next State (S*) | Out
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In = 0</td>
<td>In = 1</td>
</tr>
<tr>
<td>Init</td>
<td>Init</td>
<td>Got1</td>
</tr>
<tr>
<td>Got1</td>
<td>Init</td>
<td>Got11</td>
</tr>
<tr>
<td>Got11</td>
<td>Init</td>
<td>Got111</td>
</tr>
<tr>
<td>Got111</td>
<td>Init</td>
<td>Got111</td>
</tr>
</tbody>
</table>

• Version 1: uses descriptive state names
Moore Transition/Output Table 2

```
<table>
<thead>
<tr>
<th>S_1 S_0</th>
<th>S_1^* S_0^*</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In = 0</td>
<td>In = 1</td>
</tr>
<tr>
<td>0 0</td>
<td>0 0</td>
<td>0 1</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0</td>
<td>1 0</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0</td>
<td>1 1</td>
</tr>
<tr>
<td>1 1</td>
<td>0 0</td>
<td>1 1</td>
</tr>
</tbody>
</table>
```

- Version 2: uses state binary encodings
Minimized Equations for $S^*$ and Out

\[ \text{Out} = S_1 \cdot S_0 \]

<table>
<thead>
<tr>
<th>$S_1 \ S_0$</th>
<th>$S_1^* \ S_0^*$</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\text{In} = 0$</td>
<td>$\text{In} = 1$</td>
</tr>
<tr>
<td>0 0</td>
<td>0 0</td>
<td>0 1</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0</td>
<td>1 0</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0</td>
<td>1 1</td>
</tr>
<tr>
<td>1 1</td>
<td>0 0</td>
<td>1 1</td>
</tr>
</tbody>
</table>
Minimized Equations for $S^*$ and $Out$
Mealy Transition/Output Table 1

<table>
<thead>
<tr>
<th>Current State (S)</th>
<th>Next State (S*), Out</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In = 0</td>
</tr>
<tr>
<td>Init</td>
<td>Init, 0</td>
</tr>
<tr>
<td>Got1</td>
<td>Init, 0</td>
</tr>
<tr>
<td>Got11</td>
<td>Init, 0</td>
</tr>
</tbody>
</table>

• Version 1: uses descriptive state names

Lecture 9: 16
Mealy Transition/Output Table 2

- Version 2: uses state binary encodings

<table>
<thead>
<tr>
<th>$S_1$ $S_0$</th>
<th>$S_1^* S_0^*$, Out</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>0 0</strong></td>
<td><strong>0 0, 0</strong></td>
</tr>
<tr>
<td><strong>0 1</strong></td>
<td><strong>0 0, 0</strong></td>
</tr>
<tr>
<td><strong>1 0</strong></td>
<td><strong>0 0, 0</strong></td>
</tr>
</tbody>
</table>
Minimized Equations for $S^*$ and $Out$

<table>
<thead>
<tr>
<th>$S1$ $S0$</th>
<th>$S1^<em>$ $S0^</em>$, $Out$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$In = 0$</td>
</tr>
<tr>
<td>0 0</td>
<td>0 0, 0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0, 0</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0, 0</td>
</tr>
</tbody>
</table>
Minimized Equations for $S^*$ and $Out$

$S_1^* = S_0 \cdot In + S_1 \cdot In$

$S_0^* = S_1^* \cdot S_0^* \cdot In$

Out = $S_1 \cdot In$

<table>
<thead>
<tr>
<th>$S_1$ $S_0$</th>
<th>$S1^<em>$ $S0^</em>$, Out</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In = 0</td>
</tr>
<tr>
<td>0 0</td>
<td>0 0, 0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0, 0</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0, 0</td>
</tr>
</tbody>
</table>
FSMs in Verilog

<module statement>
<input and output declarations>

<reg declarations>

<parameter or typedef statement>

<always block for next state>

<always block for output>

<always block for state FFs>

endmodule
module PatDetectMoore (Clk, In, Reset, Out);
input Clk, In, Reset;
output Out;

reg Out;
reg [1:0] Scurr, Snext;

parameter [1:0] Init = 2'b00,
    Got1 = 2'b01,
    Got11 = 2'b10,
    Got111 = 2'b11;
Moore FSM in Verilog

always @ (In, Scurr)
begin
  case (Scurr)
    Init: if (In == 1) Snext = Got1; else Snext = Init;
    Got1: if (In == 1) Snext = Got11; else Snext = Init;
    Got11: if (In == 1) Snext = Got111; else Snext = Init;
    Got111: if (In == 1) Snext = Got111; else Snext = Init;
    default: Snext = Init;
  endcase
  Snext = Init;
end
Moore FSM in Verilog

always @ (Scurr)
    if (Scurr == Got111) Out = 1;
    else Out = 0;

always @ (posedge Clk)
    if (Reset == 1) Scurr <= Init;
    else Scurr <= Snex;

endmodule
Moore FSM Verilog Simulation
module PatDetectMealy (Clk, In, Reset, Out);
input Clk, In, Reset;
output Out;

reg Out;
reg [1:0] Scurr, Snext;

parameter [1:0] Init = 2'b00,
                Got1 = 2'b01,
                Got11 = 2'b10;
always @ (In, Scurr)
bEGIN
    case (Scurr)
        Init: if (In == 1) Snext = Got1; else Snext = Init;
        Got1: if (In == 1) Snext = Got11; else Snext = Init;
        Got11: if (In == 1) Snext = Got11; else Snext = Init;
        default: Snext = Init;
    endcase
end

next state
comb logic
Lecture 9:

Mealy FSM in Verilog

```verilog
always @ (Scurr, In)
  if ((Scurr == Got11) && (In == 1)) Out = 1;
  else Out = 0;

always @ (posedge Clk)
  if (Reset == 1) Scurr <= Init;
  else Scurr <= Snext;

endmodule
```

output comb logic

update state FFs
(current state)
Pushbutton Lock: Moore State Diagram

- Output: UL=1 with sequence X1, X2, X2
- Input: 00 (neither), 10 (X1), 01 (X2), 11 (reset)
## Moore Transition/Output Table 1

<table>
<thead>
<tr>
<th>Current State (S)</th>
<th>Next State (S*)</th>
<th>Input 0 0 (neither)</th>
<th>Input 0 1 (X2)</th>
<th>Input 1 0 (X1)</th>
<th>Input 1 1 (reset)</th>
<th>UL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init</td>
<td>Init</td>
<td>Init</td>
<td>X1</td>
<td>Init</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>X1</td>
<td>X1-X2</td>
<td>Init</td>
<td>Init</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>X1-X2</td>
<td>X1,X2</td>
<td>X1-X2-X2</td>
<td>Init</td>
<td>Init</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>X1-X2-X2</td>
<td>X1-X2-X2</td>
<td>X1-X2-X2</td>
<td>X1-X2-X2</td>
<td>Init</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- **Version 1**: uses descriptive state names
Moore Transition/Output Table 2

<table>
<thead>
<tr>
<th>$S_1$ $S_0$</th>
<th>$S_1^<em>$ $S_0^</em>$</th>
<th>UL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1</td>
<td>1 0</td>
</tr>
<tr>
<td>1 0</td>
<td>1 0</td>
<td>1 1</td>
</tr>
<tr>
<td>1 1</td>
<td>1 1</td>
<td>0 0</td>
</tr>
</tbody>
</table>

• Version 2: uses state binary encodings
Pushbutton Lock: Mealy State Diagram

- **Output**: UL=1 with sequence X1, X2, X2
- **Input**: 00 (neither), 10 (X1), 01 (X2), 11 (reset)
## Mealy Transition/Output Table

<table>
<thead>
<tr>
<th>$S_1$ $S_0$</th>
<th>$S_1^<em>$ $S_0^</em>$, UL</th>
</tr>
</thead>
</table>
| $\begin{array}{c|cccc}
   \text{Input} & 00 & 01 & 10 & 11 \\
\hline
   00 & 00,0 & 00,0 & 01,0 & 00,0 \\
   01 & 01,0 & 10,0 & 00,0 & 00,0 \\
   10 & 10,0 & 11,1 & 00,0 & 00,0 \\
   11 & 11,1 & 11,1 & 11,1 & 00,0 \\
\end{array}$ |

- uses state binary encodings

Lecture 9: 33
Analyzing the Sequential Logic

What does this circuit do?

Write down transition and output equations

$S_0^* =$

$S_1^* =$

$\text{Out} =$
Reconstruct State Diagram

- Complete the transition/output table and state diagram
- Identify the functionality of the FSM

\[ S_0^* = \text{In}' \]
\[ S_1^* = \text{In}' \cdot S_1 \cdot S_0' + \text{In} \cdot S_0 \]
\[ \text{Out} = S_1 \cdot S_0 \]

<table>
<thead>
<tr>
<th>( S_1 ) ( S_0 )</th>
<th>( S_1^* ) ( S_0^* )</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0 1</td>
<td>0 0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1</td>
<td>1 0</td>
</tr>
<tr>
<td>1 0</td>
<td></td>
<td>0 1</td>
</tr>
<tr>
<td>1 1</td>
<td>0 1</td>
<td>1 0</td>
</tr>
</tbody>
</table>

Lecture 9: 35
Another Pattern Detector

Detects 010 (overlapping patterns included)

$S_0^* = \overline{In}$

$S_1^* = \overline{In} \cdot S_1 \cdot S_0^* + \overline{In} \cdot S_0$

Out = $S_1 \cdot S_0$

<table>
<thead>
<tr>
<th>$S_1$ $S_0$</th>
<th>$S_1^<em>$ $S_0^</em>$</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0 1</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1</td>
<td>10</td>
</tr>
<tr>
<td>1 0</td>
<td>1 1</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>0 1</td>
<td>1</td>
</tr>
</tbody>
</table>
Moore State Diagram for DFF

- Input: D
- Output: Q

<table>
<thead>
<tr>
<th>S</th>
<th>S*</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Mealy State Diagram for DFF?

- **Input:** D
- **Output:** Q
Before Next Class

• H&H 2.9, 4.6, 4.9

Next Time

More FSMs
Timing
Clocking