Agenda

- Summary of Lab1
  - Common issues
  - Wisely use lab resources
- Hints for Lab2
- Lab3: Finite State Machine
  - Lab assignment
  - Pattern Recognizer
  - Design flow
  - Tutorial on a simple design
Common issues for Lab1

- Schematic entry
  - Wires cannot be connected into the input port
  - Wires cannot be connected out from the output port
- Test bench
  - Generate Expected Simulation Results
- Simulation waveforms in ModelSim
  - Please show the full name of signals
  - Correct the errors
  - Please mark the time delay using “insert cursor”
- Keep consistency of the schematic diagram, test bench and simulation waveforms
Common issues for Lab1

- Worst case delay

4-bit adder

worst case delay: B0 to Co
Common issues for Lab1

- Worst case delay
  - Find some input combination
  - Only changing B0 at time t1
  - At time t2, Co changes only because B0’s toggling goes through the longest path
  - $T = t2 - t1$ is the worst case delay

- $A = 1111, B = 0000 \rightarrow 0001, C_i = 0$
  $\Rightarrow Co = 0 \rightarrow 1$
Common issues for Lab1
Lab resources

- Online lecture notes, tutorial, Q&A
- Lab hours
- Webboard
Hints for Lab2

- Design Flow

  - Schematic entry
  - Behavior simulation
  - Implement design
  - Post place&route simulation

- Design the circuit
  - Combinational Logic
  - Flip-Flops in Xilinx

- Verify the function
  - Setup clock Information in test bench

- Timing analysis

- Worst case analysis
Hints for Lab2

- Use the given Flip-Flops and gates
  - Three different Flip-Flops will be used: FDCE, FTCE, FDPE
- In the post place & route simulation, the Flip-Flops will not function during the first 100ns of simulation time
  - The Flip-Flops in a Xilinx chip are held in reset/preset for the first 100ns.
- Complete all tasks in the assignment
Lab3 Assignment

- Design a pattern recognizer
  - Finite state machine
    - Mealy machine: produces an output for each transition
    - Moore machine: produces an output for each state
  - Assignment different encodings for the states
- Comparison on
  - # states, # FFs, # logic blocks,
  - # lines in VHDL codes
When \( x(t-2, t-1, t) = "011" \) or \( "011" \), \( z = 1 \); Otherwise \( z = 0 \);

\[
\begin{array}{cccccccccccccc}
  t & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\
  x & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
  z & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
\end{array}
\]
When $x(t-2, t-1, t) = "011"$ or "$011", $z = 1$;
Otherwise $z = 0$;
Design Flow

- Input state diagram
- Layout of the design
- Verify the function
Tutorial on a simple design

- The pattern recognizer for tutorial
  - Recognize 110 and 101

S0: input pattern ‘0’
Tutorial on a simple design

- The pattern recognizer for tutorial
  - Recognize 110 and 101

States
S0: input pattern ‘0’
S1: input pattern ‘1’
Tutorial on a simple design

- The pattern recognizer for tutorial
  - Recognize 110 and 101

States
S0: input pattern ‘0’
S1: input pattern ‘1’

<table>
<thead>
<tr>
<th>t</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S2: input pattern ‘11’
Tutorial on a simple design

- The pattern recognizer for tutorial
  - Recognize 110 and 101

States
- S0: input pattern ‘0’
- S1: input pattern ‘1’
- S2: input pattern ‘11’

<table>
<thead>
<tr>
<th>t</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S3: input pattern ‘110’
Tutorial on a simple design

- The pattern recognizer for tutorial
  - Recognize 110 and 101

States
- S0: input pattern ‘0’
- S1: input pattern ‘1’
- S2: input pattern ‘11’
- S3: input pattern ‘110’

<table>
<thead>
<tr>
<th>t</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S4: input pattern ‘101’
Tutorial on a simple design

- The pattern recognizer for tutorial
  - Recognize 110 and 101

States
- S0: input pattern ‘0’
- S1: input pattern ‘1’
- S2: input pattern ‘11’
- S3: input pattern ‘110’
- S4: input pattern ‘101’

<table>
<thead>
<tr>
<th>t</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S5: input pattern ‘10’
Tutorial on a simple design

- The pattern recognizer for tutorial
  - Recognize 110 and 101

<table>
<thead>
<tr>
<th>t</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S0: input pattern ‘0’
S1: input pattern ‘1’
S2: input pattern ‘11’
S3: input pattern ‘110’
S4: input pattern ‘101’
S5: input pattern ‘10’
Tutorial on a simple design

- The pattern recognizer for tutorial
  - Recognize 110 and 101

States
S0: input pattern ‘0’
S1: input pattern ‘1’
S2: input pattern ‘11’
S3: input pattern ‘110’
S4: input pattern ‘101’
S5: input pattern ‘10’

<table>
<thead>
<tr>
<th>t</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S1: input pattern ‘1’
Tutorial on a simple design

- The pattern recognizer for tutorial
  - Recognize 110 and 101

States
- S0: input pattern ‘0’
- S1: input pattern ‘1’
- S2: input pattern ‘11’
- S3: input pattern ‘110’
- S4: input pattern ‘101’
- S5: input pattern ‘10’

<table>
<thead>
<tr>
<th>t</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

S5: input pattern ‘10’
Tutorial on a simple design

- The pattern recognizer for tutorial
  - Recognize 110 and 101

States
- S0: input pattern ‘0’
- S1: input pattern ‘1’
- S2: input pattern ‘11’
- S3: input pattern ‘110’
- S4: input pattern ‘101’
- S5: input pattern ‘10’

<table>
<thead>
<tr>
<th>t</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

S4: input pattern ‘101’
Tutorial on a simple design

- The pattern recognizer for tutorial
  - Recognize 110 and 101

States
- S0: input pattern ‘0’
- S1: input pattern ‘1’
- S2: input pattern ‘11’
- S3: input pattern ‘110’
- S4: input pattern ‘101’
- S5: input pattern ‘10’

<table>
<thead>
<tr>
<th>t</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

S5: input pattern ‘10’
State Diagram I

Moore Machine
State Diagram II

Mealy Machine
State Diagram II

Mealy Machine
State Diagram II

Mealy Machine
State Diagram II

Mealy Machine
State Diagram II

Mealy Machine
State Diagram II

Mealy Machine
State Diagram II

Mealy Machine
States Encoding

<table>
<thead>
<tr>
<th></th>
<th>S0</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code1</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Code2</td>
<td>11</td>
<td>10</td>
<td>01</td>
<td>00</td>
</tr>
</tbody>
</table>
Manual State Assignment

- Modify the VHDL program with state assignments.

The VHDL codes look like the following:

```
ARCHITECTURE BEHAVIOR OF MEALY IS
  TYPE type_sreg IS (s0,s1,s2,s3);
  attribute enum_encoding : string;
  attribute enum_encoding of type_sreg : type is "00 01 10 11";
  SIGNAL sreg, next_sreg : type_sreg;
BEGIN
...```

Tutorial Design

- Input Mealy/Moor state diagram in StateCAD
- Generate VHDL codes
- Implement VHDL in Xilinx
  - Synthesis
  - Mapping
  - Place and Route
  - Layout checking
- Simulate in ModelSim