Exam 1
1. (10 points) Answer TRUE or FALSE.

(a) Gate and behavioral descriptions are both supported by Verilog, and can be mixed within the same module.
   True
(b) A module can be defined within another module.
   False
(c) Simulators initialize all variables to the value \( z \).
   False
(d) An assign...deassign procedural continuous assignment statement can assign value to a net variable.
False
(e) The instance name of a module is required.
True
(f) The first entry in the port list of a user-defined primitive is an input.
False
(g) Verilog has primitives for transistor/switch level design.

True

(h) The comparison 4'bxx10 === 4'bx110 produces a one-bit value 0.

True

(i) The symbol * in a user-defined primitive stands for don't care, i.e., it is an iteration of an input over the values 0, 1, and x.

False
(j) A function can have any declared input, output, or inout port.
False
2. (10 points)

(a) What is the inertial delay of the following and gate?

'timescale 1ns/10ps
and #3.536 (a, b, c);
3.54ns

(b) What is the typical falling delay of G1?

bufif1 #(1:2:3, 4:5:6, 7:8:9) G1 (y, x, enable);
(c) In the following description, if \( x=1 \) and \( y=1 \) at \( t=0 \), then the value of \( \text{buf}_\text{out} \) changes from \( x \) to 1 at \( t= \) __________.

\[
\text{wire} \ #1 \ \text{y\_out}; \\
\text{and} \ #2 \ (\text{y\_out}, \ x, \ y); \\
\text{buf} \ #2 \ (\text{buf\_out}, \ \text{y\_out}); \\
\text{initial begin} \\
\hspace{1cm} x=1; \\
\hspace{1cm} y=1; \\
\text{end}
\]
(d) Declare a memory named mem containing 256 words of 64 bits each. Bit 63 is the most significant bit.

```vhdl
reg [63:0] mem [255:0] ;
```

(e) Declare a 16-bit bus named bus with transport delay of 8 time units. Bit 0 is the most significant bit.

```vhdl
wire [0:15] #8 bus;
```
3. (15 points) Use each of the following styles to complete a Verilog model that describes one-bit comparator whose output is 1 if in_1 is greater than or equal to in_2, where in_1, in_2 are the inputs, and comp_out is the output. Assume that each input has the value 0 or 1.

(a) pre-defined primitive

```verilog
module p3_a (comp_out, in_1, in_2);
    output comp_out;
    input in_1, in_2;
    not (w1, in_2);
    or (comp_out, w1, in_1);
endmodule```

(b) user-defined primitive

primitive p3_b (comp_out, in_1, in_2);
  output comp_out;
  input in_1, in_2;
  table
    1  ?  :  1;
    0  0  :  1;
    0  1  :  0;
  endtable
endprimitive
(c) continuous assignment with conditional operator

module p3_c (comp_out, in_1, in_2);
  output comp_out;
  input in_1, in_2;
  assign comp_out = (in_1 >= in_2) ? 1 : 0 ;
endmodule
(d) cyclic behavior with user-defined function

module p3_d (comp_out, in_1, in_2);
    output comp_out;
    input in_1, in_2;
    reg   comp_out;
    always @ (in_1 or in_2)
        comp_out = comp(in_1, in_2);

    function comp;
        input in1, in2;
        comp=(~in2 | in1);
    endfunction
endmodule
(e) cyclic behavior with user-defined task

module p3_e (comp_out, in_1, in_2);
    output comp_out;
    input in_1, in_2;
    reg comp_out;
    always @ (in_1 or in_2)
        comp(in_1, in_2, comp_out);

    task comp;
        input in_1, in_2;
        output comp_out;
        comp_out = (~in2 | in1);
    endtask
endmodule
4. (10 points) Find the syntax error(s) in each of the following descriptions.

(a) module p4_a (o1, in1, in2);
    input in1, in2;
    output o1;
    o1 = in1 & in2;
endmodule

Possible fix: assign o1 = in1 & in2;
(b) module p4_b (o1, o2, in1, in2, in3, in4);
    input in1, in2, in3, in4;
    output o1, o2;
    nand (o1, in1, in2); #1 (o2, in3, in4);
endmodule

Possible fix: nand #1 (o1, in1, in2), (o2, in3, in4);
(c) module p4_c (a, b, c);
    output a;
    input b, c;
    nand #(3, 4, 5) (a, b, c);
endmodule

Possible fix: delete turn-off delay
(d) module p4_d (y2,y1, x3, x2, x1, x0);
    input x0, x1,x2, x3;
    output y1, y2;
    reg x0, x1;
    wire x2, x3;
    wire y1, y2;
    endmodule

Possible fix: wire x0, x1;
module p4_e (o1, in1, in2, in3, in4);
    input in1, in2, in3, in4;
    output o1;
    and (in1, in2, in3, in4, o1);
endmodule

Possible fix: and (o1, in1, in2, in3, in4);
5. (15 points) Draw the waveforms of sig_1, sig_2, and sig_3 produced by the following description.

```verilog
module wave;
    reg sig_1, sig_2, sig_3;
initial
    begin
        sig_1 = 1'b0;
        #1 sig_3 = 1'b1;
        #1 sig_2 = #1 1'b1;
        sig_1 <= #2 sig_3;
        sig_2 = @ (posedge sig_1) sig_1;
        fork
            #1 sig_3 = 1'b0;
            #2 sig_2 = #1 sig_1;
        join
            #1 sig_1 = sig_3;
        end
    initial
    #10 $finish;
endmodule
```
6. (10 points) Consider a Moore machine whose state transition graph is shown below. Suppose the machine has an asynchronous active-low reset, and the state transitions occur at positive clock edges. Let S0 be the reset state. Use parameters to encode the states, and complete the remaining code below to model the machine.

```verilog
reg [1:0] state, next_state;
parameter s0=2b'00;
parameter s1=2b'01;
parameter s2=2b'10;
always @ (negedge reset or posedge clk)
    if (!reset) state=s0;
    else state = next_state;
always @ (state)
    if ((state == s0) || (state == s1))
        out_bit = 0;
    else if (state == s2)
        out_bit = 1;
```

always @(in_bit or state)
  case (state)
    s0: if (in_bit == 0)
        next_state <= s0;
      else if (in_bit == 1)
        next_state <= s1;
      else next_state <= state;
6.(cont’d)

s1: if (in_bit == 0)
    next_state<=s0;
else if (in_bit == 1)
    next_state<=s2;
else next_state <= state;
s2: if (in_bit == 0)
    next_state <= s0;
else if (in_bit == 1)
    next_state <= s2;
else next_state <= state;
default: next_state <= s0;
(a) Explain why the module below does not correctly model a D flip-flop with asynchronous active-low reset. What change(s) should be made to ensure correct operation?

```
module flip (clk, data, q, qbar, reset);
    input clk, data, reset;
    output q, qbar;
    reg q;
    assign qbar = ~q;
    always @ (posedge clk or reset)
        begin
            if (reset == 0) q = 0; else
                if (clk == 1) q = data;
        end
endmodule
```

Explanation: omitted

Fix: `always@(posedge clk or negedge reset)`
(b) Explain why the module below does not correctly model a four-channel multiplexer. What change(s) should be made to ensure correct operation?

```
module mux4 (a, b, c, d, select, y_out);
  input a, b, c, d;
  input [1:0] select;
  output y_out;
  reg y_out;
  always @ (select)
    case (select)
      0: y_out = a;
      1: y_out = b;
      2: y_out = c;
      3: y_out = d;
      default: y_out = 1'bx;
    endcase
endmodule
```

Explanation: omitted

Fix:

```
0:assign y_out=a;
1:assign y_out=b;
2:assign y_out=c;
3:assign y_out=d;
```
(c) Explain why the module below does not correctly model a 2-bit or gate with propagation delay of 3 time units. What change(s) should be made to ensure correct operation?

module or_gate (y, a, b);
    input a, b;
    output y;
    reg y;
    always @ (a or b)
        begin
            #3 y = a | b;
        end
endmodule

Explanation: omitted

Fix: y <= #3 a|b;
(or replace the always behavior by
or #3(y,a,b);)
(a) (2 points) Write a primitive instance for a 2-input nor gate having the driving strength of Pull 1 and Strong 0. Suppose a and b are the inputs, and y is the output.

\[
\text{nor (pull1, strong0) (y, a, b)};
\]
(b) (3 points) Complete the following switch-level model for a 2-input nor gate having the driving strength of Pull 1 and Strong 0.

```plaintext
module nor_2 (y, a, b);
  output y;
  input a, b;
  supply0 GND;
  pullup (y);
  nmos (y,GND,a);
  nmos (y,GND,b);
endmodule
```
(c) Consider the following circuit.

(i)(6 points) Suppose the data types of Result_1, Result_2, and Result are all wire. Determine the strengths of Result_1, Result_2 and Result, respectively.

- Result_1: Pu1~St1
- Result_2: Pu0~We0
- Result: Pu0~St1

(ii)(2 points) Suppose the data type of Result is changed to wand. Determine the strength of Result.

Pu0~St1
(iii) (2 points) Suppose the data type of Result is changed to word. Determine the strength of Result.

Pu1 ~ St1