*Name: Halwa

*Registration: EasyPizzy-007

Department of Computer Systems Engineering
University of Engineering & Technology Peshawar

Digital System Design
CSE 308

Midterm Examination Spring 2022

11 May 2022, Duration: 120 Minutes

Exam Rules

Please read carefully before proceeding.

- This exam is OPEN books/notes but CLOSED Internet/laptops/phones.
- Life is better when shared with others. During this exam, however, sharing of books, notes, or other resources is not permitted.
- No calculators/phones of any kind are allowed.
- Attempt all problems on the problem sheet. Use the answer sheet for scratch space and write a neat copy of your final answer in the provided space on the problem sheet. **Very Important!**
- Be precise and concise in your answers (no extra explanatory text).
- Some problems are harder than others. Answer the easy ones first to maximize your score.
- Problems will not be interpreted during exam. **Please note!**
- This exam booklet contains 5 pages, excluding this cover page. Count them to be sure you have them all.

Problem 1		(20 pts.)
Problem 2		(20 pts.)
Problem 3		(30 pts.)
Total		**(70 pts.)**

Good luck!

Problem 1:..(20 pts.)

1(a) (10 pts.) Write a Verilog **behavioral description** (using always) of a four-bit adder module. The adder should have three inputs, a, b, and cin, and two outputs, sum and cout. Ports cin and cout are one bit, the other ports are four bits each.

```
module add4 (sum, cout, a, b, cin);

    output [3:0] sum;
    output cout;
    input [3:0] a, b;
    input cin;
    //Write your code here
    reg [3:0] sum;
    reg cout;

    always @( a or b or cin )
        {cout, sum} = a + b + cin;

endmodule
```

1(b) (10 pts.) Write a Verilog **structural description** (using hierarchical design methodology) of an eight-bit adder that uses two of the four-bit adders above. (That is, instantiate the module designed above.)

```
module add8 (sum, cout, a, b, cin);

    output [7:0] sum;
    output cout;
    input [7:0] a, b;
    input cin;
    //Write your code here
    wire cbetween;

    add4 a1 (.sum(sum[3:0]), .cout(cbetween), .a(a[3:0]), .b(b[3:0]),
    .cin(cin));
    add4 a2 (.sum(sum[7:4]), .cout(cout), .a(a[7:4]), .b(b[7:4]),
    .cin(cbetween));

endmodule
```

Problem 2:......................................(20 pts.)

Consider a two-bit comparator that has two inputs A and B and three outputs where the first output is asserted if A==B, second output is asserted if A>B and finally the last output is asserted if A<B.

2(a) (10 pts.) Write a Verilog **dataflow description** (using assign) of a two-bit comparator module. The comparator should have two inputs, A and B, and three outputs, AEB (indicating A==B), AGB (indicating A>B) and ALB (indicating A<B). Ports AEB, AGB and ALB are one bit, the other ports are two bits each.

```
module comp2 (AEB, AGB, ALB, A, B);

    output AEB, AGB, ALB;
    input [1:0] A, B;
    //Write your code here

    assign AEB = A==B, AGB = A>B, ALB = A<B;

endmodule
```

2(b) (10 pts.) Synthesize a combinational circuit that can compare two four-bit numbers A $(=A_3A_2A_1A_0)$ and B $(=B_3B_2B_1B_0)$ using **hierarchical design approach**, employing the above two-bit comparator as a building block. The hierarchical design of the four-bit comparator is provided in Figure 1.

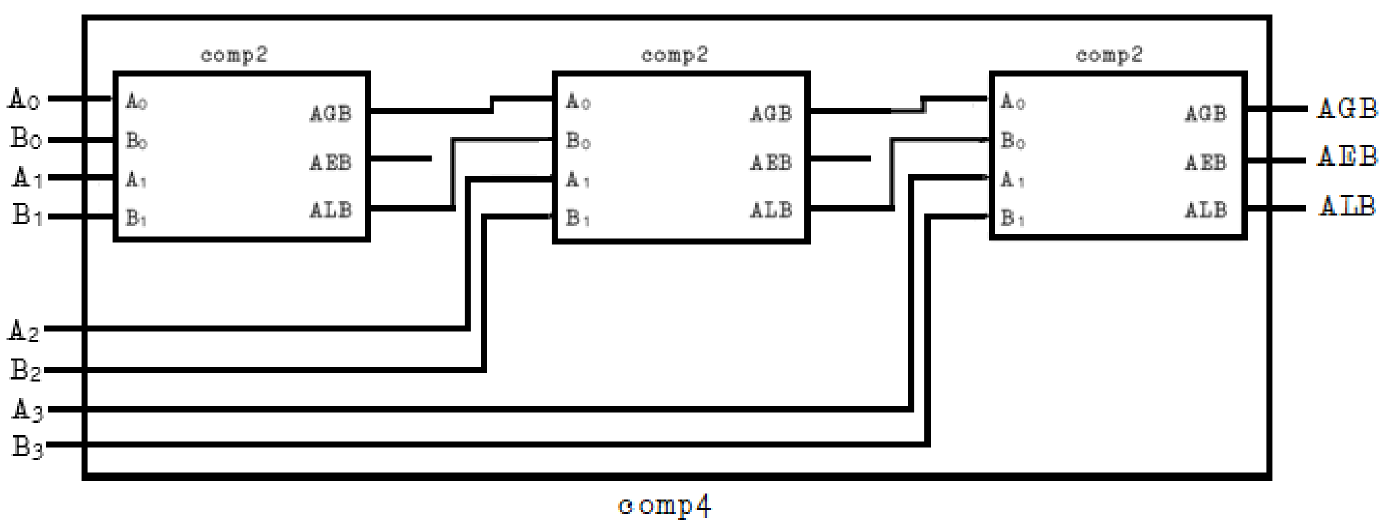

Figure 1. Hierarchical design of four-bit comparator

```
module comp4 (AEB, AGB, ALB, A, B);

    output AEB, AGB, ALB;
    input [3:0] A, B;
    //Write your code here
    wire AEB1, AGB1, ALB1, AEB2, AGB2, ALB2;

    comp2 c1 (.AEB(AEB1), .AGB(AGB1), .ALB(ALB1), .A(A[1:0]),
    .B(B[1:0]));
    comp2 c2 (.AEB(AEB2), .AGB(AGB2), .ALB(ALB2), .A({A[2], AGB1}),
    .B({B[2], ALB1}));
    comp2 c3 (.AEB(AEB), .AGB(AGB), .ALB(ALB), .A({A[3], AGB2}),
    .B({B[3], AGB2}));

endmodule
```

Problem 3:..**(30 pts.)**

3(a) **(10 pts.)** Figure 2 shows a 7-segment decoder that has a four-bit input ABCD (A is msb and D is lsb). This decoder produces seven outputs that are used to display a character on a 7-segment display. Table 1 lists the characters that should be displayed for each value of ABCD.

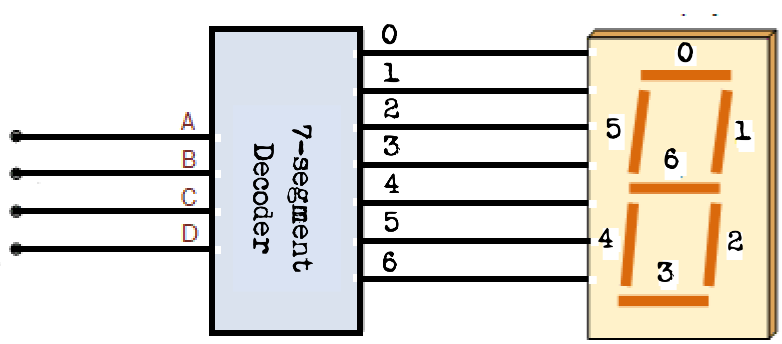

Figure 2. A 7-segment Display and Decoder

The seven segments in the display are identified by the indices 0 to 6 as shown in the figure. Each segment is illuminated by driving it to the logic value 0. Write a

Verilog module for the 7-segment decoder. Use any Verilog structure that makes sense to you.

ABCD	Character
0000	H
0001	E
0010	L
0011	L
0100	O
0101	H
0110	E
0111	L
1000	P
1001	P
1010	L
1011	E
1100	A
1101	S
1110	E
1111	-

Table 1. Character Code Table

```
module segment7 (seg, A, B, C, D);
      output [0:6] seg;
      input A, B, C, D;
      //Write your code here
      reg [0:6] seg;

      /*always block for converting ABCD digit into 7 segment
      format*/
      always @( A, B, C, D )
      begin
            case ({A, B, C, D}) //case statement
                  4'd0: seg = 7'b1001000; //H
                  4'd1: seg = 7'b0110000; //E
                  4'd2: seg = 7'b1110001; //L
                  4'd3: seg = 7'b1110001; //L
                  4'd4: seg = 7'b0000001; //O
                  4'd5: seg = 7'b1001000; //H
                  4'd6: seg = 7'b0110000; //E
                  4'd7: seg = 7'b1110001; //L
                  4'd8: seg = 7'b0011000; //P
                  4'd9: seg = 7'b0011000; //P
                  4'd10: seg = 7'b1110001; //L
                  4'd11: seg = 7'b0110000; //E
                  4'd12: seg = 7'b0001000; //A
```

```
                4'd13: seg = 7'b0100100; //S
                4'd14: seg = 7'b0110000; //E
                4'd15: seg = 7'b1111110; //-

                /*switch off 7 segment character when input is not
            a decimal number from 0 to 15.*/
                 default: seg = 7'b1111111;
            endcase
    end

endmodule
```

3(b) (10 pts.) Write a Verilog module for a 4-bit negative-edge triggered up counter with asynchronous clear.

```
module counter (count, clk, clr);
      output [3:0] count;
      input clk, clr;
      //Write your code here
      reg [3:0] count;

      always @( negedge clk or posedge clr )
      begin
        if (clr)
            count = 4'b0000;
        else
            count = count + 1'b1;
      end

endmodule
```

3(c) (10 pts.) In this problem, write a top-level module that combines the 7-segment decoder (from 3(a)) with the counter (from 3(b)) to create a circuit such that the output of the counter controls the input lines of the decoder.

The suggested skeleton file has been written below.

```
module Top (out, clk, rst);

    output [0:6] out;
    input clk, rst;
    //Write your code here
    wire [3:0] tmp;

    counter c (.count(tmp), .clk(clk), .clr(rst));
    segment7 s7 (.seg(out), .A(temp[3]), .B(temp[2]), .C(temp[1]),
    .D(temp[0]));

endmodule
```