Assignment 1

1. Write a Verilog code for 2X4 decoder.

design.sv

```
// Code your design here
module decoder24_behaviour(en,a,b,y);

   input en,a,b;
   output reg [3:0]y;

   always @(en,a,b)
     begin

        if(en==0)
          begin
            if(a==1'b0 & b==1'b0) y=4'b1110;
            else if(a==1'b0 & b==1'b1) y=4'b1101;
            else if(a==1'b1 & b==1'b0) y=4'b1011;
            else if(a==1 & b==1) y=4'b0111;
            else y=4'bxxxx;
          end
        else
         y=4'b1111;
     end
endmodule
```

testbench.sv

SV/Verilog

```
module tb;

  reg a,b,en;
  wire [3:0]y;

  initial
    begin
      $monitor("en=%b a=%b b=%b y=%b",en,a,b,y);
    end

    decoder24_behaviour dut(en,a,b,y);

   initial
    begin
    $dumpfile("dump.vcd");
    $dumpvars(1);

      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:

```
VCD info: dumpfile dump.vcd opened for output.
en=1 a=x b=x y=1111
en=0 a=0 b=0 y=1110
en=0 a=0 b=1 y=1101
en=0 a=1 b=0 y=1011
en=0 a=1 b=1 y=0111
Done
```

2. Write a Verilog code for Full subtractor.

design.sv

```
// Code your design here
module full_subtractor(input a, b, Bin, output D, Bout);
  assign D = a ^ b ^ Bin;
  assign Bout = (~a & b) | (~(a ^ b) & Bin);
endmodule
```

testbench.sv

SV/Verilog Testben

```
// Code your testbench here
// or browse Examples
module tb_top;
  reg a, b, Bin;
  wire D, Bout;

  full_subtractor fs(a, b, Bin, D, Bout);

initial begin
  $monitor("At time %0t: a=%b b=%b, Bin=%b, difference=%b, borrow=%b",$time, a,b,Bin,D,Bout);
    a = 0; b = 0; Bin = 0; #1;
    a = 0; b = 0; Bin = 1; #1;
    a = 0; b = 1; Bin = 0; #1;
    a = 0; b = 1; Bin = 1; #1;
    a = 1; b = 0; Bin = 0; #1;
    a = 1; b = 0; Bin = 1; #1;
    a = 1; b = 1; Bin = 0; #1;
    a = 1; b = 1; Bin = 1;
  end
endmodule
```

Output:

```
At time 0: a=0 b=0, Bin=0, difference=0, borrow=0
At time 1: a=0 b=0, Bin=1, difference=1, borrow=1
At time 2: a=0 b=1, Bin=0, difference=1, borrow=1
At time 3: a=0 b=1, Bin=1, difference=0, borrow=1
At time 4: a=1 b=0, Bin=0, difference=1, borrow=0
At time 5: a=1 b=0, Bin=1, difference=0, borrow=0
At time 6: a=1 b=1, Bin=0, difference=0, borrow=0
At time 7: a=1 b=1, Bin=1, difference=1, borrow=1
Done
```

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

design.sv

```
// Code your design here
module comparator_2bit(a,b,equal,lesser,greater);
  input [1:0]a;
  input [1:0]b;
  output equal,lesser,greater;

  assign greater = (a[1] & ~b[1]) | (a[1] ~^ b[1]) & (a[0] & ~b[0]);
  assign equal =  (a[1] ~^ b[1]) & (a[0] ~^ b[0]);
  assign lesser = (~a[1] & b[1]) | (a[1] ~^ b[1]) & (~a[0] & b[0]);

endmodule
```

testbench.sv

SV/Verilog Testbench

```
// Code your testbench here
// or browse Examples

module testbench;
  reg [1:0]a;
  reg [1:0]b;
  wire equal,lesser,greater;

  initial begin
    $monitor("a=%b,b=%b,equal=%b,greater=%b,lesser=%b",a,b,equal,greater,lesser);
  end

  comparator_2bit uut(a,b,equal,lesser,greater);

  initial begin

    #10 a=2'b01;b=2'b00;
    #10 a=2'b10;b=2'b10;
    #10 a=2'b11;b=2'b01;
    #10 a=2'b10;b=2'b11;

  end

  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0);
  end

endmodule
```

Output:

```
[2023-08-21 16:46:26 UTC] iverilog '-Wall' desig
VCD info: dumpfile dump.vcd opened for output.
a=xx,b=xx,equal=x,greater=x,lesser=x
a=01,b=00,equal=0,greater=1,lesser=0
a=10,b=10,equal=1,greater=0,lesser=0
a=11,b=01,equal=0,greater=1,lesser=0
a=10,b=11,equal=0,greater=0,lesser=1
Done
```

4. Write a Verilog code for 3 bit binary to grey convertor.

design.sv

```
// Code your design here
module Binary_to_Gray(
    input [3:0] b,
    output [3:0] g
    );
assign g[0]=b[1]^b[0];
assign g[1]=b[2]^b[1];
assign g[2]=b[3]^b[2];
assign g[3]=b[3];
endmodule
```

testbench.sv

```
// Code your testbench here
// or browse Examples
module Binary_to_Gray_tb;
       reg [3:0]b;
       wire [3:0]g;

       Binary_to_Gray uut (b,g);

       initial begin
         $monitor("b=%b,g=%b",b,g);
       end
   initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars;


       b=4'b0000;
       #10   b=4'b0001;
       #10   b=4'b0010;
       #10   b=4'b0011;
       #10   b=4'b0100;
       #10   b=4'b0101;
       #10   b=4'b0110;
       #10   b=4'b0111;
       #10   b=4'b1000;

       end
endmodule
```

Output:

```
VCD info: dumpfile dump.vcd opened for output.
b=0000,g=0000
b=0001,g=0001
b=0010,g=0011
b=0011,g=0010
b=0100,g=0110
b=0101,g=0111
b=0110,g=0101
b=0111,g=0100
b=1000,g=1100
Done
```

5. Write a Verilog code for BCD to excess 3 convertors.

design.sv

```
module Bcd_excess3(b,e);
input [3:0] b;
output [3:0] e;
assign e[3]=b[3]|b[2]&b[1]|b[2]&b[0];
assign e[2]=~b[2]&b[1]|~b[2]&b[0]|b[2]&~b[1]&~b[0];
assign e[1]=b[1]&b[0]|~b[1]&~b[0];
assign e[0]=~b[0];
endmodule
```

testbench.sv

```
module Bcd_excess3_tb;

reg [3:0] b;
wire [3:0] e;

Bcd_excess3 uut (
.b(b),
.e(e)
);

 initial begin

   $monitor("b = %b,e = %b", b,e);

 $dumpfile("dump.vcd");
 $dumpvars(0);
 end

initial begin

b=3'b0000;
#10;
b=4'b0001;
#10;
b=4'b0010;
#10;
b=4'b0011;
#10;
b=4'b0100;
#10;
b=4'b0101;
#10;
b=4'b0110;

end
endmodule
```

Output:

```
testbench.sv:21: warning: extra digits given fc
VCD info: dumpfile dump.vcd opened for output.
b = 0000,e = 0011
b = 0001,e = 0100
b = 0010,e = 0101
b = 0011,e = 0110
b = 0100,e = 0111
b = 0101,e = 1000
b = 0110,e = 1001
Done
```