COMP311-5: Logic Circuit Design

Spring 2019, Prof. Taigon Song

Quiz #2: May 21, 9:00 – 9:50pm [Total: 128 points]

Guidelines:

1. Read the questions carefully and pay attention to the special instructions.
2. Show all your mark to receive full credit.
3. State any assumptions you make on your solution.
4. Total number of pages in this exam: 5 (including this cover page.)
5. Write your name at ALL pages.
6. For each page with unwritten name or student ID: -5 points

#ID / Name:____

Prob. 1 (20pts)	
Prob. 2 (20pts)	
Prob. 3 (20pts)	
Prob. 4 (28pts)	
Prob. 5 (20pts)	
Prob. 6 (20pts)	
Name penalty (-25pts)	
Total (128pts)	

Student ID#: Name:

1. Design a user-defined primitive AND gate. You may not consider any 'x' or 'z' related conditions in your UDP truth table. (20 points)

```
_______________________ udp_and (out, a, b); // fill in the blank
```

2. Find the correct answer (20 points)

Vertical partitioning of a design is partitioning my huge module based on bit slices	O	X
UDPs (user-defined primitives) can take vector input terminals	O	X
UDPs do not handle 'x' values	O	X
UDPs can have multiple output terminals	O	X
UDPs can be defined inside other modules	O	X

3. Fix the source code to override the "out" value to be 1'b1 during 30 – 40ns. (20 points)

```
module xnorgate(a1, a2, z);
  input a1, a2;
  output z;

  assign z=~(a1^a2);

endmodule
```

```
`timescale 1ps / 1ps
module xnorgate_tb();
  wire out;
  //integer clock;
  reg clock1 ,clock2;

  xnorgate u1(clock1, clock2, out);

  always begin
    #1 clock1 =~clock1;
  end

  always begin
    #2 clock2 =~clock2;
  end

  initial begin
    clock1 = 0;
    clock2 = 0;
  end

endmodule
```

4. Below is a Verilog HDL source of designing a traffic light controller. For the part described below, please fix the source code to avoid any possible compilation error that can possibly happen by removing, fixing, or including any necessary statements. The conditions for designing this traffic light controller is also described below. (Total: 28 points, removing/fixing/including each line: 2 points)

Conditions:

- The traffic signal for the main highway gets highest priority because cars are continuously present on the main highway. Thus, the main highway signal remains green by default.
- Occasionally, cars from the country road arrive at the traffic signal. The traffic signal for the country road must turn green only long enough to let the cars on the country road go.
- As soon as there are no cars on the country road, the country road traffic signal turns yellow and then red and the traffic signal on the main highway turns green again.
- There is a sensor to detect cars waiting on the country road. The sensor sends a signal X as input to the controller. X = 1 if there are cars on the country road; otherwise, X= 0.
- There are delays on transitions from S1 to S2, from S2 to S3, and from S4 to S0. The delays must be controllable.

```
define Y2RDELAY 3

define R2GDELAY 2

module traffic(hwy, cntry, x, clk, clear,
state);

  output reg  hwy, cntry;

  output reg  state;

  input [1:0] x, clk, clear;

  parameter RED = 2'd0,

            YELLOW = 2'd1,

  parameter S0 = 3'd0,

            S1 = 3'd1,

            S2 = 3'd2,

            S3 = 3'd3,

            S4 = 3'd4;

  reg [4:0] next_state;

  always@(posedge clk) begin

    if(clear)

          state <= 0;

    end

  always @(state) begin

    hwy = GREEN;

        case (state)

          S0: ;

          S1: hwy = YELLOW;

          S2: hwy = RED;

          S3: begin

                hwy = RED;

                cntry = GREEN;

              end
```

```
          S4: begin

                hwy = RED;

                cntry = YELLOW;

              end

    endcase

  end

  always @(state or x) begin

    case (state)

        S0: if (x)

            next_state = S1;

           else

            next_state = S0;

          S1: begin

           repeat(Y2RDELAY) @(posedge clk) ;

            next_state = S2;

             end

          S3: if (x)

            next_state = S3;

             else

            next_state = S4;

          S4: begin

           repeat(Y2RDELAY) @(posedge clk);

           next_state = S0;

              end

          default: next_state = S0;

        endcase

  end

endmodule
```

5. Describe when (1) s1 escapes ‘x’, (2) c1 escapes ‘x’, (3) sum escapes ‘x’ (4) c_out escapes ‘x’
(Total: 20 points)

```
`timescale 1ns / 1ns
module prob5(sum, c_out, a, b, c_in, s1,
s2, c1);
  output sum, c_out;
  input a, b, c_in;

  output s1, s2, c1;

  xor #5 (s1, a, b);
  and    (c1, a, b);
  xor #9 (sum, s1, c_in);
  and #8 (s2, s1, c_in);
  xor #2 (c_out, s2, c1);

  specify
    (b    => sum  ) = 1;
       (c_in => sum  ) = 20;
  endspecify

endmodule
```

```
module fa_tb();
  reg a, b;
  reg cin;
  wire sum;
  wire cout;
  wire s1, s2, c1;

  prob5 u0(sum, cout, a, b, cin, s1, s2,
c1);

  initial begin
    a = 0; b=0; cin=0;
       #50 cin=1;
       #50       b=0; cin=1;
       #50 a = 0; cin=0;

  end

endmodule
```

6. Regarding synthesis of Verilog HDL, some loop statements (e.g., while, for) are synthesizable, and some loop statements are not. Even for the same “while” and “for” loops, it could either be synthesizable or not. Place your reasoning (1) why this difference happens, and (2) describe what is your guideline for a “synthesizable” loop statement generation. (Total: 20 points)