CSE 140L Spring 2010
Lab 3 Assignment

Logistics for Lab 3:

This lab should be done in groups of maximum 2 students. Use Xilinx tools version 10.1.

What is due:

- **Report:**
  - Submit a report in .pdf form via email to the TAs at cse140lsp10@gmail.com by the beginning of the class on May 12th, 2010. The subject line of the email should read "CSE140L Lab 3 Submission – YourLastName-FirstName". Emails sent after the class begins on the day the lab is due will not be accepted. Please include in the report all your Verilog code, diagrams, simulation results as requested in the lab assignment. Turn in only one PDF file containing all the solutions with your Full Name and PID on the first page of the report.

- **Demo:**
  - The TAs will conduct demos from Wednesday through Saturday (May 12th to May 15th for the lab3) at B250 Basement of the CSE building. You need to sign up for your demo time slot before the beginning on class on Wednesday when the lab is due. The signup is in a excel sheet available at: https://spreadsheets.google.com/ccc?key=0Avg6ufR5fmA5dEQ5aWFCSGZrR1g0Q0dmV1E4MXhPeHc&hl=en. Write your full names (two students per slot) in the slot. You can use either your laptop or the computers available at the location of the demo. Make sure your demo is ready to run by the beginning of your time slot as you will have maximum 10min to show your work and answer any questions related to it.
**Part 1: BCD and Binary Adders**

Design a 4-bit binary or BCD adder with select signal (Sel). When Sel = 0, the circuit performs BCD addition and for Sel = 1, the circuit outputs binary addition. Set overflow bit to 1 if the result of addition is bigger than the 4 bits, otherwise overflow is zero. Do not use arithmetic + to implement addition. Report should include:

- Design steps of both the adders.
- Behavioral Verilog module of the entire circuit
- Include screen snapshot of the simulator outcome for the following input combinations:

<table>
<thead>
<tr>
<th>Sel</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>6</td>
</tr>
</tbody>
</table>

**Part 2 State Reduction:**

For a state diagram in Figure Ex. 8.5 on page 394 in your text book, draw a fully reduced state diagram and implement the reduced FSM using behavioral Verilog.

Report should include:

- Finite state machine for your design. What strings cause the recognizer to output a 1?
- Your verilog code (well commented).
- Screenshot of a simulation results for the following consecutive inputs: 0111000011

**Part 3:**

Construct a Finite State Machine with 9 states (say S0 to S8), two inputs (left_in and right_in), a reset and two outputs (left_most and right_most). On reset, the machine starts in the middle of the nine states (S4). When left_in is asserted, it transitions one state to the left on the clock edge. If it reaches extreme left end of the chain, output left_most is asserted and the machine stays in the same state if left_in is still asserted. When right_in is asserted, it transitions one state to the right on each clock edge. If it reaches the extreme right end of the chain, output right_most is asserted and the machine stays in the same state if right_in is still asserted. Design an FSM with the above specifications, implement in Verilog and test its behavior via simulation.

Report should include:

- Finite state machine for your design.
- Your verilog code (well commented).
- Screenshot of simulation showing outputs (left_most and right_most) for following test input sequence (changes on each clk cycle):
  - Reset is asserted for the first clock cycle. (clock cycle 1)
  - left_in: asserted for the next 6 clock cycles (clock cycle 2 to 7)
  - right_in: asserted for the next 12 clock cycles (clock cycle 8 to 19)
Part 4. Communicating FSMs: Gray code counter
This part involves two subparts.

Part 4a: Design a 3-bit up/down Gray counter using FSMs. When input down is zero, it works as the usual 3-bit gray code counter incrementing on each clock cycle (from 000 to 100). But when down is set to one, the counter ticks in reverse order of gray code (from 100 to 000). When rst is set to one, output should be initialized to 000.

Count sequence
down = 1'b0:  000 – 001 – 011 – 010 – 110 – 111 – 101 – 100 – 000 - 001 and so on.
Input: down, rst, one bit clk
Output: 3 bit count sequence Y[2:0]

Report should include:
   a) Single FSM with up/down gray count states.
   b) Behavioral Verilog module of the above circuit.
   c) Simulation screenshot of Y[2:0] for the following:
      down = 1'b1 for first 10 clock cycles.
   d) Simulation screenshot of Y[2:0] for the following:
      down = 1'b0 for the next 10 clock cycles.

Part 4b: Use two communicating FSMs (each FSM should have 5 states including the idle state) to design the same 3-bit up/down Gray counter. Report should include:
   a) Two Communicating FSMs implementing your up/down Gray code counter.
   b) Behavioral Verilog module of the circuit.
   c) Simulation screenshot of Y[2:0] for the following:
      down = 1'b1 for first 10 clock cycles.
   d) Simulation screenshot of Y[2:0] for the following:
      down = 1'b0 for the next 10 clock cycles.
Part 5. Sequence Recognizer:
Your circuit needs to recognize 1001, 1010 and 1011 sequences from a single bit input signal. Circuit outputs a 2 bit number on successful recognition of each sequence as follows:

<table>
<thead>
<tr>
<th>Input sequence</th>
<th>Output (Y[1:0])</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>01</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
</tr>
<tr>
<td>Else</td>
<td>00</td>
</tr>
</tbody>
</table>

For Eg: when sequence of input is: 0010 0101 0010 1101 0100 01

\[ \begin{align*}
\text{Y[1]}: & & 0000 & 0000 & 1000 & 0100 & 1010 \ 00 \\
\text{Y[0]}: & & 0000 & 0100 & 0010 & 0100 & 0000 \ 00
\end{align*} \]

Report should include:

A) Finite state machine for your design with minimum possible number of states.
B) Your verilog code (well commented).
C) Screenshot of simulation showing outputs for the following input sequence (changes on each clk cycle): 1100 1101 0110 0000

***Good Luck****