*Name: ______________________

*Registration: ______________________

Department of Computer Systems Engineering
University of Engineering & Technology Peshawar

Digital System Design

CSE 308

Midterm Examination Spring 2017

27 March 2017 Duration: 120 Minutes

Exam Rules

Please read carefully before proceeding.

1- This exam is OPEN books/notes, but CLOSED computers/cell phones.
2- No calculators of any kind are allowed.
3- Sharing of books, notes, and other materials during this exam is not permitted.
4- There are 4 problems in total. Some problems are harder than others. Answer the easy ones first to maximize your score.
5- Answer all problems on the problem sheet.
6- Questions will not be interpreted during the exam.
7- This exam booklet contains 11 pages, including this cover. Count them to be sure you have them all.

Problem 1	________	(25 pts)
Problem 2	________	(30 pts)
Problem 3	________	(15 pts)
Problem 4	________	(30 pts)
Exam Total	________	(100 pts)

Good Luck!

Problem 1:

(a) The module below is in **explicit structural** form. Re-write the module in **behavioral** form. **(5 pts)**

```
module expl_str(x, y, a, b, c);
     input a, b, c;
     output x, y;
     wire a, b, c, x, y;
     wire na, nb, nc, t3, t5, t6;

     not n1(na, a);
     not n2(nb, b);
     not n3(nc, c);
     and a1(t3, na, b, c);
     and a2(t5, a, nb, c);
     and a3(t6, a, b, nc);
     or o1(x, t3, t6);
     or o2(y, a, t5);
endmodule

//SOLUTION
```

(b) Convert the following **behavioral** code to **explicit structural** code. (**5 pts**)

```
module btos(x, a, b);
     input a, b;
     output x;
     wire a, b;
     reg x;

     always @( a or b ) if( a ) x = b; else x = ~b;
endmodule

//SOLUTION
```

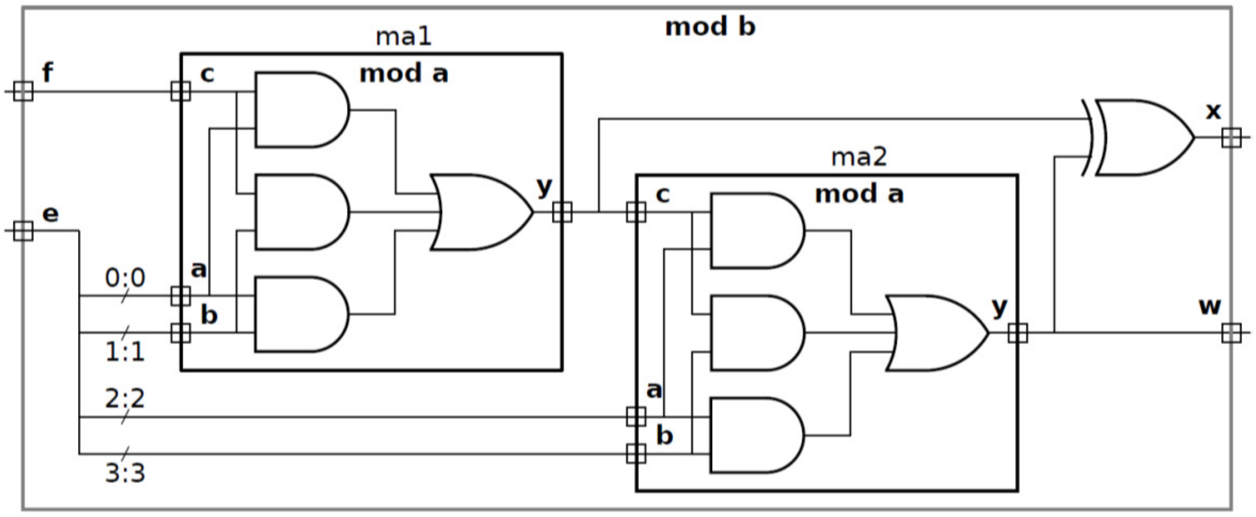
(c) The module below is in **explicit structural** form, in which only primitive gates (and module instantiations) are used. Will the synthesis program synthesize exactly that arrangement and types of gates? Explain. (**5 pts**)

```
module bfa_structural( output sum, cout, input a, b, cin );
     wire term001, term010, term100, term111;
     wire ab, bc, ac;
     wire na, nb, nc;

     not n1( na, a);
     not n2( nb, b);
     not n3( nc, cin);
     and a1( term001, na, nb, cin);
     and a2( term010, na, b, nc);
     and a3( term100, a, nb, nc);
     and a4( term111, a, b, cin);
     or o1( sum, term001, term010, term100, term111);
     and a10( ab, a, b);
     and a11( bc, b, cin);
     and a12( ac, a, cin);
     or o2( cout, ab, bc, ac);
endmodule

//SOLUTION
```

(d) Write a Verilog description of the hardware illustrated below. Base the names of ports, wires, and instances on labels in the illustration. The description can be **behavioral** or **structural**, but it must be synthesizable. **(10 pts)**



```
//SOLUTION
```

Problem 2:

Draw a schematic of the hardware the synthesis system will synthesize for the following Verilog code examples. If flip flops are used, indicate if they are level triggered or edge triggered. Otherwise, don't worry about using the precisely correct gate or symbol, as long as it's functionally correct.

(a) Show an approximate schematic for the module below. In what synthesizable form is the Verilog description? **(15 pts)**
Hint: think about what form the code is in.

```
module mod_a(x, y, a, b, c);
     input a, b, c;
     output x, y;
     wire [7:0] b, c;
     reg [8:0] x, y;

     always @( a or b or c ) begin
          if( a ) begin
               x = b + c;
               y = b - c;
          end else begin
               x = b - c;
          end
     end
endmodule

//SOLUTION
```

(b) Show an approximate schematic for the module below. What form is the description in? **(15 pts)**
Hint: think about what form the code is in.

```
module mod_b(x, y, d, e, f);
     input d, e, f;
     output x, y;
     reg x, y;

     always @( negedge e  )
          if( d ) begin
               x = 0;
               y = 1;
          end else if ( f ) begin
               x = 1;
               y = 0;
          end
endmodule

//SOLUTION
```

Problem 3:

(a) Show the values of the variables as indicated below: **(10 pts)**

```
module tryout();
     reg [15:0] a;
     reg [0:15] b;
     reg [3:0] e [3:0];
     reg [3:0] x1, x2, x3;
     reg x4;

     initial begin
         a = 16'h1234;
         b = 16'h1234;
         x1 = a[3:0];          //SOLUTION: value of x1 is
         x2 = b[0:3];          //SOLUTION: value of x2 is
         x3 = a[0:3] & b[0:3]; //SOLUTION: value of x3 is
         x3 = a[4:7] | b[3:0]; //SOLUTION: value of x3 is
         x3 = a[4:7] ^ b[7:4]; //SOLUTION: value of x3 is
         x4 = & a[8:11];       //SOLUTION: value of x4 is
         x4 = & a[11:8];       //SOLUTION: value of x4 is
         x4 = | b[12:15];      //SOLUTION: value of x4 is
         e = 16'h1234;
         e[0] = e[0] + 4'hf;   //SOLUTION: value of e is
         e = 16'h1234;
         e[0][0] = e[0][0] + 1'b1; //SOLUTION: value of e is
     end
endmodule
```

(b) Do the two code fragments below do the same thing? If not, how do they differ? **(5 pts)**

```
// Fragment A.
if ( foo > bar ) x = x + 1; else y = y + 1;

// Fragment B.
case ( foo > bar )
     1: x = x + 1;
     default: y = y + 1;
endcase

//SOLUTION
```

Problem 4:

(a) Write the hardware description of an 8-bit register with shift-left and shift-right modes of operation. The suggested skeleton file has been written below: **(10 pts)**

```
module slsr(sl, sr, din, clk, reset, Q);
     input sl, sr, din, clk, reset;
     output [7:0] Q;

     //WRITE YOUR CODE HERE
```

```
endmodule
```

(b) Write the hardware description of a 4-bit mod-13 counter. The suggested skeleton file has been written below: **(10 pts)**

```
module mod13Cntr(clk, reset, Q);
     input clk, reset;
     output [3:0] Q;

     //WRITE YOUR CODE HERE
```

```
endmodule
```

(c) Write the hardware description of a 4-bit adder/subtractor. An adder/subtractor is a piece of hardware that can give the result of addition or subtraction of the two numbers based on a control signal. Assume that the numbers are in 2's complement notation. Please keep in mind that this is a combinatorial circuit. The suggested skeleton file to start with is given below: **(10 pts)**

```
module addsub (a, b, sel, res);
     input [3:0] a, b;
     input sel;
     output [3:0] res;

     //WRITE YOUR CODE HERE









endmodule
```

⟨This page is intentionally left blank. This page can be used for scratch work or as extra space. If you write work here that you want me to grade, be sure to clearly indicate which problem(s) the work corresponds to!⟩