UNCA CSCI 320
Exam 1 Spring 2014
6 March, 2014

This is a closed book and closed notes exam. It is to be turned in by 3:00 PM. Calculators, PDA's, cell phones, and any other electronic or communication devices may not be used during this exam.
If you want partial credit for imperfect answers, explain the reason for your answer!

Name:_____

Problem 1 (6 points) Truth table to Boolean expression
On the right below is a Boolean function. Write the equivalent boolean expression for this function on its left.

X	Y	Z	W
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Problem 2 (12 points) Truth table to SystemVerilog
Now write a SystemVerilog function that implements the truth table seen in Problem 1. You may just want to build on your Problem 1 answer.

```
module problem2(input  logic x, y, z
                output logic w) ;
```

Problem 3: Floating point to IEEE 754 (7.5 points)
How is the number 6.5 expressed in 32-bit IEEE 754 format?
Your answer should be written as an 8-digit hexadecimal number.
You must show your work to get credit for this problem.

Problem 4: IEEE 754 to floating point (7.5 points)
Consider `40200000` to be the hexadecimal representation of 32-bit IEEE floating point number. Translate that number into a C/Java-style floating point number. You must show your work to get credit for this problem.

Problem 5 (20 points) D Flip-flop
Below is a SystemVerilog module for a D flip-flop

```
module dflipflop(
          input  logic   clk,
          input  logic   d,
          output logic   q) ;
   always_ff @(posedge clk)
      q<= d ;
endmodule
```

Suppose you were debugging this circuit using `vsim`, like we did in the lab. What commands would you type (in `vsim`'s `transcript` window) to go through two complete clock cycles with this circuit. The data input should be 0 in the first clock cycle and 1 in the second clock cycle.

Draw a timing diagram, complete with the appropriate arrows, for the commands you wrote above.

Problem 6 (12 points) JK Flip Flop
This time, you are going to program a SystemVerilog module for a flip-flop, once popular in digital logic courses, that is rarely used today. The JK flip-flop has two "data" inputs, J and K; a clock input, CLK; and a single output, Q. Our JK flip-flop, like the old-fashion 74107 LSI chip, is triggered on the positive edge of the clock, just like the D flip-flop of the previous problem. The following "function" table explains how the JK flip-flop changes its output on the positive edge of the clock.

J	K	Action of flip-flop
0	0	Value of Q does not change.
0	1	Value of Q cleared to 0
1	0	Value of Q set to 1
1	1	Value of Q is toggled

By "toggled" we mean that: If the value of Q is 0, it becomes 1. If the value of Q is 1, it becomes 0.

Complete a SystemVerilog module to implement a JK flip-flop.

```
module jkflipflop(
          input  logic   clk,
          input  logic   j,
          input  logic   k
          output logic   q) ;
```

Problem 7 (15 points) FSM
Here's a finite state machine implemented as a SystemVerilog module.

```
module problem7(
          input  logic clk,
          input  logic reset,
          input  logic x,
          output logic y);
   typedef enum logic [1:0] {R0, R1, R2} statetype;
   statetype state, nextstate;
   always_ff @(posedge clk, posedge reset)
      if (reset) state <= R0;
      else       state <= nextstate;
   always_comb
      case (state)
        R0: if (x) nextstate <= R1;
            else   nextstate <= R0;
        R1: if (x) nextstate <= R0;
            else   nextstate <= R2;
        R2: if (x) nextstate <= R2;
            else   nextstate <= R1;
        default:   nextstate <= R0;
      endcase
   assign y = (state == R0);
endmodule
```

In the space below, draw a finite state machine diagram (that's the one where labeled arrows connect states) for this module.

Problem 8: Verilog FSM (20 points)
Using SystemVerilog, implement a finite state machine that implements the following finite state machine table. Your implementation should start in state S0 when reset.

State transitions		
Present state	Input	Next state
S0	0	S1
S0	1	S2
S1	0	S3
S1	1	S2
S2	0	S1
S2	1	S2
S3	0	S3
S3	1	S3

Outputs	
State	Output
S0	0
S1	0
S2	0
S3	1

```
module problem8(
          input  logic clk,
          input  logic reset,
          input  logic x,
          output logic y);
```