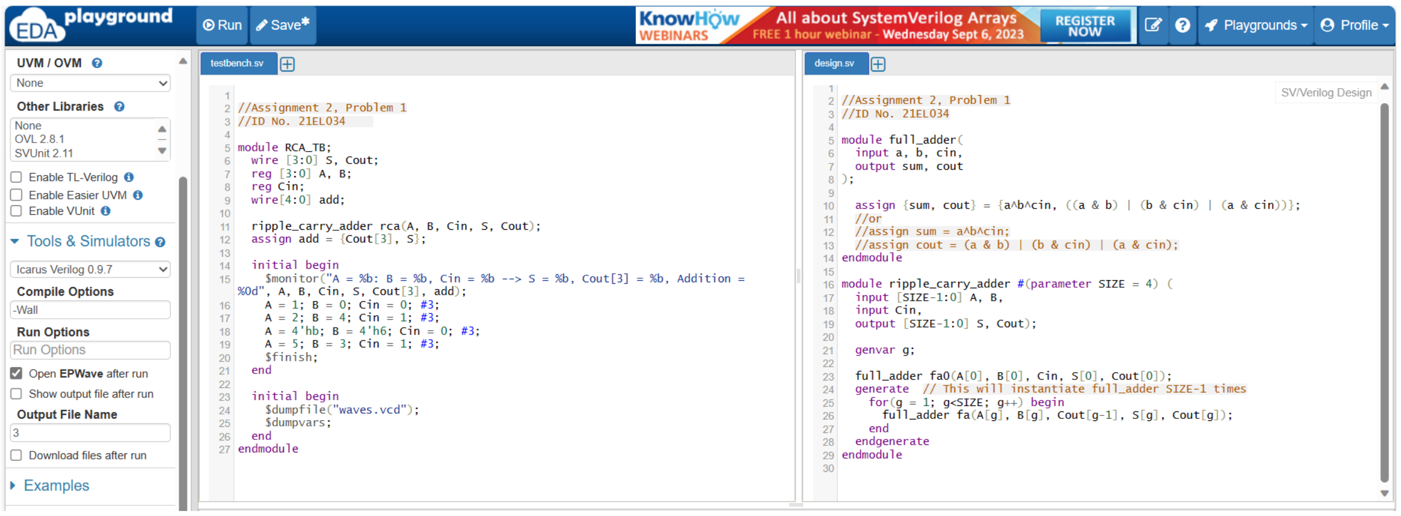
ID No. 21EL034

ASSIGNMENT-2

Q.1 Design 4-bit Ripple Carry Adder with the help of 1-bit adder



UVM / OVM: None
Tools & Simulators: Icarus Verilog 0.9.7
Compile Options: -Wall
[✓] Open EPWave after run
Output File Name: 3

testbench.sv

```verilog

//Assignment 2, Problem 1
//ID No. 21EL034

module RCA_TB;
  wire [3:0] S, Cout;
  reg [3:0] A, B;
  reg Cin;
  wire[4:0] add;

  ripple_carry_adder rca(A, B, Cin, S, Cout);
  assign add = {Cout[3], S};

  initial begin
    $monitor("A = %b: B = %b, Cin = %b --> S = %b, Cout[3] = %b, Addition = %0d", A, B, Cin, S, Cout[3], add);
    A = 1; B = 0; Cin = 0; #3;
    A = 2; B = 4; Cin = 1; #3;
    A = 4'hb; B = 4'h6; Cin = 0; #3;
    A = 5; B = 3; Cin = 1; #3;
    $finish;
  end

  initial begin
    $dumpfile("waves.vcd");
    $dumpvars;
  end
endmodule
```

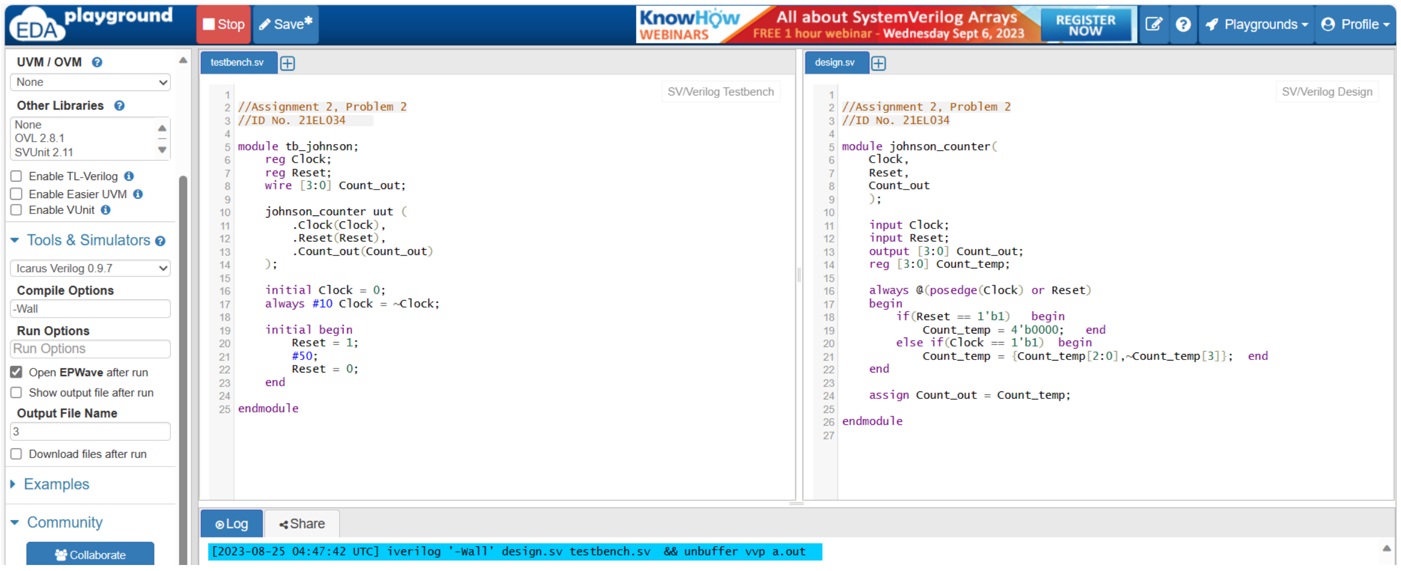
design.sv

```verilog

//Assignment 2, Problem 1
//ID No. 21EL034

module full_adder(
  input a, b, cin,
  output sum, cout
);

  assign {sum, cout} = {a^b^cin, ((a & b) | (b & cin) | (a & cin))};
  //or
  //assign sum = a^b^cin;
  //assign cout = (a & b) | (b & cin) | (a & cin);
endmodule

module ripple_carry_adder #(parameter SIZE = 4) (
  input [SIZE-1:0] A, B,
  input Cin,
  output [SIZE-1:0] S, Cout);

  genvar g;

  full_adder fa0(A[0], B[0], Cin, S[0], Cout[0]);
  generate  // This will instantiate full_adder SIZE-1 times
    for(g = 1; g<SIZE; g++) begin
      full_adder fa(A[g], B[g], Cout[g-1], S[g], Cout[g]);
    end
  endgenerate
endmodule

```

OUTPUT:

```
A = 0001: B = 0000, Cin = 0 --> S = 0001, Cout[3] = 0, Addition = 1
A = 0010: B = 0100, Cin = 1 --> S = 0111, Cout[3] = 0, Addition = 7
A = 1011: B = 0110, Cin = 0 --> S = 0001, Cout[3] = 1, Addition = 17
A = 0101: B = 0011, Cin = 1 --> S = 1001, Cout[3] = 0, Addition = 9
```

Q.2 Design D-flipflop and reuse it to implement 4- bit Johnson Counter



UVM / OVM: None
Tools & Simulators: Icarus Verilog 0.9.7
Compile Options: -Wall
[✓] Open EPWave after run
Output File Name: 3

testbench.sv

```verilog

//Assignment 2, Problem 2
//ID No. 21EL034

module tb_johnson;
    reg Clock;
    reg Reset;
    wire [3:0] Count_out;

    johnson_counter uut (
        .Clock(Clock),
        .Reset(Reset),
        .Count_out(Count_out)
    );

    initial Clock = 0;
    always #10 Clock = ~Clock;

    initial begin
        Reset = 1;
        #50;
        Reset = 0;
    end

endmodule
```

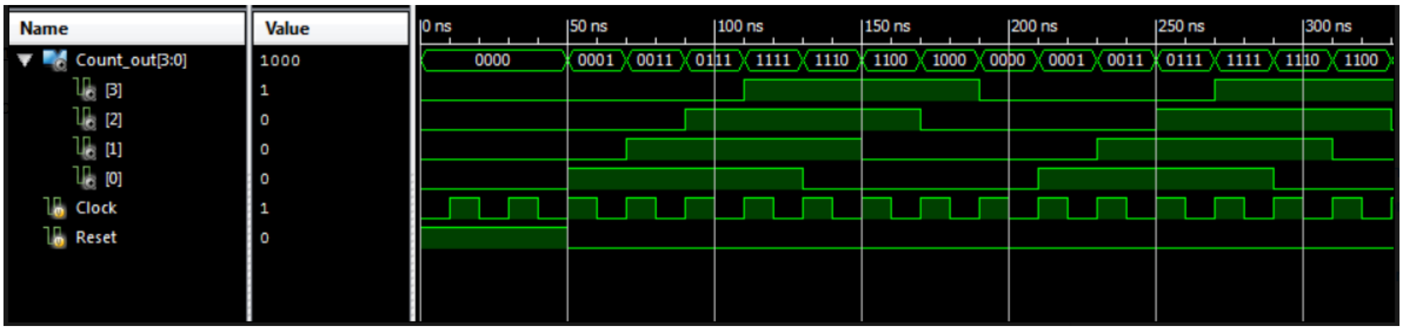
design.sv

```verilog

//Assignment 2, Problem 2
//ID No. 21EL034

module johnson_counter(
    Clock,
    Reset,
    Count_out
    );

    input Clock;
    input Reset;
    output [3:0] Count_out;
    reg [3:0] Count_temp;

    always @(posedge(Clock) or Reset)
    begin
        if(Reset == 1'b1)   begin
            Count_temp = 4'b0000;   end
        else if(Clock == 1'b1)  begin
            Count_temp = {Count_temp[2:0],~Count_temp[3]};  end
    end

    assign Count_out = Count_temp;

endmodule

```

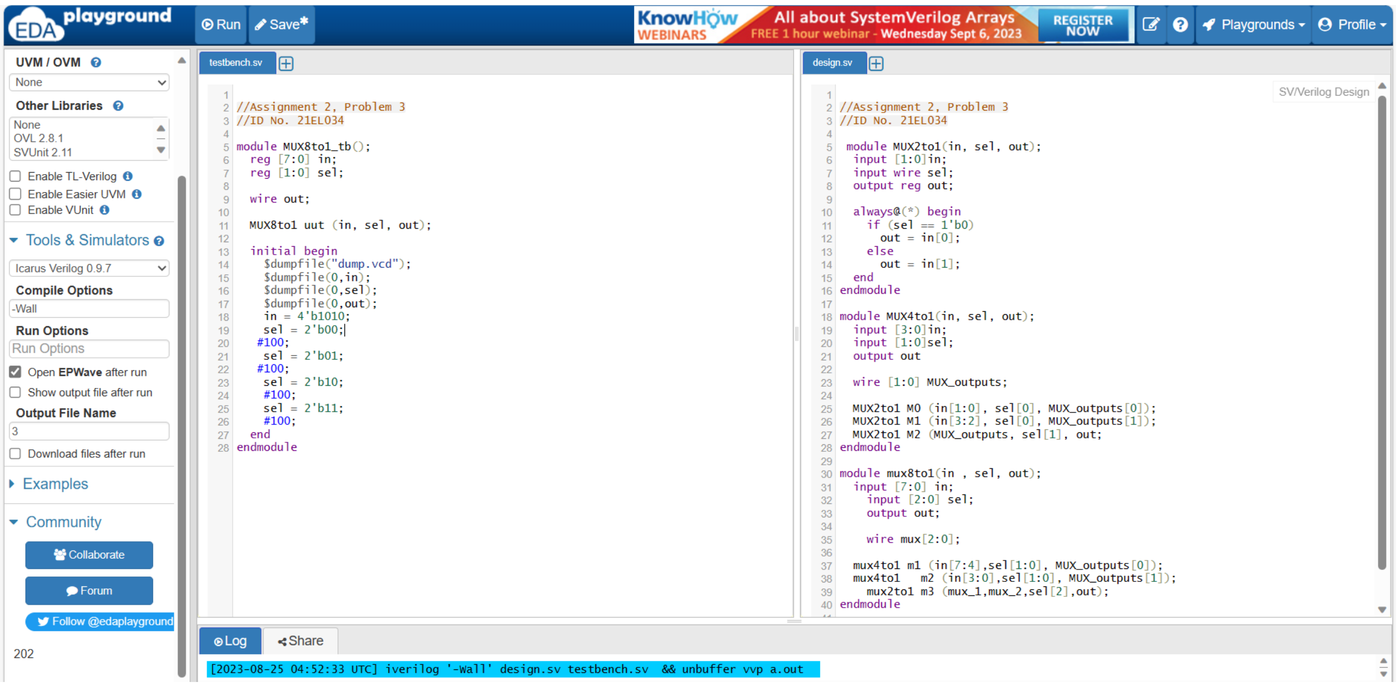
```
[2023-08-25 04:47:42 UTC] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
```

ID No. 21EL034

OUTPUT:



Q.3 Reuse 2:1 Mux code to implement 8:1 Mux



testbench.sv

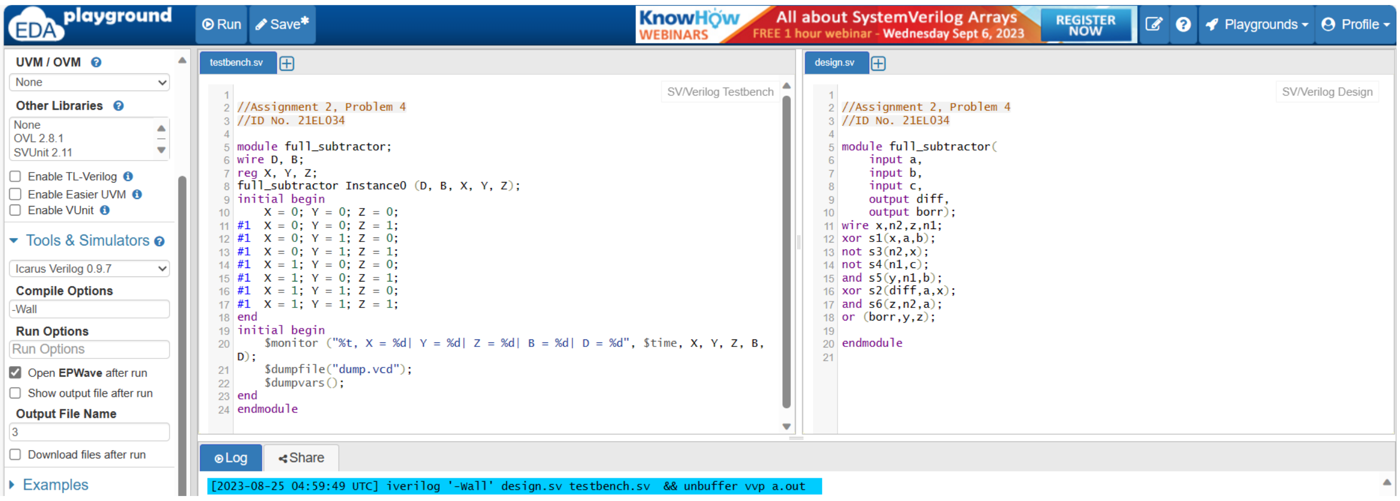
```verilog

//Assignment 2, Problem 3
//ID No. 21EL034

module MUX8to1_tb();
  reg [7:0] in;
  reg [1:0] sel;

  wire out;

  MUX8to1 uut (in, sel, out);

  initial begin
    $dumpfile("dump.vcd");
    $dumpfile(0,in);
    $dumpfile(0,sel);
    $dumpfile(0,out);
    in = 4'b1010;
    sel = 2'b00;
   #100;
    sel = 2'b01;
   #100;
    sel = 2'b10;
    #100;
    sel = 2'b11;
    #100;
  end
endmodule
```

design.sv

```verilog

//Assignment 2, Problem 3
//ID No. 21EL034

 module MUX2to1(in, sel, out);
  input [1:0]in;
  input wire sel;
  output reg out;

  always@(*) begin
    if (sel == 1'b0)
      out = in[0];
    else
      out = in[1];
  end
endmodule

module MUX4to1(in, sel, out);
  input [3:0]in;
  input [1:0]sel;
  output out

  wire [1:0] MUX_outputs;

  MUX2to1 M0 (in[1:0], sel[0], MUX_outputs[0]);
  MUX2to1 M1 (in[3:2], sel[0], MUX_outputs[1]);
  MUX2to1 M2 (MUX_outputs, sel[1], out;
endmodule

module mux8to1(in , sel, out);
  input [7:0] in;
    input [2:0] sel;
    output out;

    wire mux[2:0];

  mux4to1 m1 (in[7:4],sel[1:0], MUX_outputs[0]);
  mux4to1   m2 (in[3:0],sel[1:0], MUX_outputs[1]);
    mux2to1 m3 (mux_1,mux_2,sel[2],out);
endmodule
```

```
[2023-08-25 04:52:33 UTC] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
```

Q.4 Design a Full Subtractor with Gate Level Modelling Style(use primitive gates)



testbench.sv

```verilog

//Assignment 2, Problem 4
//ID No. 21EL034

module full_subtractor;
wire D, B;
reg X, Y, Z;
full_subtractor Instance0 (D, B, X, Y, Z);
initial begin
    X = 0; Y = 0; Z = 0;
#1  X = 0; Y = 0; Z = 1;
#1  X = 0; Y = 1; Z = 0;
#1  X = 0; Y = 1; Z = 1;
#1  X = 1; Y = 0; Z = 0;
#1  X = 1; Y = 0; Z = 1;
#1  X = 1; Y = 1; Z = 0;
#1  X = 1; Y = 1; Z = 1;
end
initial begin
    $monitor ("%t, X = %d| Y = %d| Z = %d| B = %d| D = %d", $time, X, Y, Z, B,
D);
    $dumpfile("dump.vcd");
    $dumpvars();
end
endmodule
```

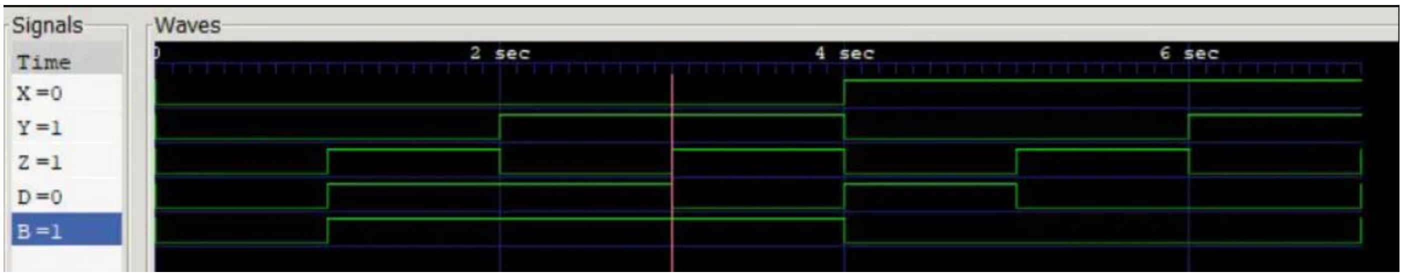
design.sv

```verilog

//Assignment 2, Problem 4
//ID No. 21EL034

module full_subtractor(
    input a,
    input b,
    input c,
    output diff,
    output borr);
wire x,n2,z,n1;
xor s1(x,a,b);
not s3(n2,x);
not s4(n1,c);
and s5(y,n1,b);
xor s2(diff,a,x);
and s6(z,n2,a);
or (borr,y,z);

endmodule

```

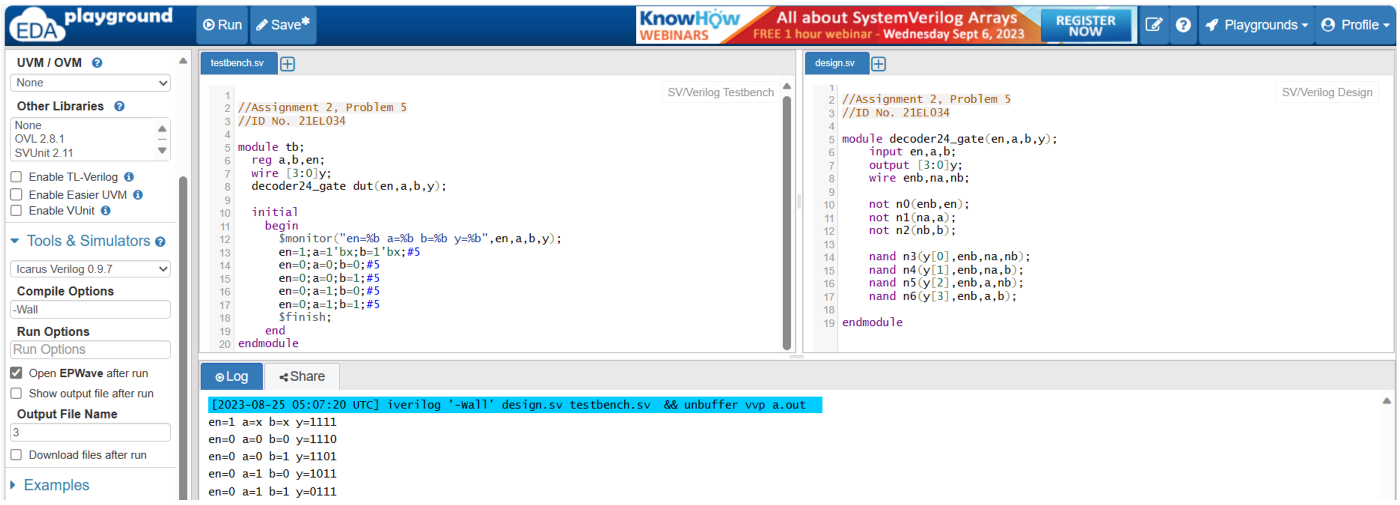
```
[2023-08-25 04:59:49 UTC] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
```

ID No. 21EL034

OUTPUT:



Q.5 Design a 2X4 decoder using gate level modelling



testbench.sv

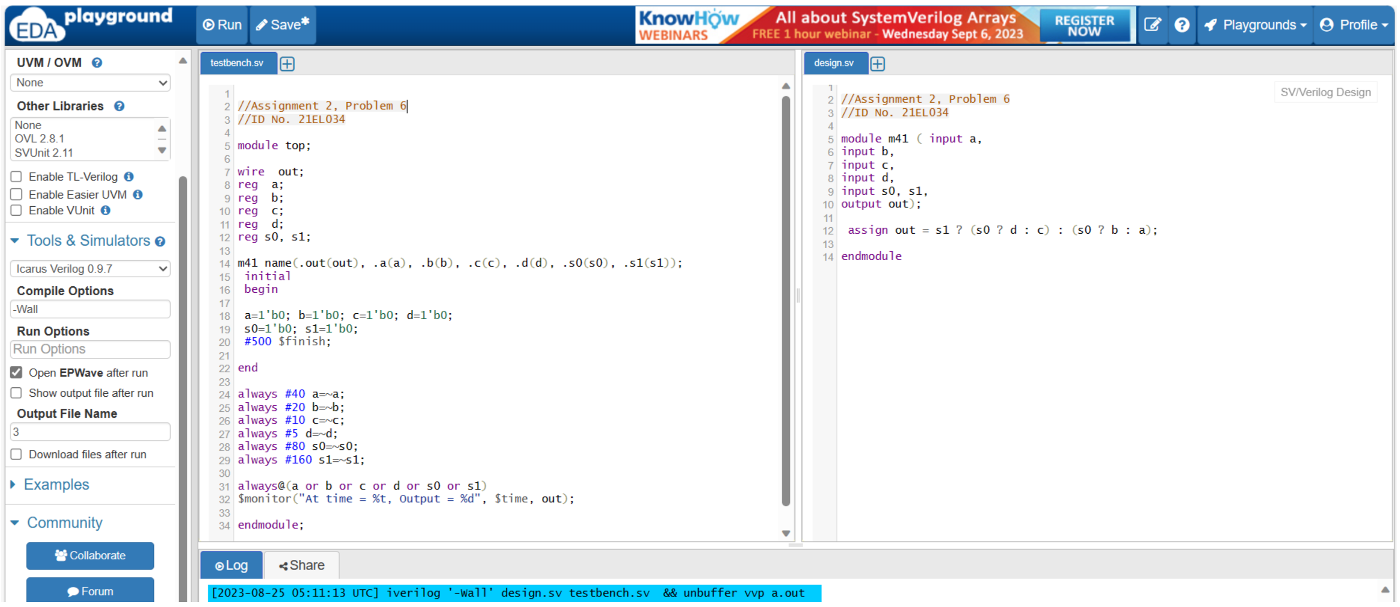
```
//Assignment 2, Problem 5
//ID No. 21EL034

module tb;
  reg a,b,en;
  wire [3:0]y;
  decoder24_gate 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
```

design.sv

```
//Assignment 2, Problem 5
//ID No. 21EL034

module decoder24_gate(en,a,b,y);
    input en,a,b;
    output [3:0]y;
    wire enb,na,nb;

    not n0(enb,en);
    not n1(na,a);
    not n2(nb,b);

    nand n3(y[0],enb,na,nb);
    nand n4(y[1],enb,na,b);
    nand n5(y[2],enb,a,nb);
    nand n6(y[3],enb,a,b);

endmodule
```

```
[2023-08-25 05:07:20 UTC] 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=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
```

Q.6 Design a 4x1 mux using operators(use data flow)



testbench.sv

```
//Assignment 2, Problem 6
//ID No. 21EL034

module top;

wire  out;
reg  a;
reg  b;
reg  c;
reg  d;
reg s0, s1;

m41 name(.out(out), .a(a), .b(b), .c(c), .d(d), .s0(s0), .s1(s1));
 initial
 begin

 a=1'b0; b=1'b0; c=1'b0; d=1'b0;
 s0=1'b0; s1=1'b0;
 #500 $finish;

end

always #40 a=~a;
always #20 b=~b;
always #10 c=~c;
always #5 d=~d;
always #80 s0=~s0;
always #160 s1=~s1;

always@(a or b or c or d or s0 or s1)
$monitor("At time = %t, Output = %d", $time, out);

endmodule;
```

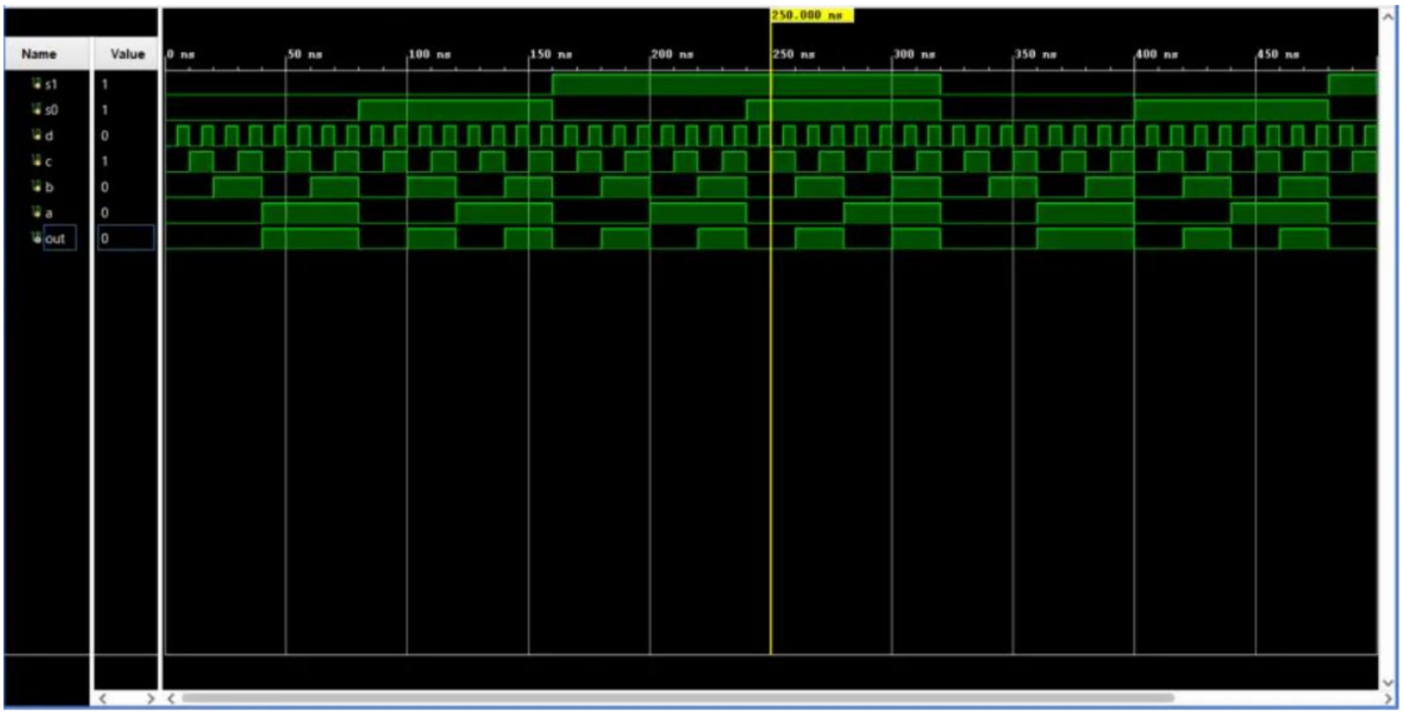
design.sv

```
//Assignment 2, Problem 6
//ID No. 21EL034

module m41 ( input a,
input b,
input c,
input d,
input s0, s1,
output out);

 assign out = s1 ? (s0 ? d : c) : (s0 ? b : a);

endmodule
```

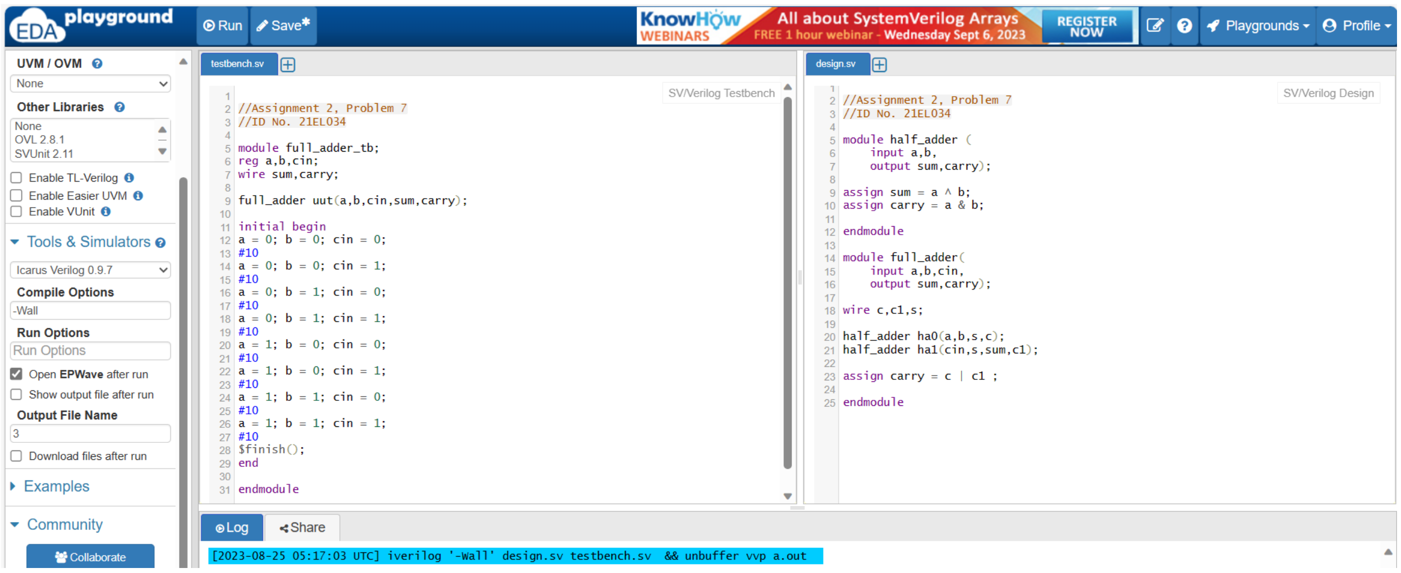
```
[2023-08-25 05:11:13 UTC] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
```

ID No. 21EL034

OUTPUT:



Q.7 Design a Full adder using half adder



testbench.sv

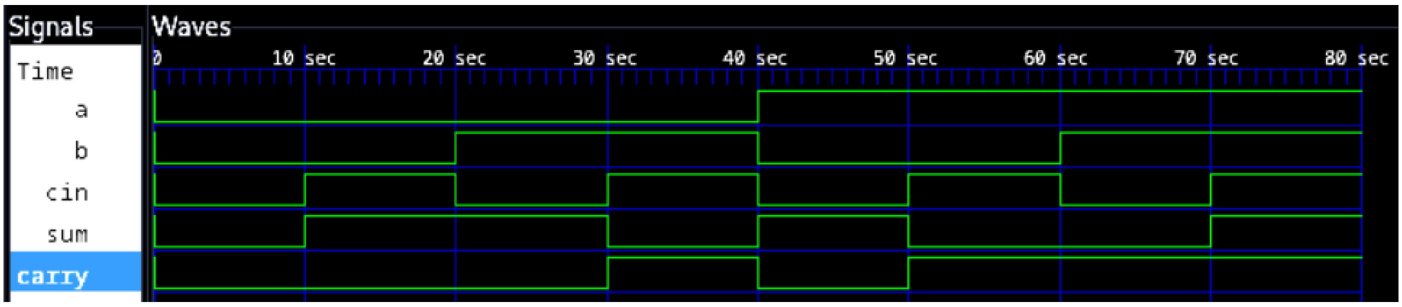
```
//Assignment 2, Problem 7
//ID No. 21EL034

module full_adder_tb;
reg a,b,cin;
wire sum,carry;

full_adder uut(a,b,cin,sum,carry);

initial begin
a = 0; b = 0; cin = 0;
#10
a = 0; b = 0; cin = 1;
#10
a = 0; b = 1; cin = 0;
#10
a = 0; b = 1; cin = 1;
#10
a = 1; b = 0; cin = 0;
#10
a = 1; b = 0; cin = 1;
#10
a = 1; b = 1; cin = 0;
#10
a = 1; b = 1; cin = 1;
#10
$finish();
end

endmodule
```

design.sv

```
//Assignment 2, Problem 7
//ID No. 21EL034

module half_adder (
    input a,b,
    output sum,carry);

assign sum = a ^ b;
assign carry = a & b;

endmodule

module full_adder(
    input a,b,cin,
    output sum,carry);

wire c,c1,s;

half_adder ha0(a,b,s,c);
half_adder ha1(cin,s,sum,c1);

assign carry = c | c1 ;

endmodule
```

```
[2023-08-25 05:17:03 UTC] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
```

OUTPUT:


Signals
Waves
Time
a
b
cin
sum
carry
10 sec
20 sec
30 sec
40 sec
50 sec
60 sec
70 sec
80 sec