

# DIGITAL SYSTEM DESIGN (3EL42)

## ASSIGNMENT – 1

Q-1: Write a Verilog code for 2X4 decoder.

CODE :

| testbench.sv  | design.sv   |
|---|---|
| <pre> 1 //2*4 decoder (21EL026) 2 module tb; 3   reg a,b,en; 4   wire [3:0]y; 5   decoder2_4 dut(en,a,b,y); 6 7   initial 8   begin 9     \$monitor("en=%b a=%b b=%b y=%b",en,a,b,y); 10 11    en=1;a=1'b1;b=1'b1;#5 12    en=0;a=0;b=0;#5 13    en=0;a=0;b=1;#5 14    en=0;a=1;b=0;#5 15    en=0;a=1;b=1;#5 16 17    \$finish; 18  end 19 endmodule </pre> | <pre> 1 module decoder2_4(en,a,b,y); 2   input en,a,b; 3 4   output reg [3:0]y; 5 6   always @(en,a,b) 7   begin 8     if(en==0) 9     begin 10      if(a==1'b0 &amp; b==1'b0) y=4'b1110; 11      else if(a==1'b0 &amp; b==1'b1) y=4'b1101; 12      else if(a==1'b1 &amp; b==1'b0) y=4'b1011; 13      else if(a==1 &amp; b==1) y=4'b0111; 14      else y=4'bxxxx; 15    end 16  else 17    y=4'b1111; 18  end 19 endmodule </pre> |

OUTPUT :

```

Log Share
[2022-08-07 05:34:07 EDT] iverilog '-wall' design.sv testbench.sv && unbuffer vvp a.out
en=1 a=x b=x y=1111
en=0 a=0 b=0 y=0000
en=0 a=0 b=1 y=0001
en=0 a=1 b=0 y=0101
en=0 a=1 b=1 y=1000
Done

```

Q-2: Write a Verilog code for Full subtractor.

CODE :

```

testbench.v
1 //full_sub (21EL026)
2 module tb_top;
3   reg a, b, Bin;
4   wire D, Bout;
5   |
6   full_subtractor(a, b, Bin, D, Bout);
7
8 initial begin
9   $monitor("At time %0t: a=%b b=%b, Bin=%b, difference=%b,
10  borrow=%b", $time, a,b,Bin,D,Bout);
11   a = 0; b = 0; Bin = 0; #1;
12   a = 0; b = 0; Bin = 1; #1;
13   a = 0; b = 1; Bin = 0; #1;
14   a = 0; b = 1; Bin = 1; #1;
15   a = 1; b = 0; Bin = 0; #1;
16   a = 1; b = 0; Bin = 1; #1;
17   a = 1; b = 1; Bin = 0; #1;
18   a = 1; b = 1; Bin = 1;
19 end
endmodule

design.v
1 module full_subtractor(input a, b, Bin, output D, Bout);
2   assign D = a ^ b ^ Bin;
3   assign Bout = (~a & b) | ~(a ^ b) & Bin;
4 endmodule
  
```

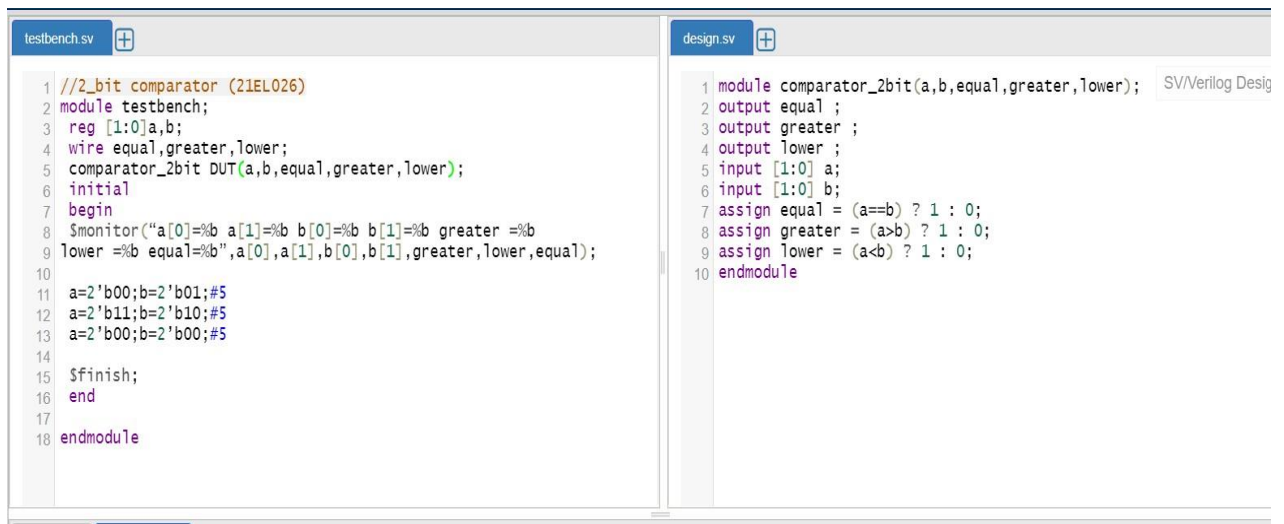
OUTPUT :

```

Log Share
[2022-08-07 05:47:04 EDT] iverilog '-wall' design.v testbench.v && unbuffer vvp a.out
a=0 b=0 cin=0 d=0 bout=0
a=0 b=1 cin=0 d=1 bout=1
a=1 b=0 cin=0 d=1 bout=0
a=1 b=1 cin=0 d=0 bout=0
a=1 b=1 cin=1 d=1 bout=1
Done
  
```

Q-3: Write a Verilog code for 2-bit comparator.

CODE:



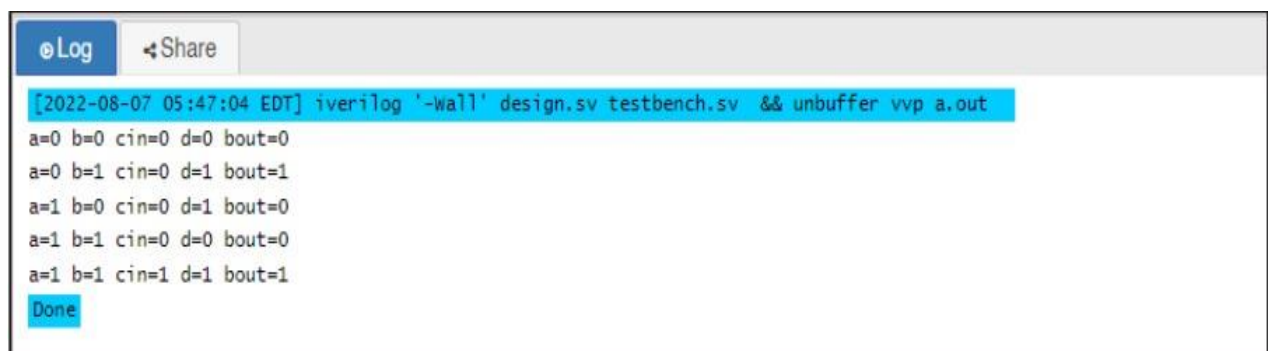
```

testbench.sv
1 //2_bit comparator (21EL026)
2 module testbench;
3   reg [1:0] a,b;
4   wire equal,greater,lower;
5   comparator_2bit DUT(a,b,equal,greater,lower);
6   initial
7   begin
8     $monitor("a[0]=%b a[1]=%b b[0]=%b b[1]=%b greater =%b
9     lower =%b equal=%b",a[0],a[1],b[0],b[1],greater,lower,equal);
10
11    a=2'b00;b=2'b01;#5
12    a=2'b11;b=2'b10;#5
13    a=2'b00;b=2'b00;#5
14
15    $finish;
16  end
17
18 endmodule

design.sv
1 module comparator_2bit(a,b,equal,greater,lower);
2   output equal ;
3   output greater ;
4   output lower ;
5   input [1:0] a;
6   input [1:0] b;
7   assign equal = (a==b) ? 1 : 0;
8   assign greater = (a>b) ? 1 : 0;
9   assign lower = (a<b) ? 1 : 0;
10 endmodule

```

OUTPUT:



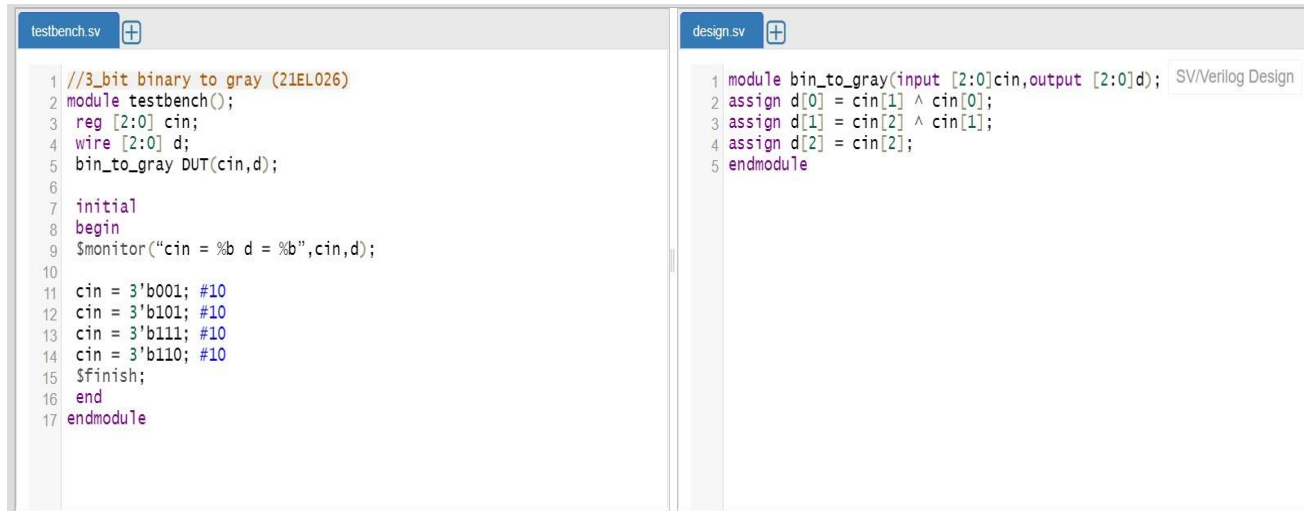
```

Log Share
[2022-08-07 05:47:04 EDT] iverilog '-wall' design.sv testbench.sv && unbuffer vvp a.out
a=0 b=0 cin=0 d=0 bout=0
a=0 b=1 cin=0 d=1 bout=1
a=1 b=0 cin=0 d=1 bout=0
a=1 b=1 cin=0 d=0 bout=0
a=1 b=1 cin=1 d=1 bout=1
Done

```

Q-4: Write a Verilog code for 3 bit binary to gray convertor.

CODE:



The screenshot shows two panels of a Verilog code editor. The left panel, titled 'testbench.sv', contains a testbench module that instantiates the 'bin\_to\_gray' module and applies test vectors. The right panel, titled 'design.sv', contains the 'bin\_to\_gray' module definition, which uses assign statements to calculate the Gray code output from the 3-bit binary input.

```
testbench.sv
1 //3_bit binary to gray (21EL026)
2 module testbench();
3   reg [2:0] cin;
4   wire [2:0] d;
5   bin_to_gray DUT(cin,d);
6
7   initial
8   begin
9     $monitor("cin = %b d = %b",cin,d);
10
11    cin = 3'b001; #10
12    cin = 3'b101; #10
13    cin = 3'b111; #10
14    cin = 3'b110; #10
15    $finish;
16  end
17 endmodule

design.sv
1 module bin_to_gray(input [2:0]cin,output [2:0]d);
2   assign d[0] = cin[1] ^ cin[0];
3   assign d[1] = cin[2] ^ cin[1];
4   assign d[2] = cin[2];
5 endmodule
```

OUTPUT:



The screenshot shows the output log of a Verilog simulation. It displays the values of 'cin' and 'd' for four different test cases. The output is as follows:

```
Log Share
[2022-08-07 16:23:16 EDT] iverilog '-Wall' design.sv testbench.sv && unbuffer vvp a.out
cin = 001 d = 001
cin = 101 d = 111
cin = 111 d = 100
cin = 110 d = 101
Done
```

Q-5: Write a Verilog code for BCD to excess 3 convertor.

CODE:

| testbench.sv  | design.sv  |
|---|--|
| <pre> 1 //BCD to excess 3 convertor (21EL026) 2 module testbench(); 3   reg a,b,c,d; 4   wire x,y,z,w; 5   bcd_to_excess3 DUT(a,b,c,d,x,y,z,w); 6 7   initial 8   begin 9     \$monitor("a = %b b = %b c = %b d = %b, x = %b 10    y = %b z = %b w = %b",a,b,c,d,x,y,z,w); 11 12    a = 1'b0; b = 1'b1; c = 1'b0; d = 1'b0;#50 13    a = 1'b1; b = 1'b1; c = 1'b0; d = 1'b0;#50 14    a = 1'b0; b = 1'b0; c = 1'b0; d = 1'b1;#50 15    a = 1'b1; b = 1'b0; c = 1'b0; d = 1'b1;#50 16    \$finish; 17  end 18 endmodule </pre> | <pre> 1 module bcd_to_excess3(input a,b,c,d,output x,y,z,w SV/Verilog Design 2   assign x = a   (b &amp; c)   b &amp; d; 3   assign y = (~b &amp; c)   (~b &amp; d)   (b &amp; ~c &amp; ~d); 4   assign z = (c &amp; d)   (~c &amp; ~d); 5   assign w = ~d; 6 endmodule </pre> |

OUTPUT:

| Log  | Share |
|--|-------|
| <pre> [2022-08-07 16:39:34 EDT] iverilog '-wall' design.sv testbench.sv &amp;&amp; unbuffer vvp a.out a = 0 b = 1 c = 0 d = 0, x = 0 y = 1 z = 1 w = 1 a = 1 b = 1 c = 0 d = 0, x = 1 y = 1 z = 1 w = 1 a = 0 b = 0 c = 0 d = 1, x = 0 y = 1 z = 0 w = 0 a = 1 b = 0 c = 0 d = 1, x = 1 y = 1 z = 0 w = 0 </pre> |       |
| Done   |       |