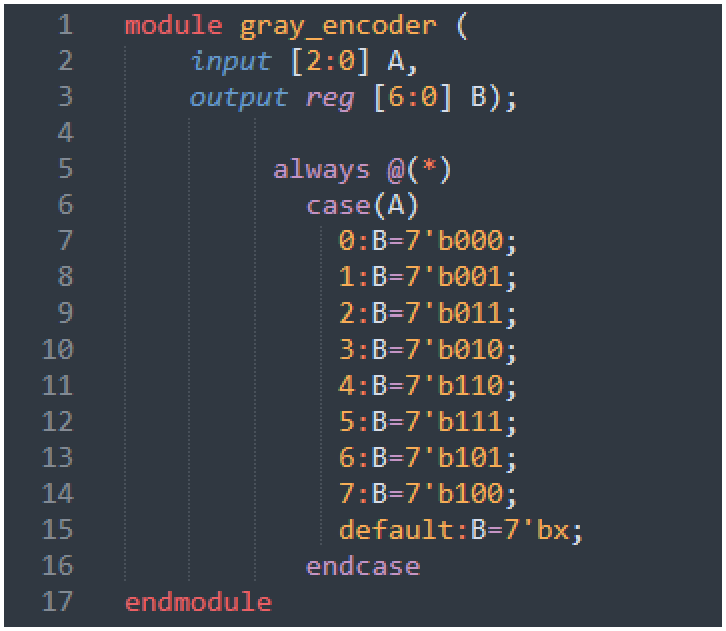
Assignment 2 Extra

By Amr Mohamed Ahmed Ebrahim

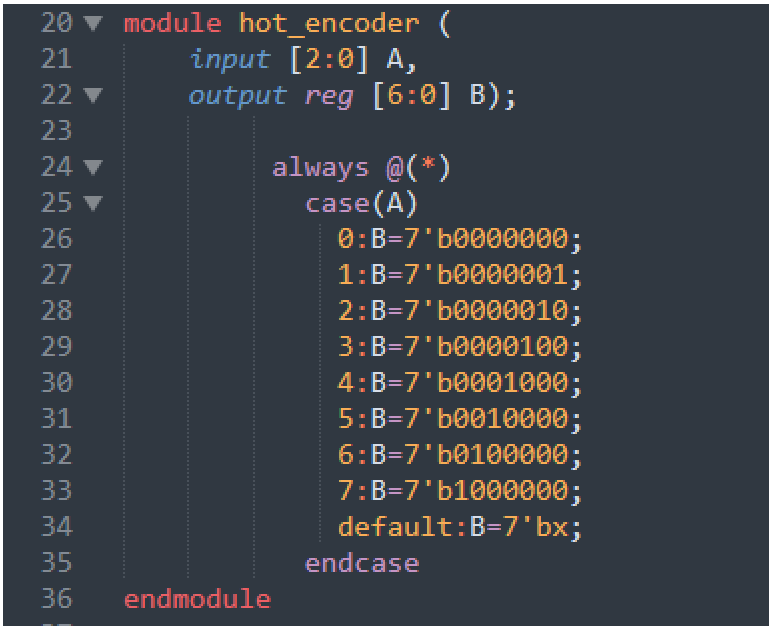
Question 1

Gray Encoder Code



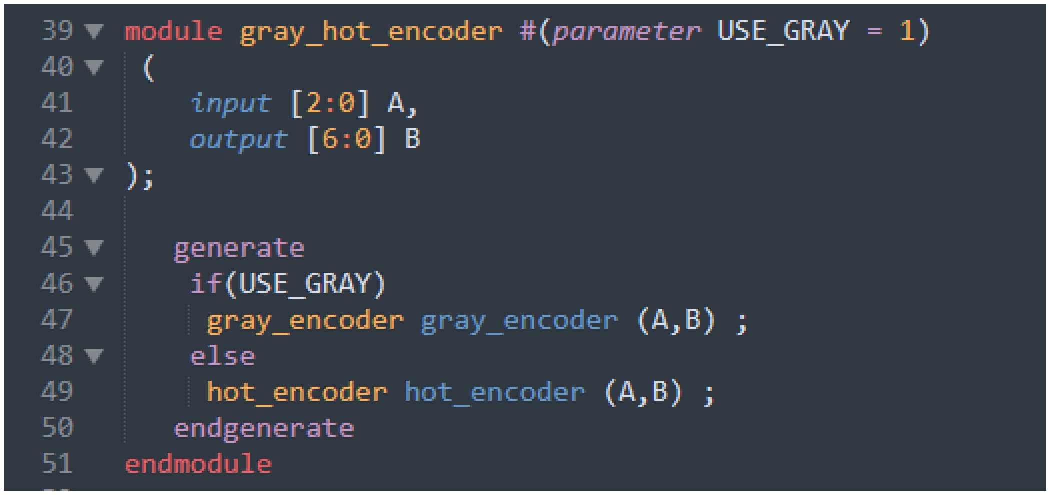
```
module gray_encoder (
    input [2:0] A,
    output reg [6:0] B);

         always @(*)
           case(A)
             0:B=7'b000;
             1:B=7'b001;
             2:B=7'b011;
             3:B=7'b010;
             4:B=7'b110;
             5:B=7'b111;
             6:B=7'b101;
             7:B=7'b100;
             default:B=7'bx;
           endcase
endmodule
```

Hot Encoder



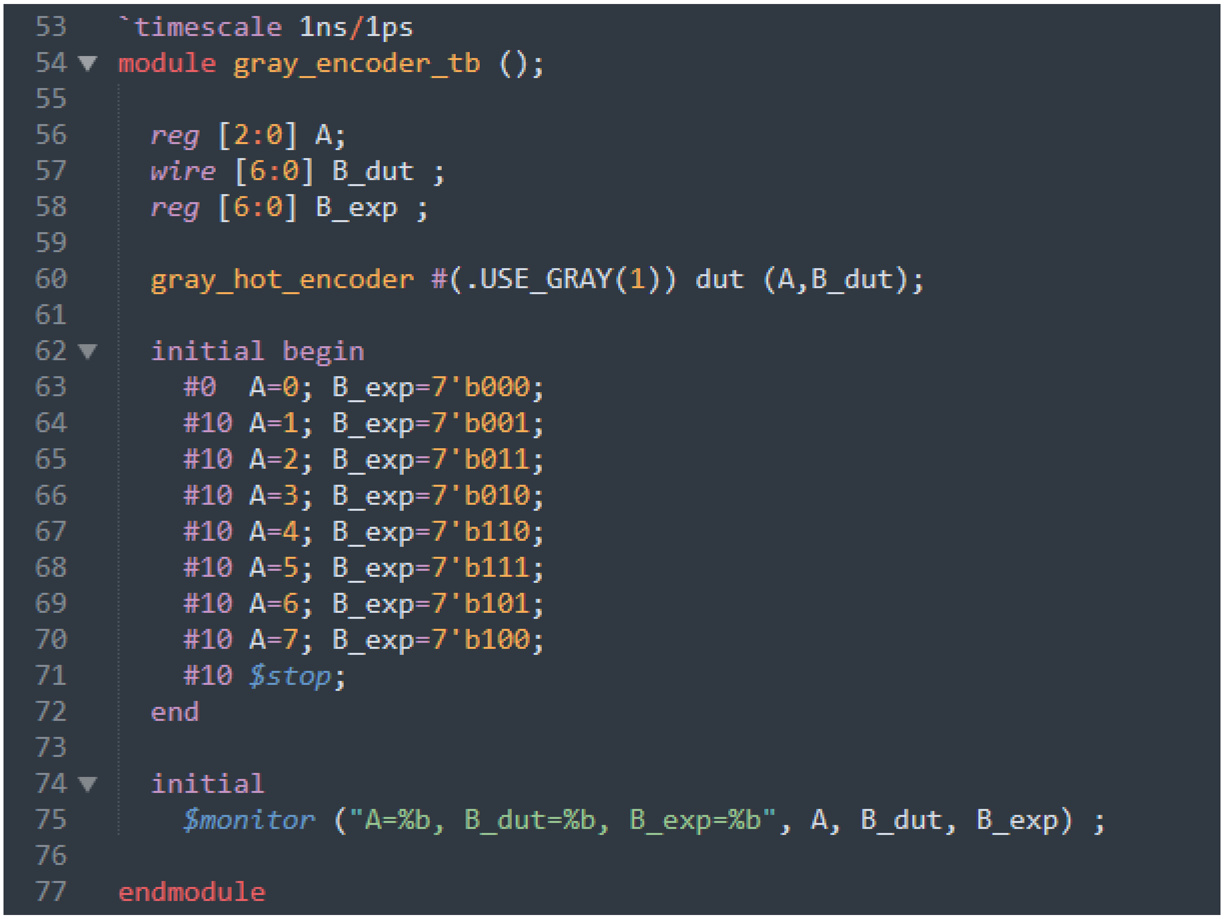
```
module hot_encoder (
    input [2:0] A,
    output reg [6:0] B);

         always @(*)
           case(A)
             0:B=7'b0000000;
             1:B=7'b0000001;
             2:B=7'b0000010;
             3:B=7'b0000100;
             4:B=7'b0001000;
             5:B=7'b0010000;
             6:B=7'b0100000;
             7:B=7'b1000000;
             default:B=7'bx;
           endcase
endmodule
```

Gray Hot Encoder



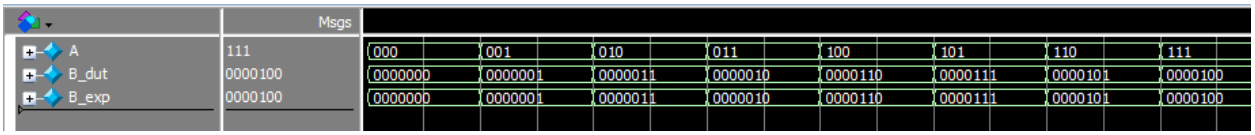
```
module gray_hot_encoder #(parameter USE_GRAY = 1)
 (
    input [2:0] A,
    output [6:0] B
);

   generate
    if(USE_GRAY)
     gray_encoder gray_encoder (A,B) ;
    else
     hot_encoder hot_encoder (A,B) ;
   endgenerate
endmodule
```

Gray Encoder Testbench

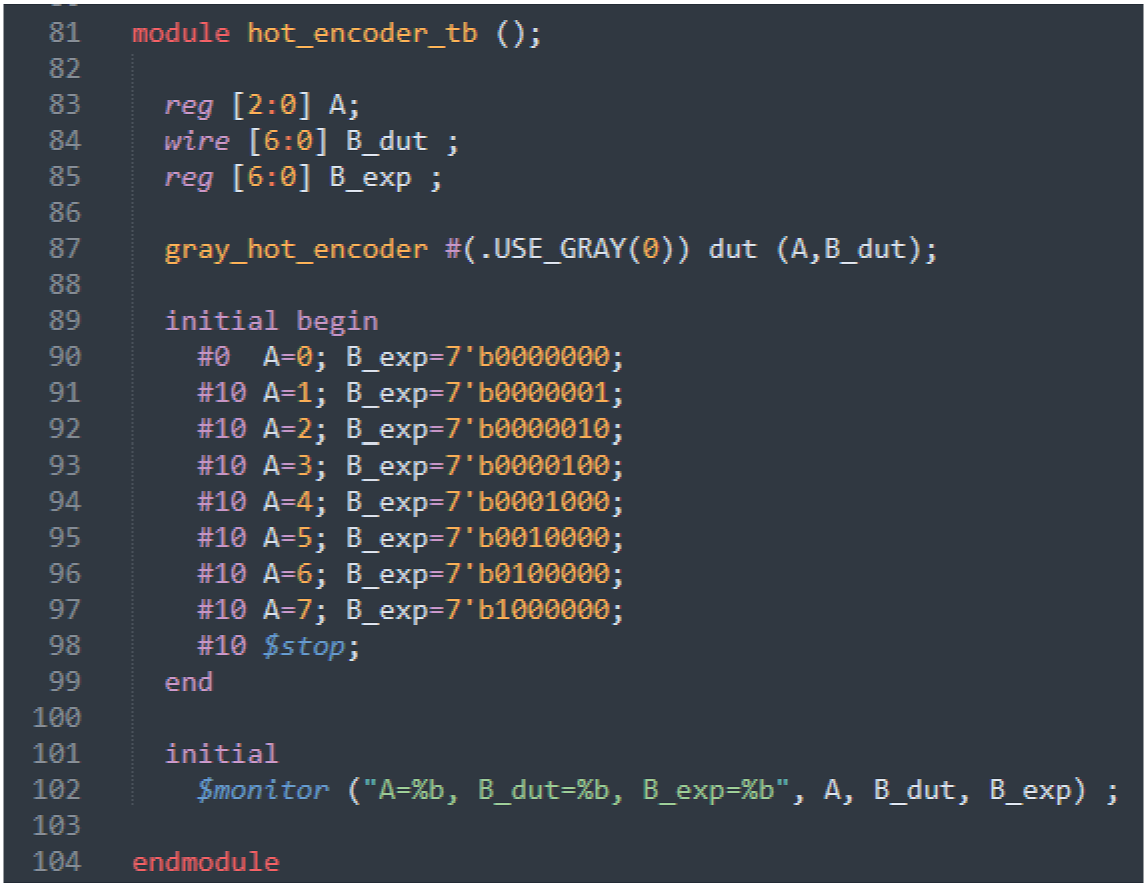


```
`timescale 1ns/1ps
module gray_encoder_tb ();

  reg [2:0] A;
  wire [6:0] B_dut ;
  reg [6:0] B_exp ;

  gray_hot_encoder #(.USE_GRAY(1)) dut (A,B_dut);

  initial begin
    #0  A=0; B_exp=7'b000;
    #10 A=1; B_exp=7'b001;
    #10 A=2; B_exp=7'b011;
    #10 A=3; B_exp=7'b010;
    #10 A=4; B_exp=7'b110;
    #10 A=5; B_exp=7'b111;
    #10 A=6; B_exp=7'b101;
    #10 A=7; B_exp=7'b100;
    #10 $stop;
  end

  initial
    $monitor ("A=%b, B_dut=%b, B_exp=%b", A, B_dut, B_exp) ;

endmodule
```

Gray Encoder TestBench Wave

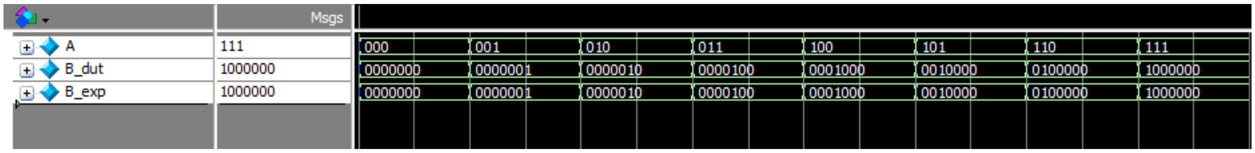


Hot Encoder TestBench



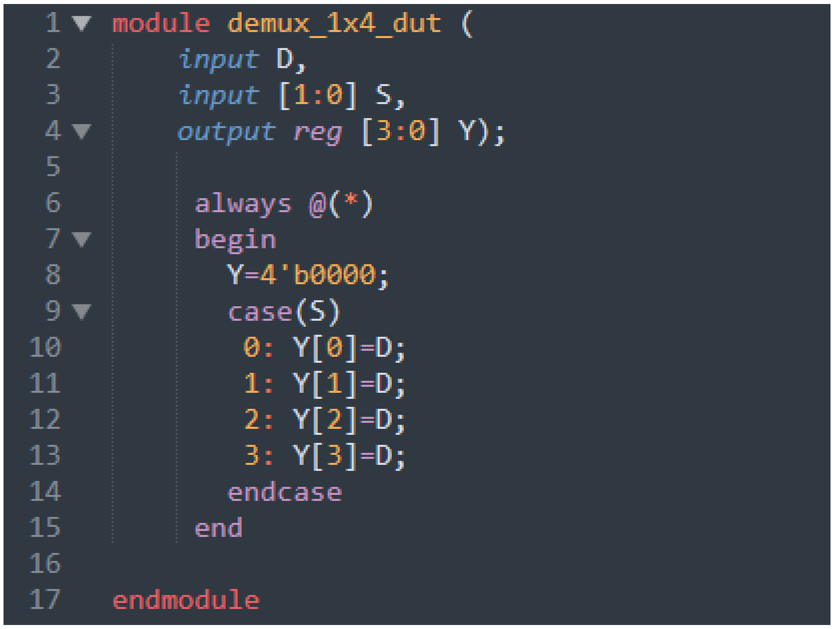
```
module hot_encoder_tb ();

  reg [2:0] A;
  wire [6:0] B_dut ;
  reg [6:0] B_exp ;

  gray_hot_encoder #(.USE_GRAY(0)) dut (A,B_dut);

  initial begin
    #0  A=0; B_exp=7'b0000000;
    #10 A=1; B_exp=7'b0000001;
    #10 A=2; B_exp=7'b0000010;
    #10 A=3; B_exp=7'b0000100;
    #10 A=4; B_exp=7'b0001000;
    #10 A=5; B_exp=7'b0010000;
    #10 A=6; B_exp=7'b0100000;
    #10 A=7; B_exp=7'b1000000;
    #10 $stop;
  end

  initial
    $monitor ("A=%b, B_dut=%b, B_exp=%b", A, B_dut, B_exp) ;

endmodule
```

Hot Encoder TestBench Wave



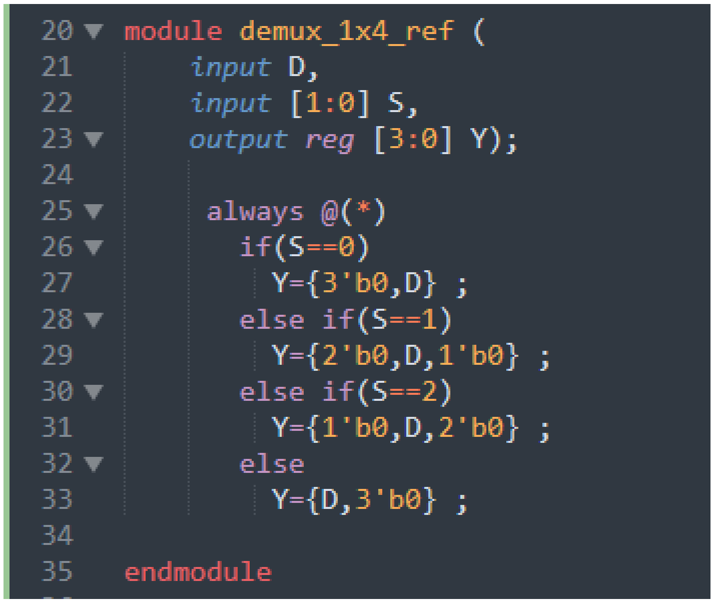
Question 2

DUT Code



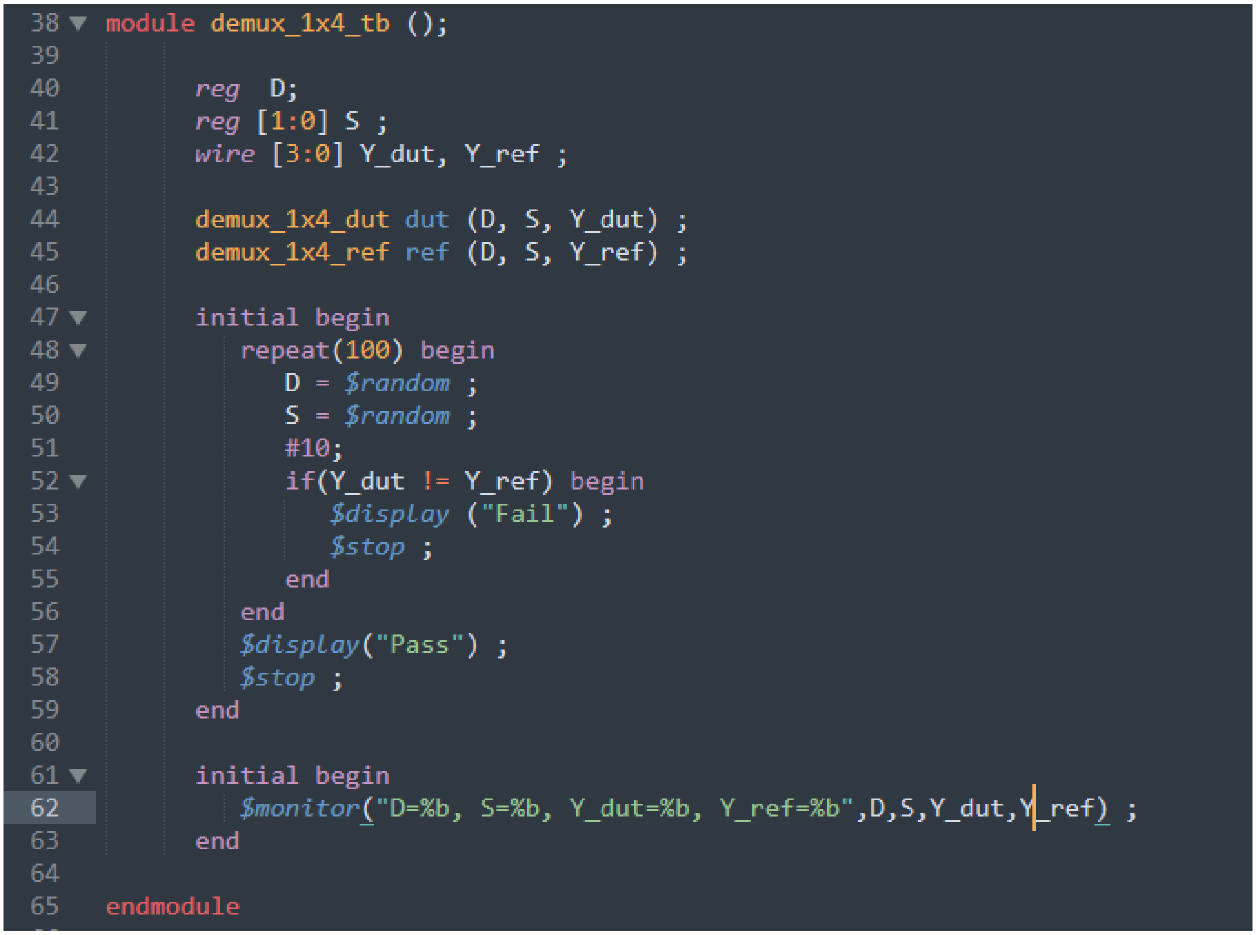
```
module demux_1x4_dut (
    input D,
    input [1:0] S,
    output reg [3:0] Y);

     always @(*)
     begin
       Y=4'b0000;
       case(S)
        0: Y[0]=D;
        1: Y[1]=D;
        2: Y[2]=D;
        3: Y[3]=D;
       endcase
     end

endmodule
```

REF Code



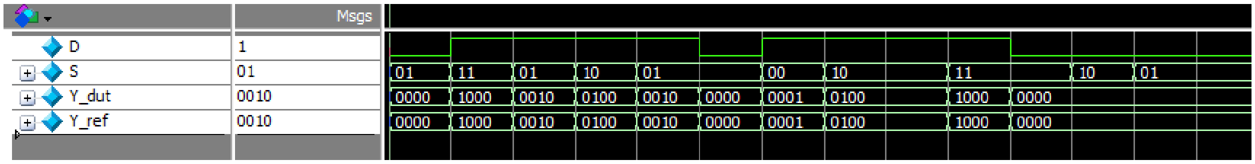
```
module demux_1x4_ref (
    input D,
    input [1:0] S,
    output reg [3:0] Y);

     always @(*)
       if(S==0)
         Y={3'b0,D} ;
       else if(S==1)
         Y={2'b0,D,1'b0} ;
       else if(S==2)
         Y={1'b0,D,2'b0} ;
       else
         Y={D,3'b0} ;

endmodule
```

TestBench Code



```verilog
module demux_1x4_tb ();

        reg  D;
        reg [1:0] S ;
        wire [3:0] Y_dut, Y_ref ;

        demux_1x4_dut dut (D, S, Y_dut) ;
        demux_1x4_ref ref (D, S, Y_ref) ;

        initial begin
           repeat(100) begin
              D = $random ;
              S = $random ;
              #10;
              if(Y_dut != Y_ref) begin
                 $display ("Fail") ;
                 $stop ;
              end
           end
           $display("Pass") ;
           $stop ;
        end

        initial begin
           $monitor("D=%b, S=%b, Y_dut=%b, Y_ref=%b",D,S,Y_dut,Y_ref) ;
        end

endmodule
```

TestBench Wave



	Msgs
D	1
S	01
Y_dut	0010
Y_ref	0010