# CSE390C Autumn 2014 – Final Exam (3 December)

Name:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Max Score</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>2A</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>2B</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>2C</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
1. Combinational Logic (25 points)

What function is implemented by the multiplexer below. S2 is the most significant bit of the control signals, S0 is the least significant (e.g., S2=1, S1=1, S0=0 selects input #6).
2. Sequential Logic (75 points)

The following Verilog was found among old papers in a dusty drawer of a now defunct dot-com company. Unfortunately, there were no comments in the code.

```verilog
module Mystery (In, Clk, Reset, A, B, Out);
    input In, Clk, Reset;
    output A, B, Out;
    reg [5:0] state;
    reg [5:0] next_state;
    wire [2:0] count;

    parameter S0 = 6'b000000;
    parameter S1 = 6'b000001;
    parameter S2 = 6'b000010;
    parameter S3a = 6'b100011;
    parameter S3b = 6'b001011;
    parameter S4aa = 6'b100100;
    parameter S4ab = 6'b010100;
    parameter S4bb = 6'b001100;

    always @(posedge Clk) begin
        if (Reset) begin state = `S0; end
        else begin state = next_state; end
    end

    always @(In or state) begin
        case(state)
            `S0:  next_state = `S1;
            `S1:  next_state = `S2;
            `S2:  if (In) next_state = `S3b; else next_state = `S3a;
            `S3a: if (In) next_state = `S4ab; else next_state = `S4aa;
            `S3b: if (In) next_state = `S4bb; else next_state = `S4ab;
            `S4aa: next_state = `S0;
            `S4ab: next_state = `S0;
            `S4bb: next_state = `S0;
        endcase
    end

    assign count = state[2:0];
    assign A = state[5];
    assign B = state[3];
    assign Out = state[4];
endmodule
```
A. (25 points) Complete its state diagram and clearly label all transitions with the input value. For the values for A, B, and Out in each state, use Output = [A, B, Out]. You can write the triplet within the state bubble. Use the template below. Clearly label all transitions and outputs.
B. (25 points) Simulate the state machine for the following sample input waveforms. You are provided the values for the input signals, Reset and In. Fill in the details for the signals A, B, and Out. Also, please indicate the state the FSM is in for each clock cycle (i.e., write the symbolic name of the state in the space provided). Assume that the FSM is initially in an unknown state. Assume that all FFs are positive edge-triggered.
C. (25 points) When we look to see how this state machine was actually implemented, we find that there is a 3-bit binary counter in the circuit and 2 individual D flip-flops with some very logic around them. We need to figure out if we have an up to date description for the module. The Verilog describing the actual circuit implementation is below.

```verilog
module MysteryImplementation (In, Clk, Reset, A, B, Out);
  input   In, Clk, Reset;
  output  A, B, Out;
  reg     [2:0]counter, A, B;

  always @(posedge Clk) begin
    if (Reset | counter[2]) counter = 0;
    else counter = counter + 1;
  end

  always @(posedge Clk) begin
    if (Reset | counter[2]) A = 0;
    else A = A | (counter[1] & ~In);
  end

  always @(posedge Clk) begin
    if (Reset | counter[2]) B = 0;
    else B = B | (counter[1] & In);
  end

  assign Out = A & B;
endmodule
```

It turns out this is a correct implementation of the Mystery module? In fact, Mystery and MysteryImplementation describe the same state diagram. What is the correspondence of states between them? For each of the states of the Mystery module list the states for the sequential elements of the MysteryImplementation module – note that there are 3 state elements in the MysteryImplementation: the counter (3 bits), the A FF (1 bit), and the B FF (1 bit). Fill in the table below.

<table>
<thead>
<tr>
<th>Mystery</th>
<th>MysteryImplementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0 [5:0] = 6'b000000</td>
<td>counter [2:0] = 000</td>
</tr>
<tr>
<td>S1 [5:0] = 6'b000001</td>
<td></td>
</tr>
<tr>
<td>S2 [5:0] = 6'b000010</td>
<td></td>
</tr>
<tr>
<td>S3a [5:0] = 6'b100011</td>
<td></td>
</tr>
<tr>
<td>S3b [5:0] = 6'b001011</td>
<td></td>
</tr>
<tr>
<td>S4aa [5:0] = 6'b100100</td>
<td></td>
</tr>
<tr>
<td>S4ab [5:0] = 6'b010100</td>
<td></td>
</tr>
<tr>
<td>S4bb [5:0] = 6'b001100</td>
<td></td>
</tr>
</tbody>
</table>