

Assignment_2_Extra

Name: Ahmed Nour

Sheet ID: 16

Email: ahmedmohamednoour@gmail.com

Q1)

Models Code:

```
1 module Gray_encoder (B_g, A_g);
2     input [2:0] A_g;
3     reg [2:0] gray_T;
4     output reg [6:0] B_g;
5
6     always @(*) begin
7         case(A_g)
8             3'd0 : gray_T = 3'b000;
9             3'd1 : gray_T = 3'b001;
10            3'd2 : gray_T = 3'b011;
11            3'd3 : gray_T = 3'b010;
12            3'd4 : gray_T = 3'b110;
13            3'd5 : gray_T = 3'b111;
14            3'd6 : gray_T = 3'b101;
15            3'd7 : gray_T = 3'b100;
16            default: gray_T = 3'b000;
17        endcase
18        B_g = {4'b0 , gray_T};
19    end
20 endmodule
```

```
1 module Onehot_encoder (B_oh, A_oh);
2     input [2:0] A_oh;
3     output reg [6:0] B_oh;
4
5     always @(*) begin
6         B_oh = 0;
7         case(A_oh)
8             3'd0: B_oh = 7'b000_0000;
9             default: B_oh[A_oh - 1] = B_oh[A_oh - 1] | 1;
10        endcase
11    end
12 endmodule
```

```
1 module Gray_onehot_encoder (B, A);
2     parameter USE_GRAY =1;
3
4     input [2:0] A;
5     output [6:0] B;
6
7     generate
8         if (USE_GRAY == 1) begin
9             Gray_encoder G_encoder1 (B, A);
10            //this encoder get the 3-bit code and place it in the 3 Least bits in B, as B is 7 bits in the specs.
11        end
12        else begin
13            Onehot_encoder Oh_encoder2 (B, A);
14        end
15    endgenerate
16
17 endmodule
```

Testbench Codes:

```
1 module Gray_encoder_tb_exhaustive();
2     parameter USE_GRAY_tb = 1;
3     reg [2:0] A_tb;
4     reg [2:0] gray_tb;
5     reg [6:0] B_tb;
6     wire [6:0] B_dut;
7
8     Gray_onehot_encoder #(USE_GRAY_tb) DUT (B_dut, A_tb);
9
10    integer i;
11    initial begin
12        for (i=0; i<8; i=i+1) begin
13            A_tb = i;
14            case(A_tb)
15                3'd0 : gray_tb = 3'b000;
16                3'd1 : gray_tb = 3'b001;
17                3'd2 : gray_tb = 3'b011;
18                3'd3 : gray_tb = 3'b010;
19                3'd4 : gray_tb = 3'b110;
20                3'd5 : gray_tb = 3'b111;
21                3'd6 : gray_tb = 3'b101;
22                3'd7 : gray_tb = 3'b100;
23            endcase
24            B_tb = {4'b0 , gray_tb};
25            #10;
26            $display("Value of golden model = %b , Value of DUT = %b", B_tb, B_dut);
27        end
28    end
29
30 endmodule
```

```
1 module Onehot_encoder_tb_exhaustive();
2     parameter USE_GRAY_tb = 0;
3     reg [2:0] A_tb;
4     reg [2:0] gray_tb;
5     reg [6:0] B_tb;
6     wire [6:0] B_dut;
7
8     Gray_onehot_encoder #(USE_GRAY_tb) DUT (B_dut, A_tb);
9
10    integer i;
11    initial begin
12        for (i=0; i<8; i=i+1) begin
13            A_tb = i;
14            case(A_tb)
15                3'd0 : B_tb = 7'b000_0000;
16                3'd1 : B_tb = 7'b000_0001;
17                3'd2 : B_tb = 7'b000_0010;
18                3'd3 : B_tb = 7'b000_0100;
19                3'd4 : B_tb = 7'b000_1000;
20                3'd5 : B_tb = 7'b001_0000;
21                3'd6 : B_tb = 7'b010_0000;
22                3'd7 : B_tb = 7'b100_0000;
23            endcase
24            #10;
25            $display("Value of golden model = %b , Value of DUT = %b", B_tb, B_dut);
26        end
27    end
28
29 endmodule
```

Simulation:

Grey Encoder test, Successful.

	Msgs								
/Gray_encoder_tb_exhaustive/USE...	-No Data-	00000001							
/Gray_encoder_tb_exhaustive/i	-No Data-	00000000	00000001	00000002	00000003	00000004	00000005	00000006	00000007
/Gray_encoder_tb_exhaustive/A_tb	-No Data-	0	1	2	3	4	5	6	7
/Gray_encoder_tb_exhaustive/B_dut	-No Data-	00	01	03	02	06	07	05	04
/Gray_encoder_tb_exhaustive/B_tb	-No Data-	00	01	03	02	06	07	05	04

```
VSIM 7> run -all
# Value of golden model = 0000000 , Value of DUT = 0000000
# Value of golden model = 0000001 , Value of DUT = 0000001
# Value of golden model = 0000011 , Value of DUT = 0000011
# Value of golden model = 0000010 , Value of DUT = 0000010
# Value of golden model = 0000110 , Value of DUT = 0000110
# Value of golden model = 0000111 , Value of DUT = 0000111
# Value of golden model = 0000101 , Value of DUT = 0000101
# Value of golden model = 0000100 , Value of DUT = 0000100
```


One-hot Encoder test, Successful.

	Msgs								
/Onehot_encoder_tb_exhaustive/USE_GR...	-No Data-	00000000							
/Onehot_encoder_tb_exhaustive/i	-No Data-	00000000	00000001	00000002	00000003	00000004	00000005	00000006	00000007
/Onehot_encoder_tb_exhaustive/A_tb	-No Data-	0	1	2	3	4	5	6	7
/Onehot_encoder_tb_exhaustive/B_dut	-No Data-	00	01	02	04	08	10	20	40
/Onehot_encoder_tb_exhaustive/B_tb	-No Data-	00	01	02	04	08	10	20	40

```
VSIM 19> run -all
# Value of golden model = 0000000 , Value of DUT = 0000000
# Value of golden model = 0000001 , Value of DUT = 0000001
# Value of golden model = 0000010 , Value of DUT = 0000010
# Value of golden model = 0000100 , Value of DUT = 0000100
# Value of golden model = 0001000 , Value of DUT = 0001000
# Value of golden model = 0010000 , Value of DUT = 0010000
# Value of golden model = 0100000 , Value of DUT = 0100000
# Value of golden model = 1000000 , Value of DUT = 1000000
```

Q2)

Code:



```
1  module demux14 (y, s, d);
2  input d;
3  input [1:0] s;
4  output reg [3:0] y;
5
6  always @(*) begin
7      y = 0;
8      case(s)
9          2'b00: y[0] = d;
10         2'b01: y[1] = d;
11         2'b10: y[2] = d;
12         default: y[3] = d;
13     endcase
14 end
15 endmodule
```

Testbench:

```
1 module demux_tb_exhaustive_self_checking ();
2 reg d_tb;
3 reg [1:0] s_tb;
4 reg [3:0] y_exp, temp1;
5 wire [3:0] y_dut;
6
7 demux14 DUT (y_dut, s_tb, d_tb);
8
9
10 integer i, j;
11 initial begin
12     temp1 = 2'b01;
13     for(i = 0; i < 4; i=i+1)begin
14         s_tb = i;
15         for(j = 0; j < 2; j=j+1)begin
16             d_tb = temp1[j];
17             if(s_tb == 0)
18                 y_exp = {3'b000, d_tb};
19             else if(s_tb == 1)
20                 y_exp = {2'b00, d_tb, 1'b0};
21             else if(s_tb == 2)
22                 y_exp = {1'b0, d_tb, 2'b00};
23             else
24                 y_exp = {d_tb, 3'b000};
25             #10;
26             if(y_dut != y_exp)begin
27                 $display("Error at %b in the demux", s_tb);
28                 $stop;
29             end
30         end
31     end
32     $display("No errors found");
33
34 end
35
36 endmodule
```

Simulation:

Msgs													
+	/demux_tb_exhaus...	-No Data-											
+	/demux_tb_exhaus...	-No Data-	00000000		00000001			00000002			00000003		
+	/demux_tb_exhaus...	-No Data-	00000000	00000001	00000000	00000001	00000000	00000001	00000000	00000001	00000000	00000001	
+	/demux_tb_exhaus...	-No Data-	0		1		2		3				
+	/demux_tb_exhaus...	-No Data-	1	0	2	0	4	0	8	0			
+	/demux_tb_exhaus...	-No Data-	1	0	2	0	4	0	8	0			

```
# .main_pane.wave.interior.cs.body.pw.wf
add wave -position insertpoint \
sim:/demux_tb_exhaustive_self_checking/d_tb \
sim:/demux_tb_exhaustive_self_checking/i \
sim:/demux_tb_exhaustive_self_checking/j \
sim:/demux_tb_exhaustive_self_checking/s_tb \
sim:/demux_tb_exhaustive_self_checking/y_dut \
sim:/demux_tb_exhaustive_self_checking/y_exp
VSIM 246> run -all
# No errors found
```