Verilog HDL

Assignment 2 Extended

By: Magdy Ahmed Abbas Abdelhamid

Question 1

Gray Coding

Verilog Code:

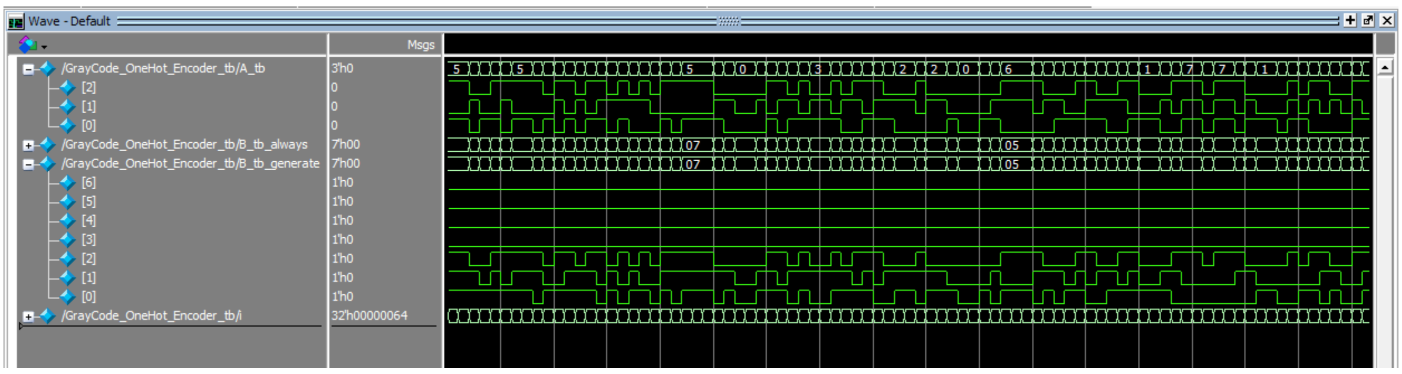
```
module GrayCode_OneHot_Encoder_always (A, B);

parameter USE_GRAY = 1;
input [2:0] A;
output reg [6:0] B;

always @(*) begin
  if (USE_GRAY)
    case (A)
      3'b000 : B = 7'b0000_000;
      3'b001 : B = 7'b0000_001;
      3'b010 : B = 7'b0000_011;
      3'b011 : B = 7'b0000_010;
      3'b100 : B = 7'b0000_110;
      3'b101 : B = 7'b0000_111;
      3'b110 : B = 7'b0000_101;
      3'b111 : B = 7'b0000_100;
      default : B = 7'b0000_000;
    endcase
  else
    case (A)
      3'b000 : B = 7'b0000_000;
      3'b001 : B = 7'b0000_001;
      3'b010 : B = 7'b0000_010;
      3'b011 : B = 7'b0000_100;
      3'b100 : B = 7'b0001_000;
      3'b101 : B = 7'b0010_000;
      3'b110 : B = 7'b0100_000;
      3'b111 : B = 7'b1000_000;
      default : B = 7'b0000_000;
    endcase
  end

endmodule
```

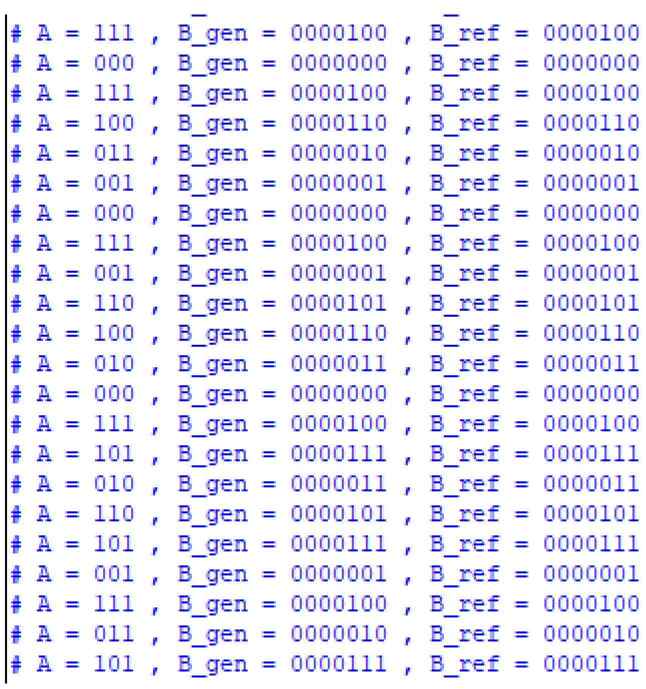
```
module GrayCode_OneHot_Encoder_generate (A, B);

parameter USE_GRAY = 1;
input [2:0] A;
output reg [6:0] B;

generate
  if (USE_GRAY)
    always @(A)
      case (A)
        3'b000 : B = 7'b0000_000;
        3'b001 : B = 7'b0000_001;
        3'b010 : B = 7'b0000_011;
        3'b011 : B = 7'b0000_010;
        3'b100 : B = 7'b0000_110;
        3'b101 : B = 7'b0000_111;
        3'b110 : B = 7'b0000_101;
        3'b111 : B = 7'b0000_100;
        default : B = 7'b0000_000;
      endcase
  else
    always @(A)
      case (A)
        3'b000 : B = 7'b0000_000;
        3'b001 : B = 7'b0000_001;
        3'b010 : B = 7'b0000_010;
        3'b011 : B = 7'b0000_100;
        3'b100 : B = 7'b0001_000;
        3'b101 : B = 7'b0010_000;
        3'b110 : B = 7'b0100_000;
        3'b111 : B = 7'b1000_000;
        default : B = 7'b0000_000;
      endcase
endgenerate
endmodule
```

Testbench Code:

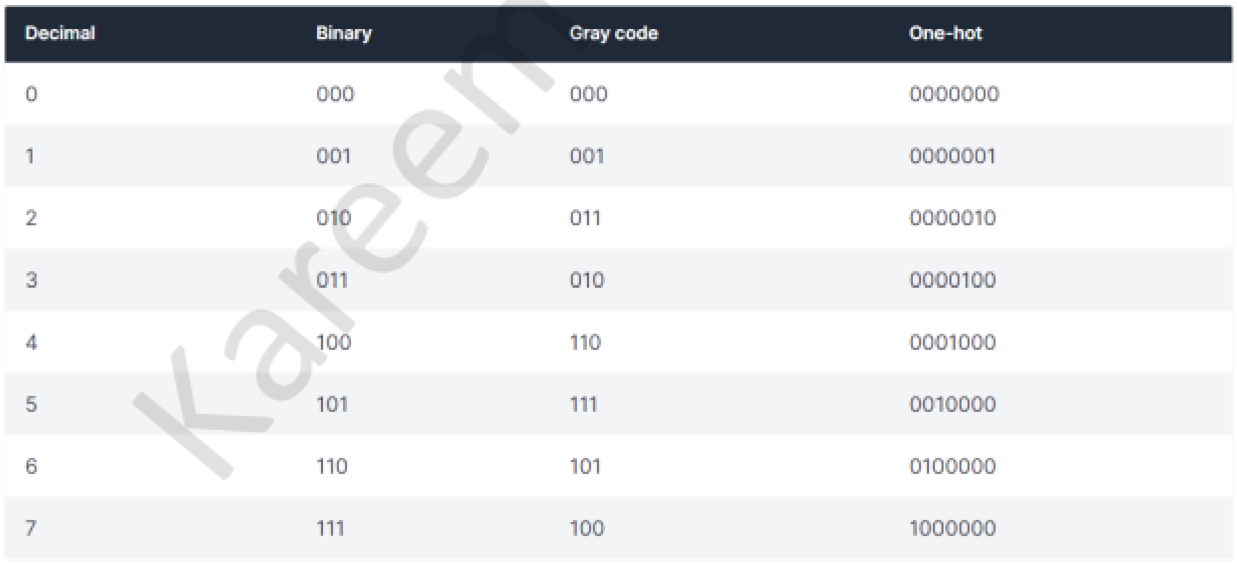
```
module GrayCode_OneHot_Encoder_tb ();

parameter use_gray = 1;
reg [2: 0] A_tb;
wire [6: 0] B_tb_always, B_tb_generate;

GrayCode_OneHot_Encoder_always #(.USE_GRAY(use_gray)) DUT1 (.A(A_tb), .B(B_tb_always));
GrayCode_OneHot_Encoder_generate #(.USE_GRAY(use_gray)) DUT2 (.A(A_tb), .B(B_tb_generate));

integer i;
initial begin
   for (i = 0 ; i < 100 ; i = i + 1) begin
      A_tb = $random;
      #10;
      if (B_tb_always ! = B_tb_generate) begin
         $display ("The Encoding Design is Wrong! ");
         $stop;
      end
   end
   $stop;
end

initial begin
   $monitor("A = %b , B_gen = %b , B_ref = %b", A_tb, B_tb_generate, B_tb_always);
end

endmodule
```

Waveform:


Wave - Default
Msgs
/GrayCode_OneHot_Encoder_tb/A_tb 3'h0
[2] 0
[1] 0
[0] 0
/GrayCode_OneHot_Encoder_tb/B_tb_always 7'h00
/GrayCode_OneHot_Encoder_tb/B_tb_generate 7'h00
[6] 1'h0
[5] 1'h0
[4] 1'h0
[3] 1'h0
[2] 1'h0
[1] 1'h0
[0] 1'h0
/GrayCode_OneHot_Encoder_tb/i 32'h00000064



```
# A = 111 , B_gen = 0000100 , B_ref = 0000100
# A = 000 , B_gen = 0000000 , B_ref = 0000000
# A = 111 , B_gen = 0000100 , B_ref = 0000100
# A = 100 , B_gen = 0000110 , B_ref = 0000110
# A = 011 , B_gen = 0000010 , B_ref = 0000010
# A = 001 , B_gen = 0000001 , B_ref = 0000001
# A = 000 , B_gen = 0000000 , B_ref = 0000000
# A = 111 , B_gen = 0000100 , B_ref = 0000100
# A = 001 , B_gen = 0000001 , B_ref = 0000001
# A = 110 , B_gen = 0000101 , B_ref = 0000101
# A = 100 , B_gen = 0000110 , B_ref = 0000110
# A = 010 , B_gen = 0000011 , B_ref = 0000011
# A = 000 , B_gen = 0000000 , B_ref = 0000000
# A = 111 , B_gen = 0000100 , B_ref = 0000100
# A = 101 , B_gen = 0000111 , B_ref = 0000111
# A = 010 , B_gen = 0000011 , B_ref = 0000011
# A = 110 , B_gen = 0000101 , B_ref = 0000101
# A = 101 , B_gen = 0000111 , B_ref = 0000111
# A = 001 , B_gen = 0000001 , B_ref = 0000001
# A = 111 , B_gen = 0000100 , B_ref = 0000100
# A = 011 , B_gen = 0000010 , B_ref = 0000010
# A = 101 , B_gen = 0000111 , B_ref = 0000111
```



Decimal	Binary	Gray code	One-hot
0	000	000	0000000
1	001	001	0000001
2	010	011	0000010
3	011	010	0000100
4	100	110	0001000
5	101	111	0010000
6	110	101	0100000
7	111	100	1000000

Question 2

Design a 1 - 4 Demultiplexer.

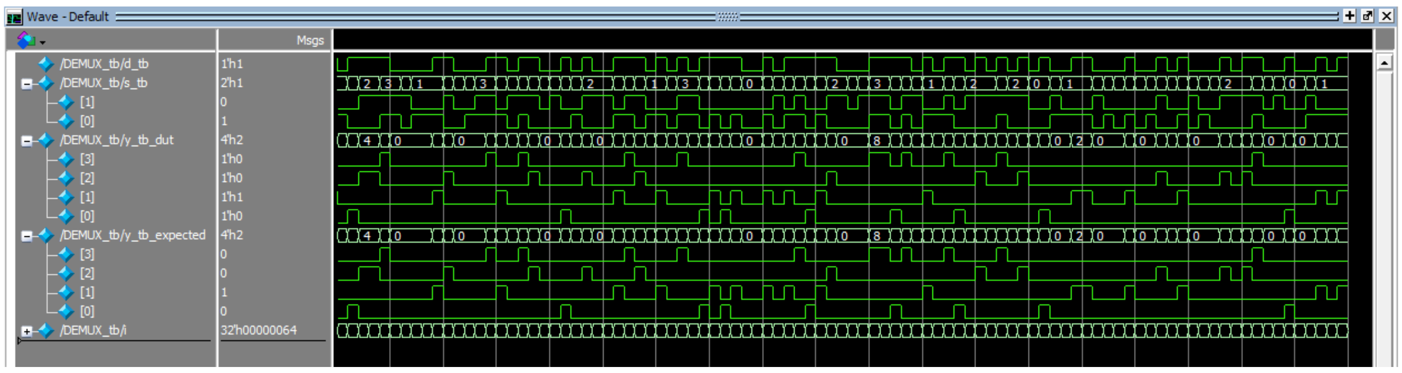
Verilog Code:

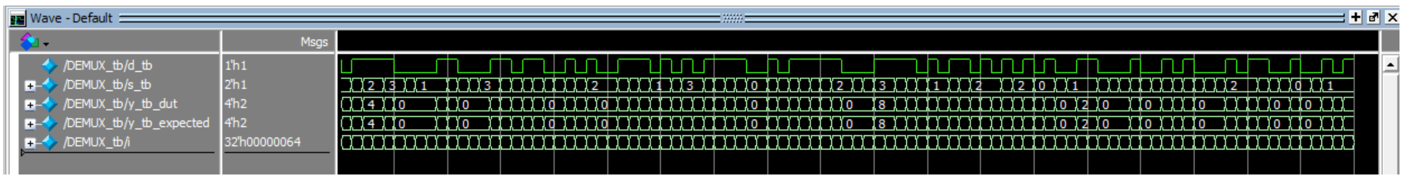
```
module DEMUX (D, S, Y);

input D;
input [1: 0] S;
output reg [3: 0] Y;

always @(*) begin
   case (S)
      2′b00 :  Y  =  {3′b000 , D};
      2′b01 :  Y  =  {2′b00 , D , 1′b0};
      2′b10 :  Y  =  {1′b0 , D , 2′b00};
      2′b11 :  Y  =  {D , 3′b000};
   endcase
end

endmodule
```

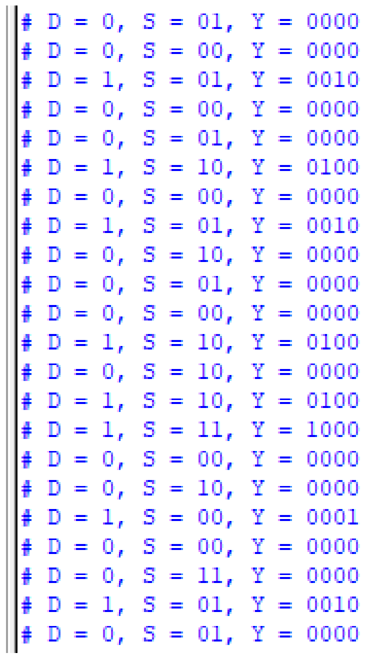
Testbench Code:

```
module DEMUX_tb ();

reg d_tb;
reg [1:0] s_tb;
wire [3:0] y_tb_dut;
reg [3:0] y_tb_expected;

DEMUX DUT(.D(d_tb),.S(s_tb),.Y(y_tb_dut));

integer i;
initial begin
  for (i = 0 ; i < 100 ; i = i + 1) begin
    d_tb = $random;
    s_tb = $random;

    y_tb_expected[3] = s_tb[1] & s_tb[0] & d_tb;
    y_tb_expected[2] = s_tb[1] & ~s_tb[0] & d_tb;
    y_tb_expected[1] = ~s_tb[1] & s_tb[0] & d_tb;
    y_tb_expected[0] = ~s_tb[1] & ~s_tb[0] & d_tb;
    #10;
    if (y_tb_dut != y_tb_expected) begin
      $display("The DEMUX Design is Wrong! ");
      $stop;
    end
  end
  $stop;
end

initial begin
  $monitor("D = %b, S = %b, Y = %b ", d_tb, s_tb, y_tb_dut);
end

endmodule
```

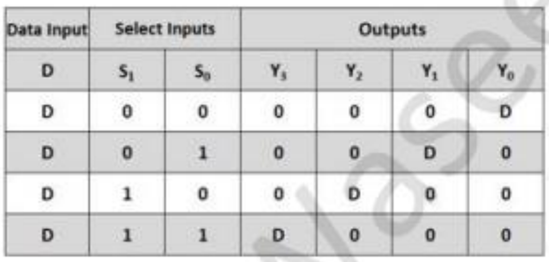
Waveform:



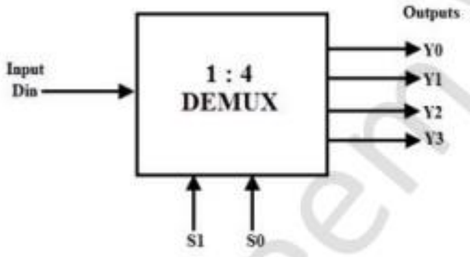


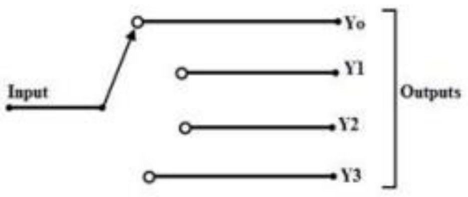


```
# D = 0, S = 01, Y = 0000
# D = 0, S = 00, Y = 0000
# D = 1, S = 01, Y = 0010
# D = 0, S = 00, Y = 0000
# D = 0, S = 01, Y = 0000
# D = 1, S = 10, Y = 0100
# D = 0, S = 00, Y = 0000
# D = 1, S = 01, Y = 0010
# D = 0, S = 10, Y = 0000
# D = 0, S = 01, Y = 0000
# D = 0, S = 00, Y = 0000
# D = 1, S = 10, Y = 0100
# D = 0, S = 10, Y = 0000
# D = 1, S = 10, Y = 0100
# D = 1, S = 11, Y = 1000
# D = 0, S = 00, Y = 0000
# D = 0, S = 10, Y = 0000
# D = 1, S = 00, Y = 0001
# D = 0, S = 00, Y = 0000
# D = 0, S = 11, Y = 0000
# D = 1, S = 01, Y = 0010
# D = 0, S = 01, Y = 0000
```



Data Input	Select Inputs		Outputs			
D	S_1	S_0	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0


Input
Din
1 : 4
DEMUX
Outputs
Y0
Y1
Y2
Y3
S1
S0


Input
Yo
Y1
Y2
Y3
Outputs