Solution

*Name: EasyPizzy

*Registration: ______________________

Department of Computer Systems Engineering
University of Engineering & Technology Peshawar

Digital System Design
CSE 308

Midterm Examination Spring 2023
13 April 2023, Duration: 120 Minutes

****Exam Rules****
Please read carefully before proceeding.
1- This exam is closed books/notes/Internet.
2- Answer all problems on the answer sheet.
3- Problems will not be interpreted during the exam.

Good Luck!

Problem 1. (25 pts., CLO-1)
Below is an RTL (or Dataflow) description for a circuit.

```
module RTL_circuit (x, y, a, b, c);

     input a, b, c;
     output x, y;
     wire a, b, c, x, y;
     wire na, nb, nc, t3, t4, t5;

     assign  na  =  !a;
     assign  nb  =  !b;
     assign nc = !c;
     assign t3 = na && b && c;
     assign t4 = a && nb && c;
     assign t5 = a && b && nc;
     assign x = t3 || t5;
     assign y = a || t4;

endmodule
```

1(a) (3 pts.) Give a Verilog statement that instantiates the above RTL_circuit, with the instance name MID. When you instantiate the circuit, use the same names for wires as is used in the module port list.

Solution:

```
RTL_circuit MID (x, y, a, b, c);
```

1(b) (6 pts.) Rewrite the RTL_circuit using Verilog built-in primitives and structural Verilog. Part of the module is done for you.

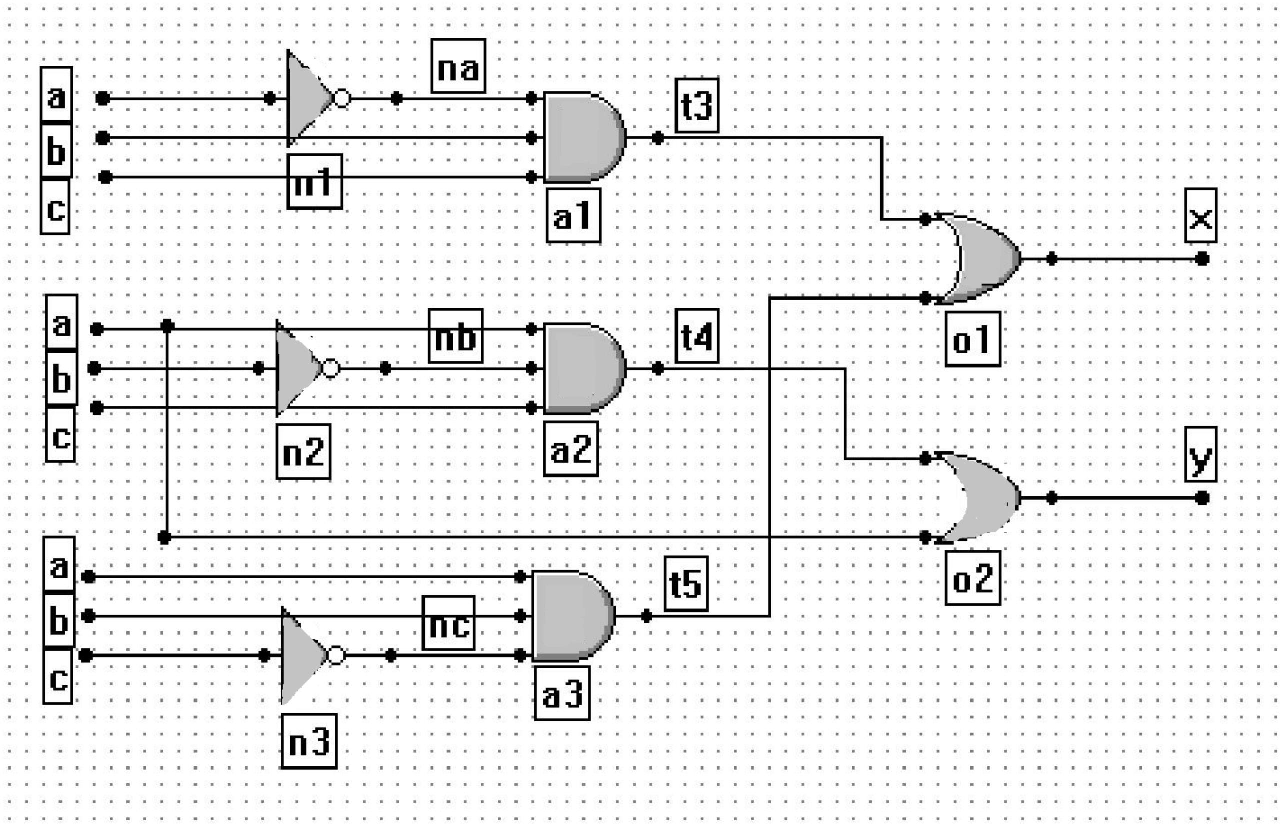
```
module struct_circuit (x, y, a, b, c);

     input a, b, c;
     output x, y;
     //Write your code here



endmodule
```

Solution:

```
module struct_circuit (x, y, a, b, c);

      input a, b, c;
      output x, y;
      //Write your code here
      wire a, b, c, x, y;
      wire na, nb, nc, t3, t4, t5;
      not n1(na, a);
      not n2(nb, b);
      not n3(nc, c);
      and a1(t3, na, b, c);
      and a2(t4, a, nb, c);
      and a3(t5, a, b, nc);
      or o1(x, t3, t5);
      or o2(y, a, t4);

endmodule
```

1(c) (3 pts.) Draw a gate-level diagram for your module in **1(b)**. Label all nets on the diagram.

Solution:



1(d) (6 pts.) Rewrite the RTL_circuit using behavioral Verilog. Part of the module is done for you.

```
module behav_circuit (x, y, a, b, c);

      input a, b, c;
      output x, y;
      //Write your code here




endmodule
```

Solution:

```
module behav_circuit (x, y, a, b, c);

      input a, b, c;
      output x, y;
      //Write your code here
      wire a, b, c;
      reg x, y;
      reg t3, t4, t5;
      always @( a or b or c ) t3 <= !a & b & c;
      always @( a or b or c ) t4 <= a & !b & c;
      always @( a or b or c ) t5 <= a & b & !c;
      always @( t3 or t5 ) x <= t3 | t5;
      always @( a or t4 ) y <= a | t4;

endmodule
```

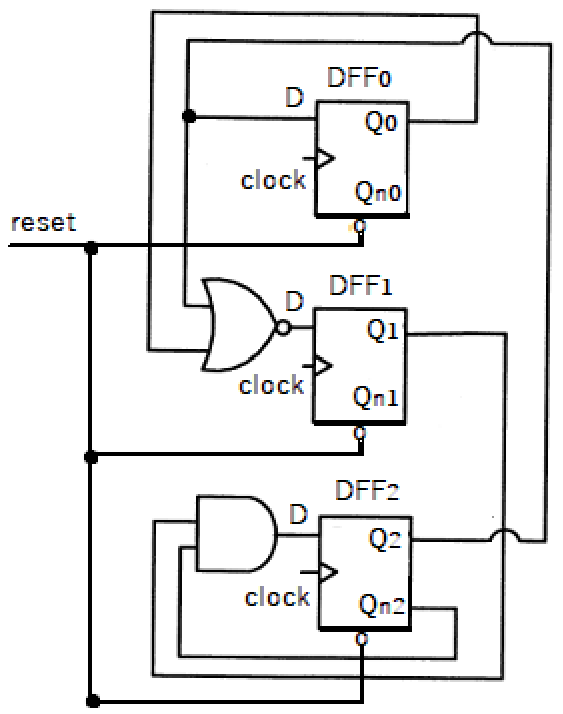
1(e) (7 pts.) Write the output of RTL_circuit for the following test bench.

```
//test bench for RTL_circuit
module test_RTL_circuit;

      reg a, b, c;
      wire x, y;

      RTL_circuit RTLC (x, y, a, b, c);

      Initial begin
            a = 1; b = 1; c = 1;
            #30 a = 0; b = 1; c = 1;
      end

      initial begin
            $display ($time, " x = %b and y = %b", x, y);
            #6 $display ($time, " x = %b and y = %b", x, y);
            #5 $display ($time, " x = %b and y = %b", x, y);
            #20 $display ($time, " x = %b and y = %b", x, y);
            #5 $display ($time, " x = %b and y = %b", x, y);
            #6 $display ($time, " x = %b and y = %b", x, y);
            #8 $display ($time, " x = %b and y = %b", x, y);
      end

endmodule
```

Solution:

```
0 x = 0 and y = 1
6 x = 0 and y = 1
11 x = 0 and y = 1
31 x = 1 and y = 0
36 x = 1 and y = 0
42 x = 1 and y = 0
50 x = 1 and y = 0
```

Problem 2. (10 pts., CLO-2)
In this problem, design a 3-bit counter (the circuit diagram and state table are shown in **Figure 1**) using as building block the D flip-flop given in **Figure 2.**



Input		Output
reset	clock	$Q_2Q_1Q_0$
0	x	000 **(0)**
1	↑	010 **(2)**
1	↑	110 **(6)**
1	↑	001 **(1)**
1	↑	000 **(0)**

Figure 1. The circuit diagram (left) and state table (right) of the 3-bit counter

```
module DFF (D, clock, reset, Q, Qn);

      input D; // Data input
      input clock; // Clock input
      input reset; // Asynchronous active-low reset
      output Q, Qn; // Outputs Q and Q'
      reg Q;

      always @ (posedge clock or negedge reset)
                 if (reset==1'b0)
                       Q <= 0;
                 else
                       Q <= D;

      assign Qn = ~Q;
endmodule
```

Figure 2. Verilog code for rising-edge D flip-flop with asynchronous active-low reset

The suggested skeleton file for the counter has been written below. The module has 2 inputs – **clock** and **reset** which is active-low. The output is **out** which is 3-bit in size.

```
module counter (clock, reset, out);

      input clock, reset;
      output [2:0] out;
      // Write your code here




endmodule
```

Solution:

```
module counter (clock, reset, out);

      input clock, reset; output [2:0] out;
      // Write your code here
      wire [2:0] Qn;

      DFF dff0 (out[2], clock, reset, out[0], Qn[0]);
      DFF dff1 (~(out[0]|out[2]), clock, reset, out[1], Qn[1]);
      DFF dff2 (out[1]& Qn[2], clock, reset, out[2], Qn[2]);

endmodule
```

Problem 3. (10 pts., CLO-1)
In this problem, write Verilog code for an 8-to-1 multiplexer. In this case, the value on the 3-bit select line will route 1 of 8 inputs to the output. This module is purely combinatorial.

The following are the ports of the module:

Sel	3-bit select line
I0, I1, I2, I3, I4, I5, I6, and I7	1-bit data inputs
OUT	1-bit output

Solution:

```
module m81 (OUT, I0, I1, I2, I3, I4, I5, I6, I7, Sel);

     input I0, I1, I2, I3, I4, I5, I6, I7;
     input [2:0] Sel;
     output OUT;
     reg OUT;

     always @(*)
          case (Sel)
               0: OUT=I0;
               1: OUT=I1;
               2: OUT=I2;
               3: OUT=I3;
               4: OUT=I4;
               5: OUT=I5;
               6: OUT=I6;
               7: OUT=I7;
          endcase

endmodule
```

Problem 4. (10 pts., CLO-1/CLO-2)
For this design, combine the counter (from **Problem 2**) with the multiplexer (from **Problem 3**) to create a circuit such that the output of the counter controls the select lines of the multiplexer.

This top-level design has the following port definitions:

clk	1-bit clock
reset	1-bit reset line
D0, D1, D2, D3, D4, D5, D6, and D7	1-bit data inputs
OUT	1-bit output

Solution:

```
module top (OUT, clk, reset, D0, D1, D2, D3, D4, D5, D6, D7);

     input clk, reset, D0, D1, D2, D3, D4, D5, D6, D7;
     output OUT;

     wire [2:0] out;

     counter c (clk, reset, out);
     m81 m (OUT, D0, D1, D2, D3, D4, D5, D6, D7, out);

endmodule
```