ASSIGNMENT - 1

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

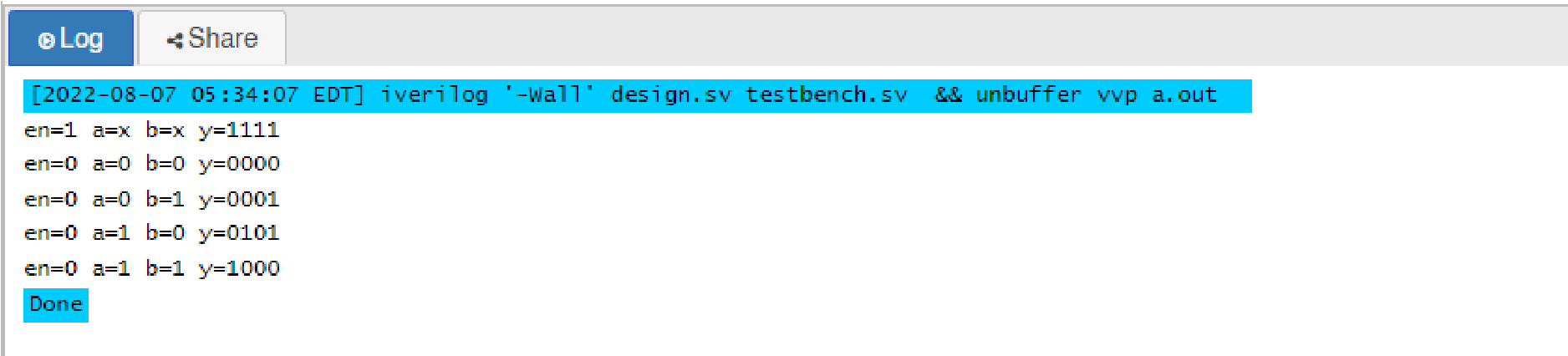
|  |
| --- |
| module decoder2\_4(input en,a,b,output reg [3:0]y);  always @(\*)  begin  if(en==0)  begin  if(a==1’b0 & b==1’b0) y=4’b0000;  else if(a==1’b0 & b==1’b1) y=4’b0001;  else if(a==1’b1 & b==1’b0) y=4’b0101;  else if(a==1 & b==1) y=4’b1000;  else y=4’bxxxx;  end  else  y=4’b1111;  end  endmodule |

CODE:

TESTBENCH:

|  |
| --- |
| module testbench;  reg a,b,en;  wire [3:0]y;  decoder2\_4 DUT(en,a,b,y);  initial  begin  $monitor(“en=%b a=%b b=%b y=%b”,en,a,b,y);  en=1;a=1’bx;b=1’bx;#5  en=0;a=0;b=0;#5  en=0;a=0;b=1;#5  en=0;a=1;b=0;#5  en=0;a=1;b=1;#5  $finish;  end  endmodule |

OUTPUT :



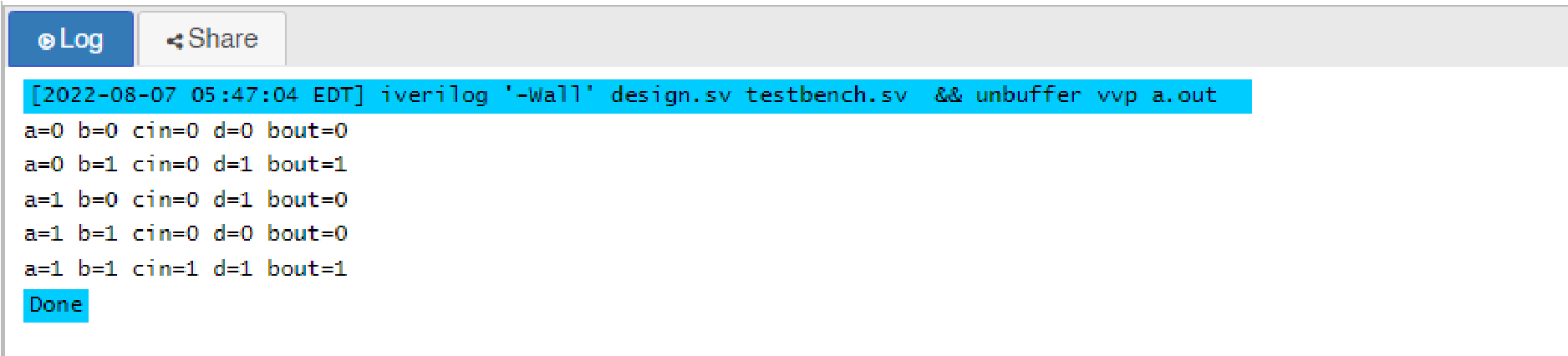
**Q-2: Write a Verilog code for Full subtractor.**

CODE :

|  |
| --- |
| module full\_subtractor(input a,b,cin, output d,bout);  assign d = a^b^cin ;  assign bout = ~a&b | ~(a^b)&cin;  endmodule |

TESTBENCH :

|  |
| --- |
| module testbench;  reg a,b,cin;  wire d,bout;  full\_subtractor DUT(a,b,cin,d,bout);  initial  begin  $monitor(“a=%b b=%b cin=%b d=%b bout=%b”,a,b,cin,d,bout);  cin=0;a=0;b=0;#5  cin=0;a=0;b=1;#5  cin=0;a=1;b=0;#5  cin=0;a=1;b=1;#5  cin=1;a=1;b=1;#5  $finish;  end  endmodule |



OUTPUT :

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

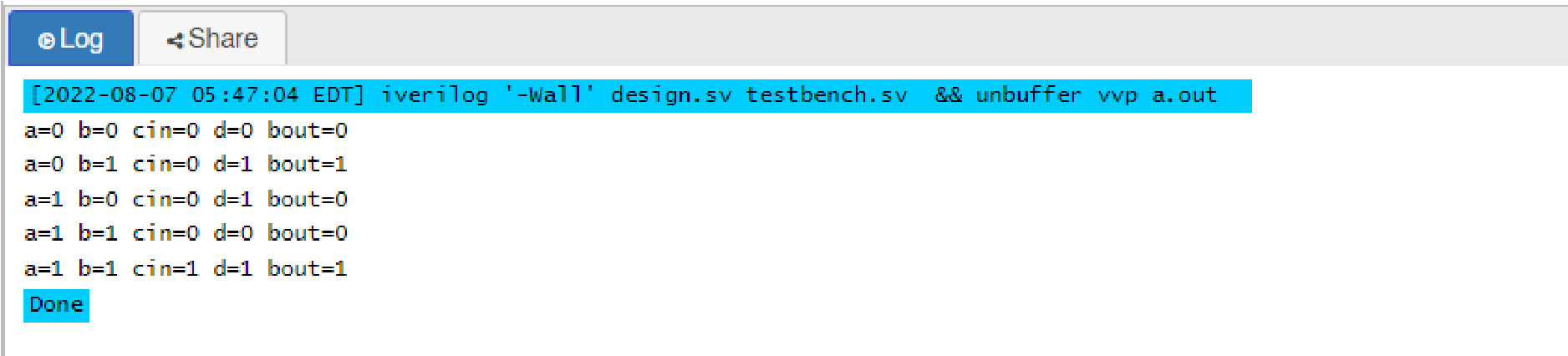
CODE :

|  |
| --- |
| module comparator\_2bit(a,b,equal,greater,lower);  output equal;  output greater ;  output lower ;  input [1:0] a;  input [1:0] b;  assign equal = (a==b) ? 1 : 0;  assign greater = (a>b) ? 1 : 0;  assign lower = (a<b) ? 1 : 0;  endmodule |

TESTBENCH:

|  |
| --- |
| module testbench;  reg [1:0]a,b;  wire equal,greater,lower;  comparator\_2bit DUT(a,b,equal,greater,lower);  initial  begin  $monitor(“a[0]=%b a[1]=%b b[0]=%b b[1]=%b greater =%b lower =%b equal=%b”,a[0],a[1],b[0],b[1],greater,lower,equal);    a=2’b00;b=2’b01;#5 a=2’b11;b=2’b10;#5 a=2’b00;b=2’b00;#5    $finish;  end  endmodule |

OUTPUT:

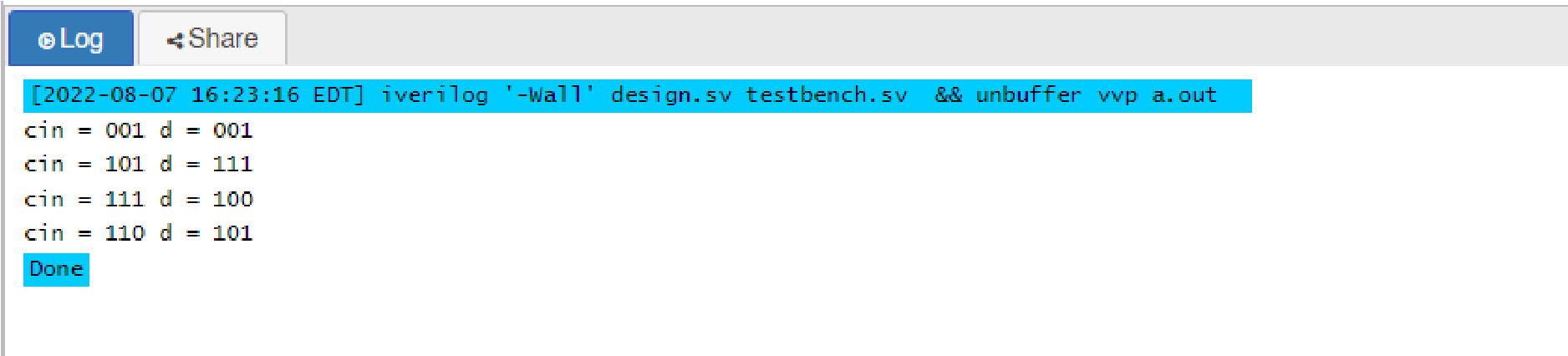


**Q-4: Write a Verilog code for 3 bit binary to gray convertor.**

CODE :

module bin\_to\_gray(input [2:0]cin,output [2:0]d);

OUTPUT:



assign d[0] = cin[1] ^ cin[0]; assign d[1] = cin[2] ^ cin[1]; assign d[2] = cin[2]; endmodule

TESTBENCH:

module testbench();

reg [2:0] cin;

wire [2:0] d;

bin\_to\_gray DUT(cin,d);

initial

begin

$monitor(“cin = %b d = %b”,cin,d);

cin = 3’b001; #10

cin = 3’b101; #10

cin = 3’b111; #10

cin = 3’b110; #10

$finish;

end

endmodule

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

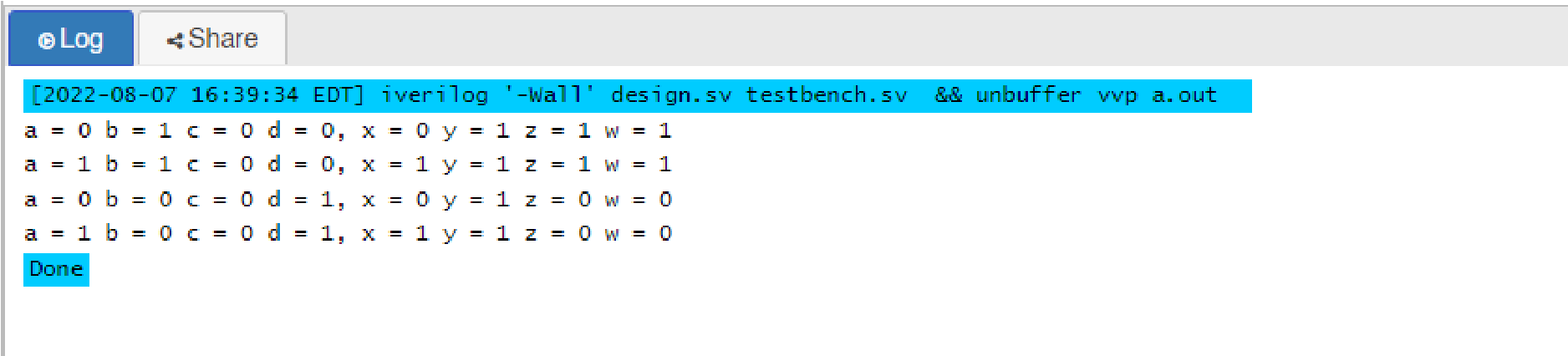
CODE :

module bcd\_to\_excess3(input a,b,c,d,output x,y,z,w);

assign x = a | (b & c) | b & d;

assign y = (~b & c) | (~b & d) | (b & ~c & ~d);assign z = (c & d) | (~c & ~d); assign w = ~d; endmodule

OUTPUT:



TESTBENCH:

module testbench();

reg a,b,c,d;

wire x,y,z,w;

bcd\_to\_excess3 DUT(a,b,c,d,x,y,z,w);

initial

begin

$monitor(“a = %b b = %b c = %b d = %b, x = %b

y = %b z = %b w = %b”,a,b,c,d,x,y,z,w);

a = 1’b0; b = 1’b1; c = 1’b0; d = 1’b0;#50

a = 1’b1; b = 1’b1; c = 1’b0; d = 1’b0;#50

a = 1’b0; b = 1’b0; c = 1’b0; d = 1’b1;#50

a = 1’b1; b = 1’b0; c = 1’b0; d = 1’b1;#50

$finish;

end

endmodule