# Assignment 2

By: Abdelrahman Maher Hassan

# Question 1

Codes:

Four_Bit_Encoder_first_implementation.v | Four_Bit_Encoder_Second_implementation.v | Four_Bit_Encoder_TB.v

```
module Four_Bit_Encoder_First_implementation(X,Y);
    input      [3:0]X;
    output reg [1:0]Y;

    always @(X) begin
        casex (X)
            4'b1xxx: Y = 2'b11;
            4'b01xx: Y = 2'b10;
            4'b001x: Y = 2'b01;
            4'b000x: Y = 2'b00;
            default: Y = 2'b00;
        endcase
    end
endmodule
```

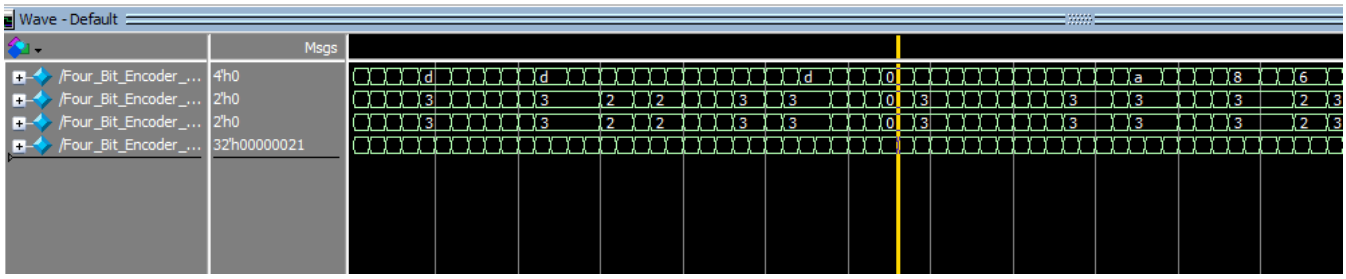
Four_Bit_Encoder_first_implementation.v | Four_Bit_Encoder_Second_implementation.v | Four_Bit_Encoder_TB.v

```
module Four_Bit_Encoder_Second_implementation (In,Out);
    input      [3:0] In;
    output reg [1:0] Out;

    always@(In) begin
        if(In[3] == 1)
            Out = 2'b11;
        else if(In[2] == 1)
            Out = 2'b10;
        else if(In[1] == 1)
            Out = 2'b01;
        else
            Out = 2'b00;
    end
endmodule
```

Test Bench:

```
Four_Bit_Encoder_first_implementation.v   Four_Bit_Encoder_Second_implementation.v   Four_Bit_Encoder_TB.v

module Four_Bit_Encoder_TB();

    reg  [3:0]X0;
    wire [1:0]Y1,Y2;

    Four_Bit_Encoder_First_implementation Encoder_DUT (X0,Y1);

    Four_Bit_Encoder_Second_implementation Encoder_Golden (X0,Y2);

    integer i;

    initial begin
        for(i=0;i<99;i=i+1) begin
            X0 = $random;
            #10;

            if(Y1 != Y2) begin
            $display ("Error-The 2 outputs of the 2 Encoder modules is not the same.");
            $stop;
            end
        end
        $stop;
    end
endmodule

```

Simulation:



# Question 2

Codes:

File Edit Selection Find View Goto Tools Project Preferences Help

Decimal_to_BCD.v | Decimal_to_BCD_TB.v

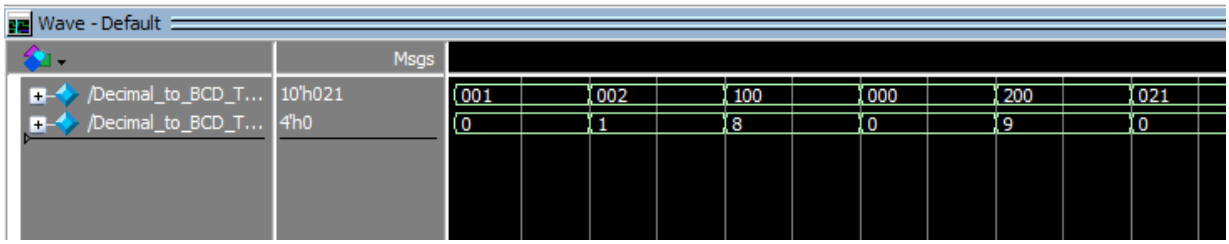
```
module Decimal_to_BCD (D,Y);
    input  [9:0]D;
    output reg [3:0]Y;

    always @(D) begin
        case (D)
        10'b0000000001: Y = 0;
        10'b0000000010: Y = 1;
        10'b0000000100: Y = 2;
        10'b0000001000: Y = 3;
        10'b0000010000: Y = 4;
        10'b0000100000: Y = 5;
        10'b0001000000: Y = 6;
        10'b0010000000: Y = 7;
        10'b0100000000: Y = 8;
        10'b1000000000: Y = 9;
        default: Y = 4'b0000;
        endcase
    end
endmodule
```

Test Bench:

Decimal_to_BCD.v | Decimal_to_BCD_TB.v

```
module Decimal_to_BCD_TB();

    reg  [9:0]In;
    wire [3:0]Out;

    Decimal_to_BCD converter1 (In,Out);

initial begin
    In = 10'b0000000001; //0
    #10;
    In = 10'b0000000010; //1
    #10;
    In = 10'b0100000000; //8
    #10;
    In = 10'b0000000000; //0
    #10;
    In = 10'b1000000000; //9
    #10;
    In = 10'b0000100001; //0
    $stop;
end

initial begin
    $monitor("In= %b, Out=%b",In,Out);
end

endmodule
```

Simulation:


Wave - Default
Msgs
/Decimal_to_BCD_T...
/Decimal_to_BCD_T...
10'h021
4'h0
001
002
100
000
200
021
0
1
8
0
9
0

# Question 3

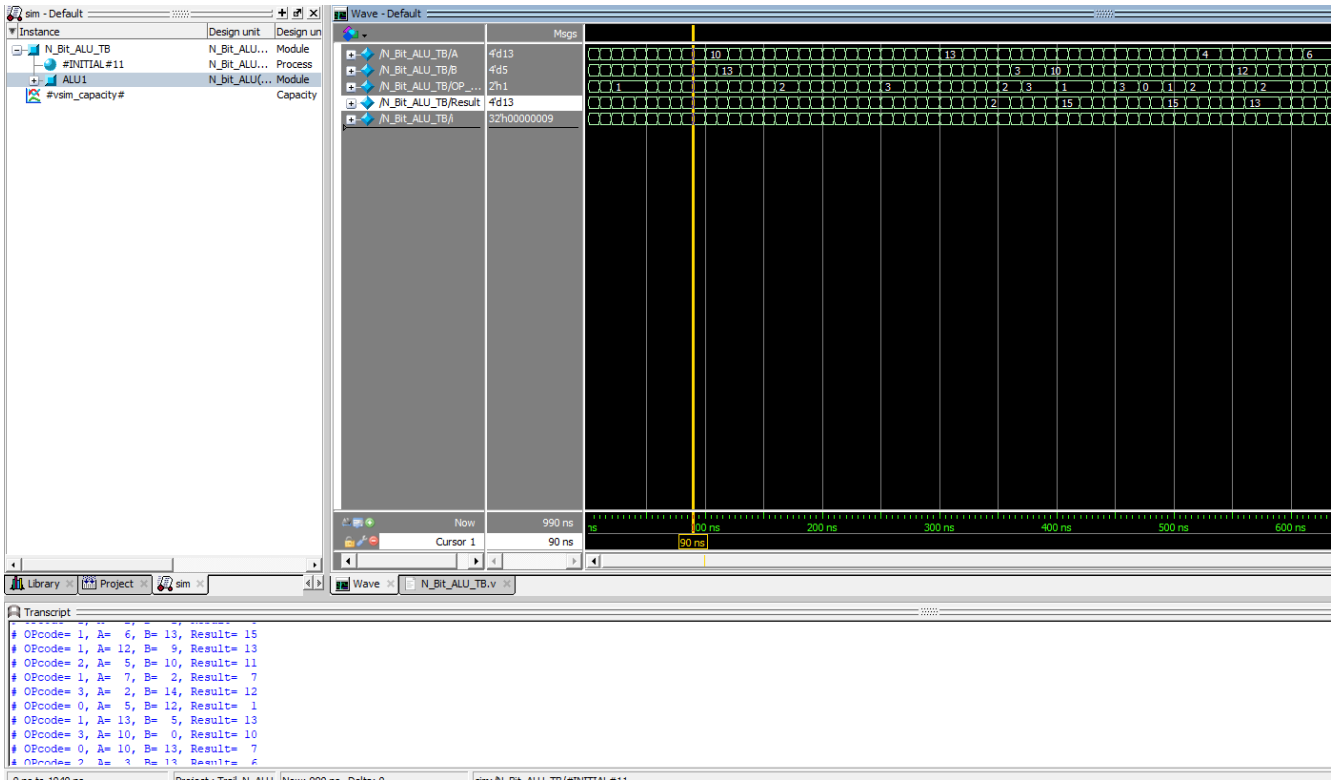
Codes:

```
N_bit_ALU.v        N_Bit_ALU_TB.v

module N_bit_ALU (A, B, OP_Code, Result);

    parameter NO_of_bits = 4;

    input  [NO_of_bits-1:0] A, B;
    input  [1:0] OP_Code;
    output [NO_of_bits-1:0] Result;

    assign Result = (OP_Code == 0) ? (A + B) : (OP_Code == 1) ? (A | B) : (OP_Code == 2) ? (A - B) :  (A ^ B);

endmodule
```

Test Bench:

```
N_bit_ALU.v        N_Bit_ALU_TB.v

module N_Bit_ALU_TB();

    reg  [3:0] A, B;
    reg  [1:0] OP_Code;
    wire [3:0] Result;

    N_bit_ALU #(4) ALU1 (A, B, OP_Code, Result);

    integer i;

    initial begin
        for(i=0;i<99;i=i+1) begin
            OP_Code = $random;
            A = $random;
            B = $random;
            #10;
        end
        $stop;
    end

    initial begin
        $monitor ("OPcode= %d, A= %d, B= %d, Result= %d",OP_Code, A, B, Result);
    end

endmodule
```

Simulation:


sim - Default
Instance
Design unit
Design un
N_Bit_ALU_TB
N_Bit_ALU...
Module
#INITIAL#11
N_Bit_ALU...
Process
ALU1
N_Bit_ALU(...
Module
#vsim_capacity#
Capacity
Library
Project
sim
Wave - Default
Msgs
/N_Bit_ALU_TB/A
4'd13
/N_Bit_ALU_TB/B
4'd5
/N_Bit_ALU_TB/OP_...
2'h1
/N_Bit_ALU_TB/Result
4'd13
/N_Bit_ALU_TB/i
32'h00000009
Now
990 ns
Cursor 1
90 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
Wave
N_Bit_ALU_TB.v
Transcript
# OPcode= 1, A= 6, B= 13, Result= 15
# OPcode= 1, A= 12, B= 9, Result= 13
# OPcode= 2, A= 5, B= 10, Result= 11
# OPcode= 1, A= 7, B= 2, Result= 7
# OPcode= 3, A= 2, B= 14, Result= 12
# OPcode= 0, A= 5, B= 12, Result= 1
# OPcode= 1, A= 13, B= 5, Result= 13
# OPcode= 3, A= 10, B= 0, Result= 10
# OPcode= 0, A= 10, B= 13, Result= 7

# Question 4

Code:

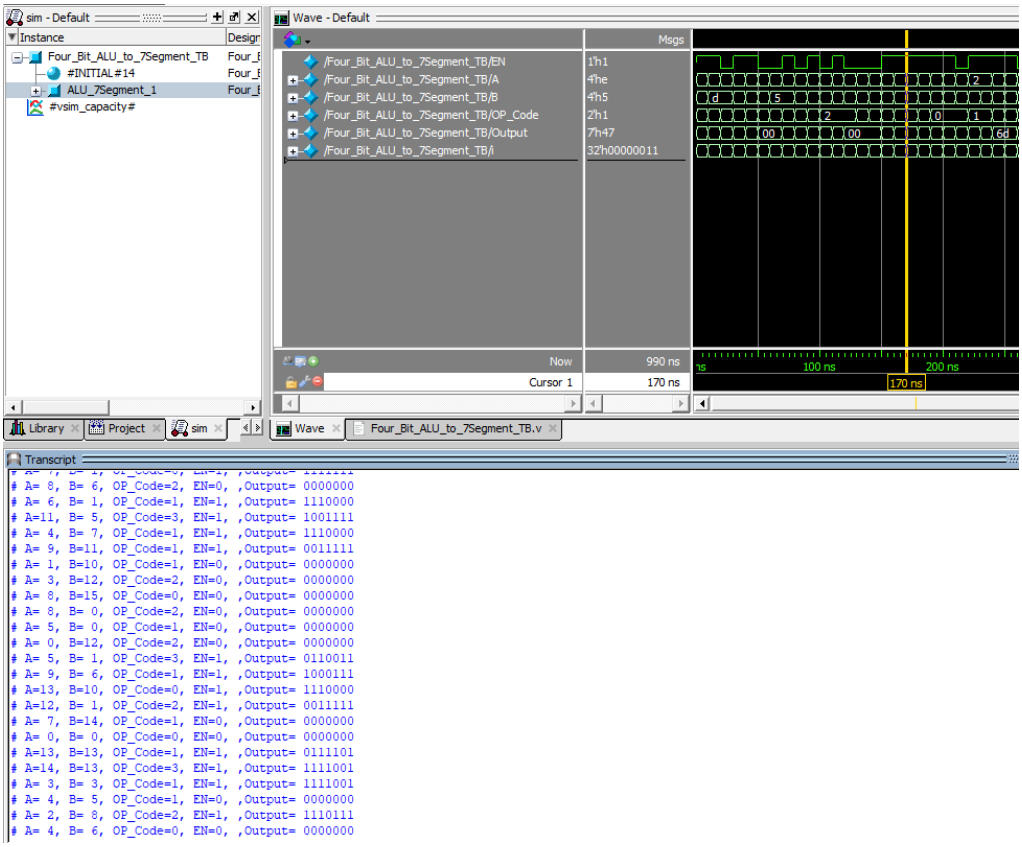
Four_Bit_ALU_to_7Segment.v | Four_Bit_ALU_to_7Segment_TB.v

```
    N_bit_ALU #(Width) ALU2 (A, B, OP_Code, ALU_Output);

    always@(ALU_Output or EN)begin
        if(EN == 1) begin
            case(ALU_Output)
                0 : Output = 7'b1111110;
                1 : Output = 7'b0110000;
                2 : Output = 7'b1101101;
                3 : Output = 7'b1111001;
                4 : Output = 7'b0110011;
                5 : Output = 7'b1011011;
                6 : Output = 7'b1011111;
                7 : Output = 7'b1110000;
                8 : Output = 7'b1111111;
                9 : Output = 7'b1111011;
                10: Output = 7'b1110111;
                11: Output = 7'b0011111;
                12: Output = 7'b1001110;
                13: Output = 7'b0111101;
                14: Output = 7'b1001111;
                15: Output = 7'b1000111;
                default: Output = 0;
            endcase
        end
        else begin
            Output = 0;
        end
    end
endmodule
```

Test Bench:

```
Four_Bit_ALU_to_7Segment.v   Four_Bit_ALU_to_7Segment_TB.v
module Four_Bit_ALU_to_7Segment_TB();

    parameter Width1 = 4;

    reg  EN;
    reg  [Width1-1:0] A, B;
    reg  [1:0]OP_Code;
    wire [6:0]Output;

    Four_Bit_ALU_to_7Segment #(Width1) ALU_7Segment_1 (A, B, OP_Code, EN, Output);

    integer i;

    initial begin
        for(i=0;i<99;i=i+1) begin
            A = $random;
            B = $random;
            EN = $random;
            OP_Code = $random;
            #10;
        end
        $stop;
    end

    initial begin
        $monitor("A=%d, B=%d, OP_Code=%d, EN=%d,Output= %b",A,B,OP_Code,EN,Output);
        $stop;
    end
endmodule
```

Simulation:



Question 5

Code:

D_flipflop.v

```
module D_flipflop (D, E, CLK, Q);

    input  D, E, CLK;
    output reg Q;

    always @( posedge CLK) begin
        if(E) begin
            Q <= D;
        end
        else begin
            Q <= Q;
        end
    end

endmodule
```