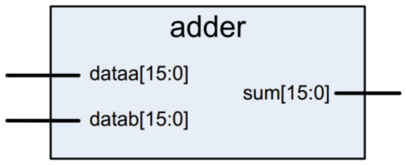
Group 1

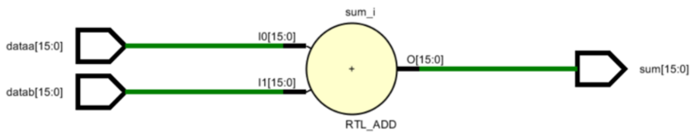
Mohamed Mohamed tarek   Mohamed salah   Abdalla Elgohary   waleed Emad

Sequential 8x8 multiplier

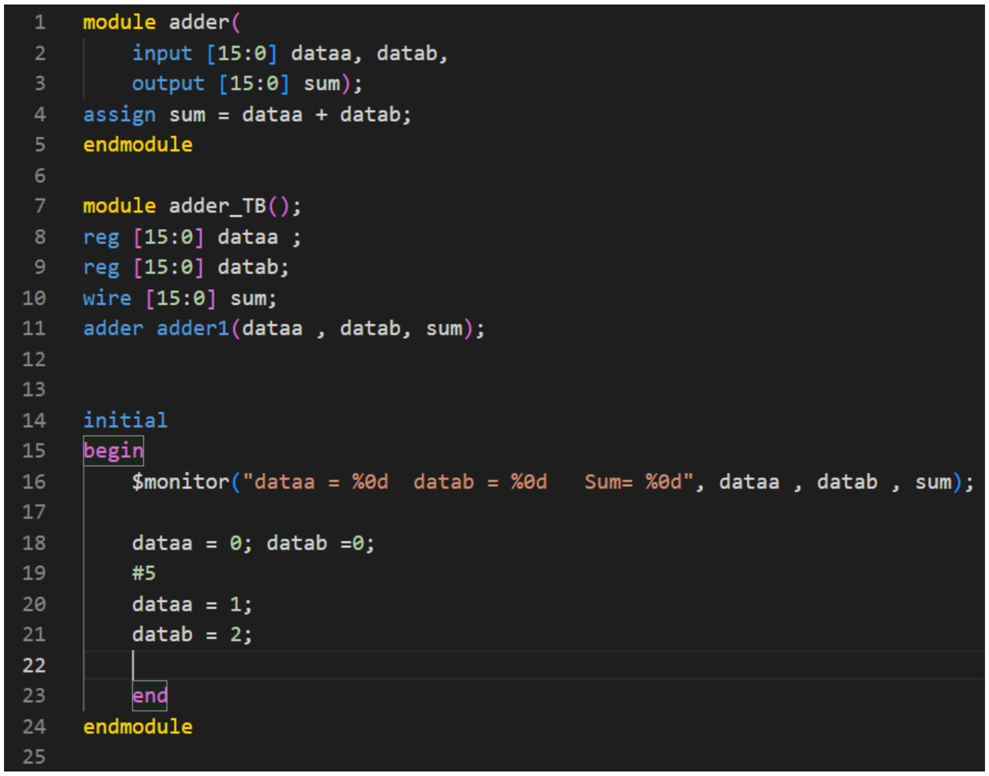
Build an 8x8 multiplier. The input to the multiplier consists of two 8-bit multiplicands (dataa and datab) and the output from the multiplier is 16-bit product (product8x8_out). Additional outputs are a done bit (done_flag) and seven signals to drive a 7 segment display (seg_a, seg_b, seg_c, seg_d, seg_e, seg_f & seg_g).

1- 16-bit adder :

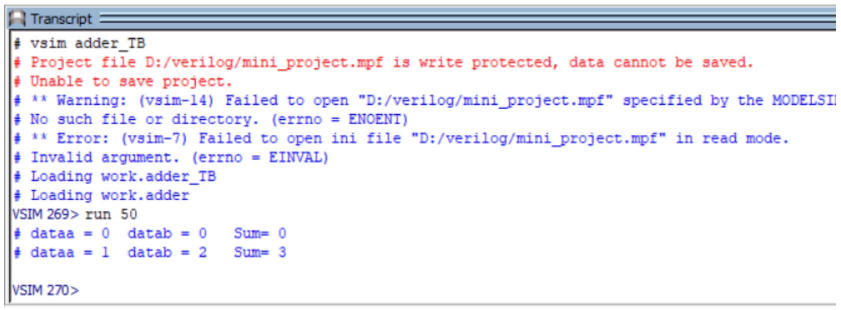




Verilog code to perform addition ands testbench :



```verilog
module adder(
    input [15:0] dataa, datab,
    output [15:0] sum);
assign sum = dataa + datab;
endmodule

module adder_TB();
reg [15:0] dataa ;
reg [15:0] datab;
wire [15:0] sum;
adder adder1(dataa , datab, sum);


initial
begin
    $monitor("dataa = %0d  datab = %0d   Sum= %0d", dataa , datab , sum);

    dataa = 0; datab =0;
    #5
    dataa = 1;
    datab = 2;

    end
endmodule
```



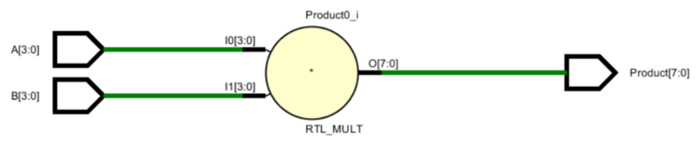
Transcript

```
# vsim adder_TB
# Project file D:/verilog/mini_project.mpf is write protected, data cannot be saved.
# Unable to save project.
# ** Warning: (vsim-14) Failed to open "D:/verilog/mini_project.mpf" specified by the MODELSI
# No such file or directory. (errno = ENOENT)
# ** Error: (vsim-7) Failed to open ini file "D:/verilog/mini_project.mpf" in read mode.
# Invalid argument. (errno = EINVAL)
# Loading work.adder_TB
# Loading work.adder
VSIM 269> run 50
# dataa = 0  datab = 0   Sum= 0
# dataa = 1  datab = 2   Sum= 3

VSIM 270>
```

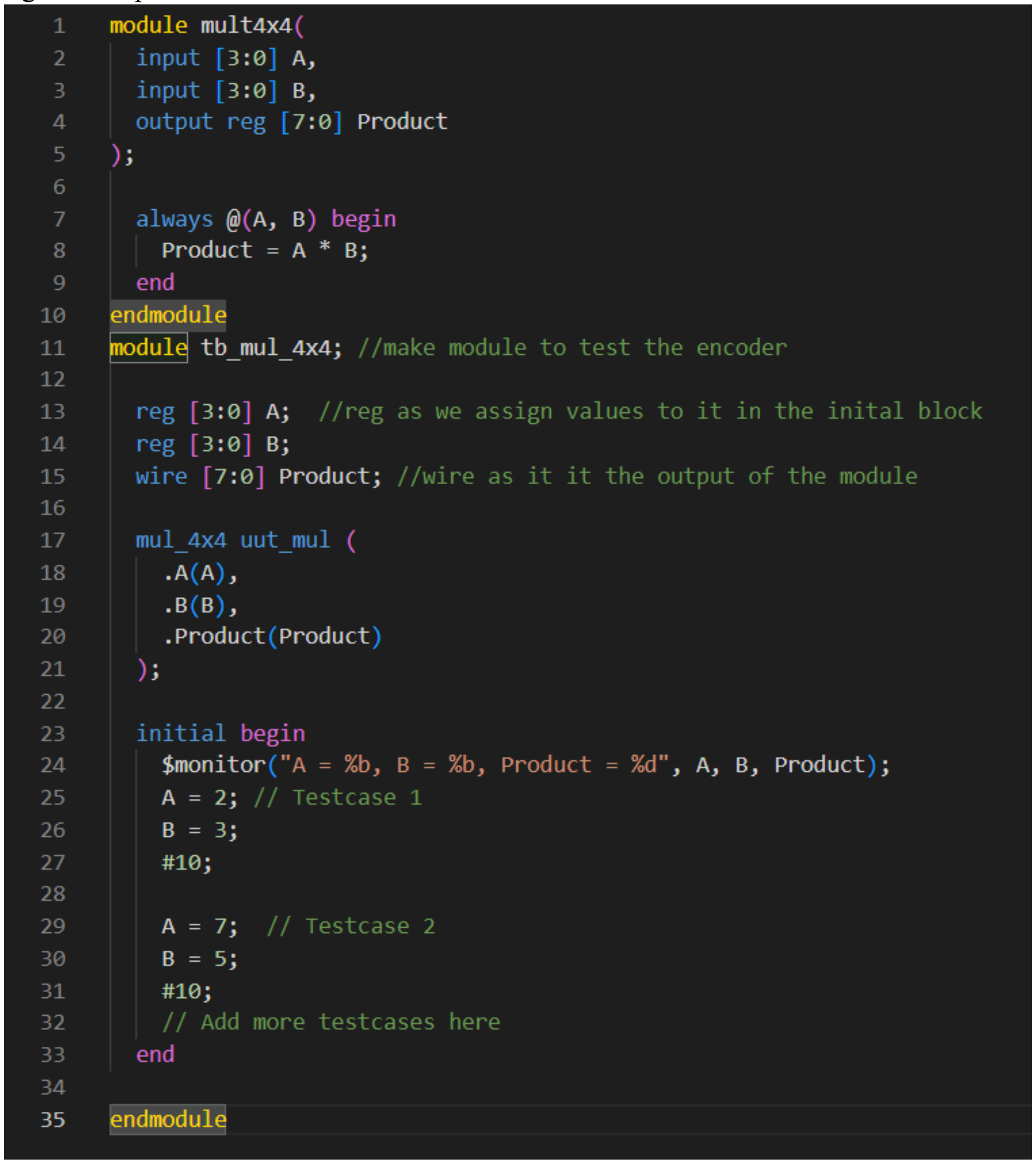
Group 1

Mohamed Mohamed tarek Mohamed salah Abdalla Elgohary waleed Emad

2- 4x4 multiplier :



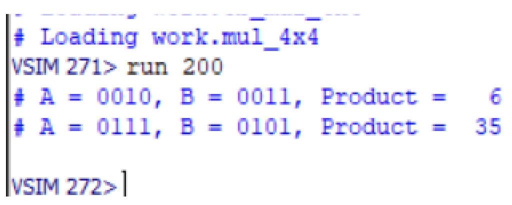
Verilog code to perform addition ands testbench :



```
module mult4x4(
  input [3:0] A,
  input [3:0] B,
  output reg [7:0] Product
);

  always @(A, B) begin
    Product = A * B;
  end
endmodule
module tb_mul_4x4; //make module to test the encoder

  reg [3:0] A;  //reg as we assign values to it in the inital block
  reg [3:0] B;
  wire [7:0] Product; //wire as it it the output of the module

  mul_4x4 uut_mul (
    .A(A),
    .B(B),
    .Product(Product)
  );

  initial begin
    $monitor("A = %b, B = %b, Product = %d", A, B, Product);
    A = 2; // Testcase 1
    B = 3;
    #10;

    A = 7;  // Testcase 2
    B = 5;
    #10;
    // Add more testcases here
  end

endmodule
```

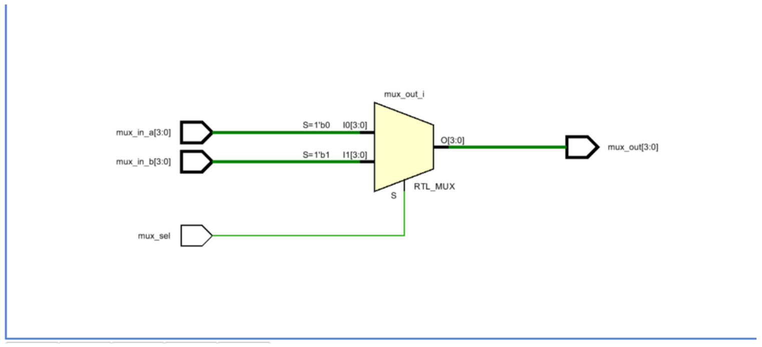
Group 1

Mohamed Mohamed tarek   Mohamed salah   Abdalla Elgohary   waleed Emad

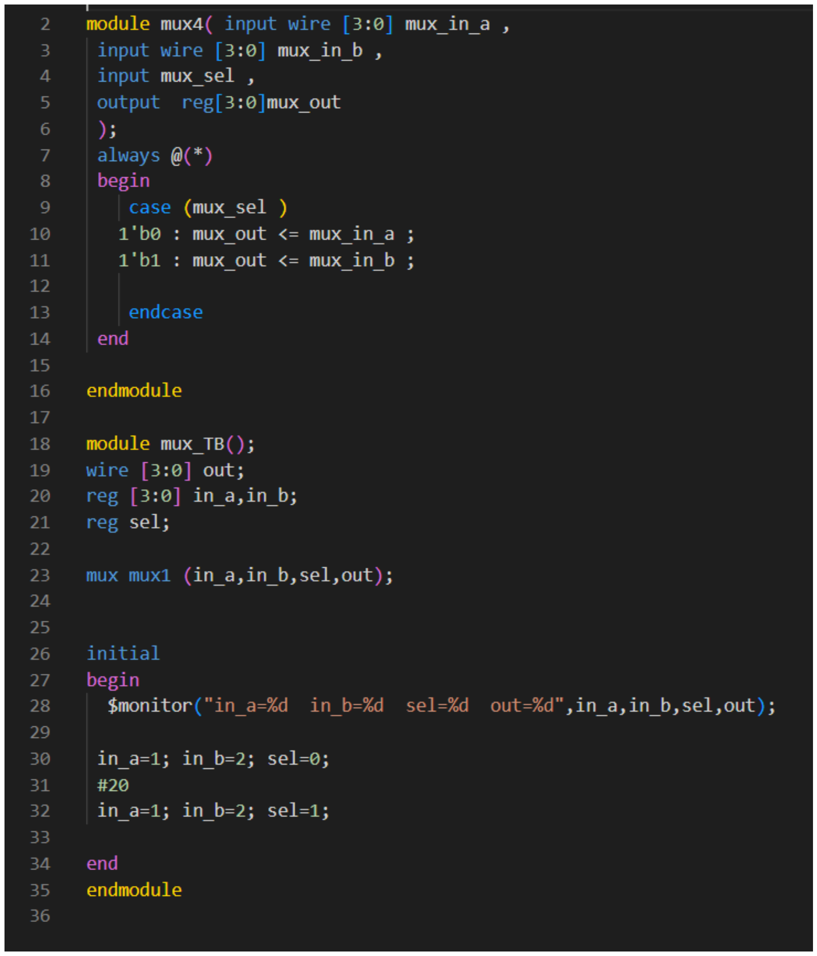


```
# Loading work.mul_4x4
VSIM 271> run 200
# A = 0010, B = 0011, Product =   6
# A = 0111, B = 0101, Product =  35

VSIM 272>
```

3-1 multiplexer:



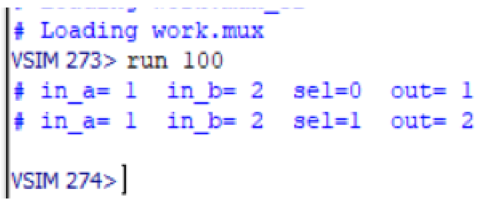
Verilog code to perform addition ands testbench :



```
module mux4( input wire [3:0] mux_in_a ,
 input wire [3:0] mux_in_b ,
 input mux_sel ,
 output  reg[3:0]mux_out
 );
 always @(*)
 begin
    case (mux_sel )
   1'b0 : mux_out <= mux_in_a ;
   1'b1 : mux_out <= mux_in_b ;

    endcase
 end

endmodule

module mux_TB();
wire [3:0] out;
reg [3:0] in_a,in_b;
reg sel;

mux mux1 (in_a,in_b,sel,out);


initial
begin
  $monitor("in_a=%d  in_b=%d  sel=%d  out=%d",in_a,in_b,sel,out);

 in_a=1; in_b=2; sel=0;
 #20
 in_a=1; in_b=2; sel=1;

end
endmodule
```

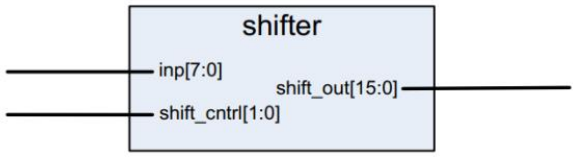
Group 1

Mohamed Mohamed tarek   Mohamed salah   Abdalla Elgohary   waleed Emad

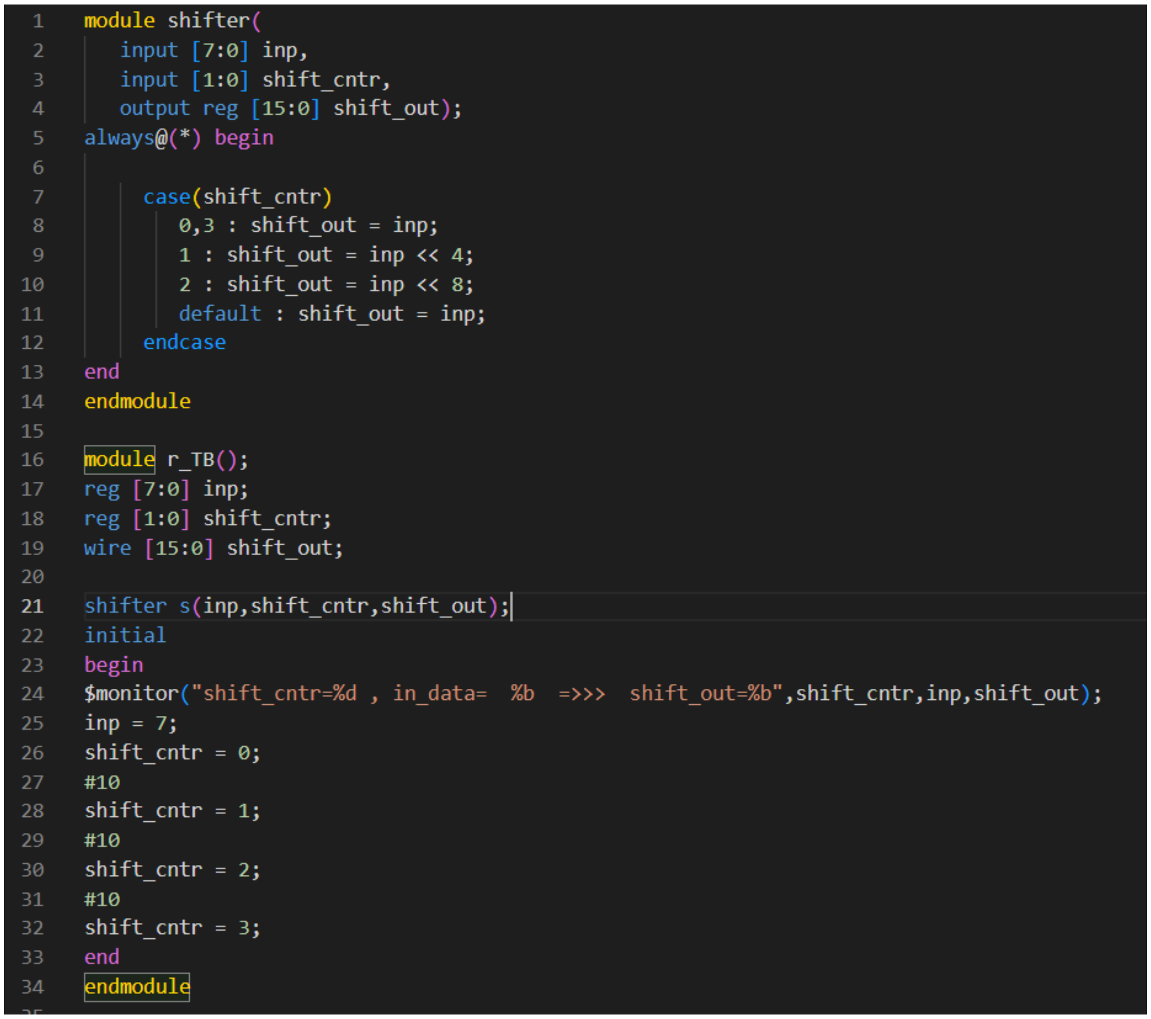


```
# Loading work.mux
VSIM 273> run 100
# in_a= 1  in_b= 2  sel=0  out= 1
# in_a= 1  in_b= 2  sel=1  out= 2

VSIM 274>
```

4-Shifter



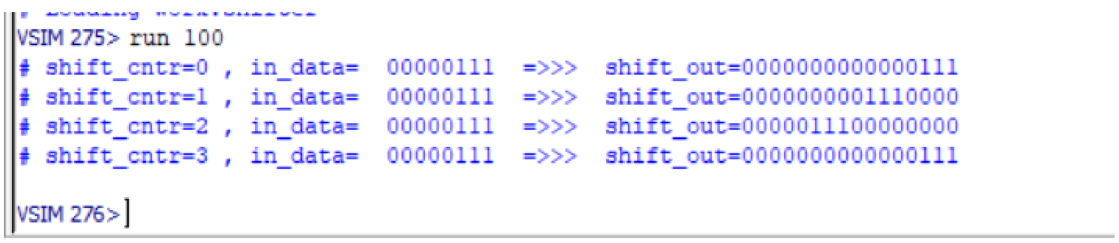
Verilog code to perform addition ands testbench :



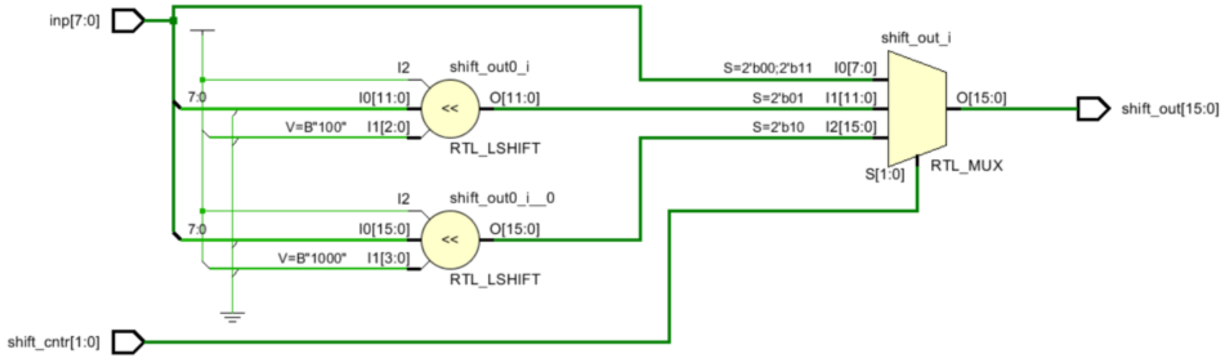
```
module shifter(
    input [7:0] inp,
    input [1:0] shift_cntr,
    output reg [15:0] shift_out);
always@(*) begin

     case(shift_cntr)
        0,3 : shift_out = inp;
        1 : shift_out = inp << 4;
        2 : shift_out = inp << 8;
        default : shift_out = inp;
     endcase
end
endmodule

module r_TB();
reg [7:0] inp;
reg [1:0] shift_cntr;
wire [15:0] shift_out;

shifter s(inp,shift_cntr,shift_out);
initial
begin
$monitor("shift_cntr=%d , in_data=  %b  =>>>  shift_out=%b",shift_cntr,inp,shift_out);
inp = 7;
shift_cntr = 0;
#10
shift_cntr = 1;
#10
shift_cntr = 2;
#10
shift_cntr = 3;
end
endmodule
```

Group 1

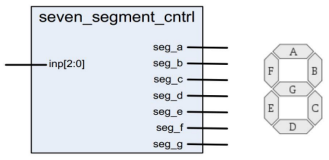
Mohamed Mohamed tarek   Mohamed salah   Abdalla Elgohary   waleed Emad



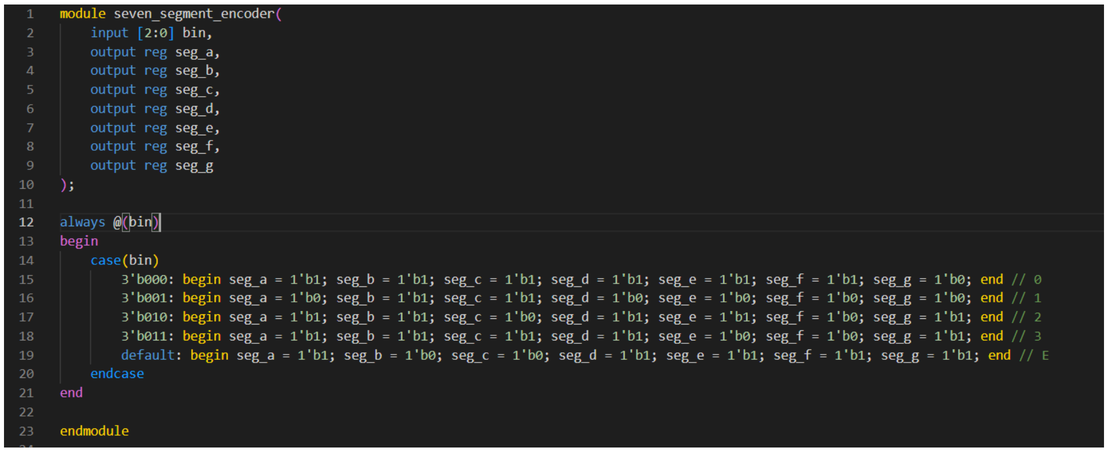
```
VSIM 275> run 100
# shift_cntr=0 , in_data=  00000111  =>>>  shift_out=0000000000000111
# shift_cntr=1 , in_data=  00000111  =>>>  shift_out=0000000001110000
# shift_cntr=2 , in_data=  00000111  =>>>  shift_out=0000011100000000
# shift_cntr=3 , in_data=  00000111  =>>>  shift_out=0000000000000111

VSIM 276>
```



7-segment display encoder:



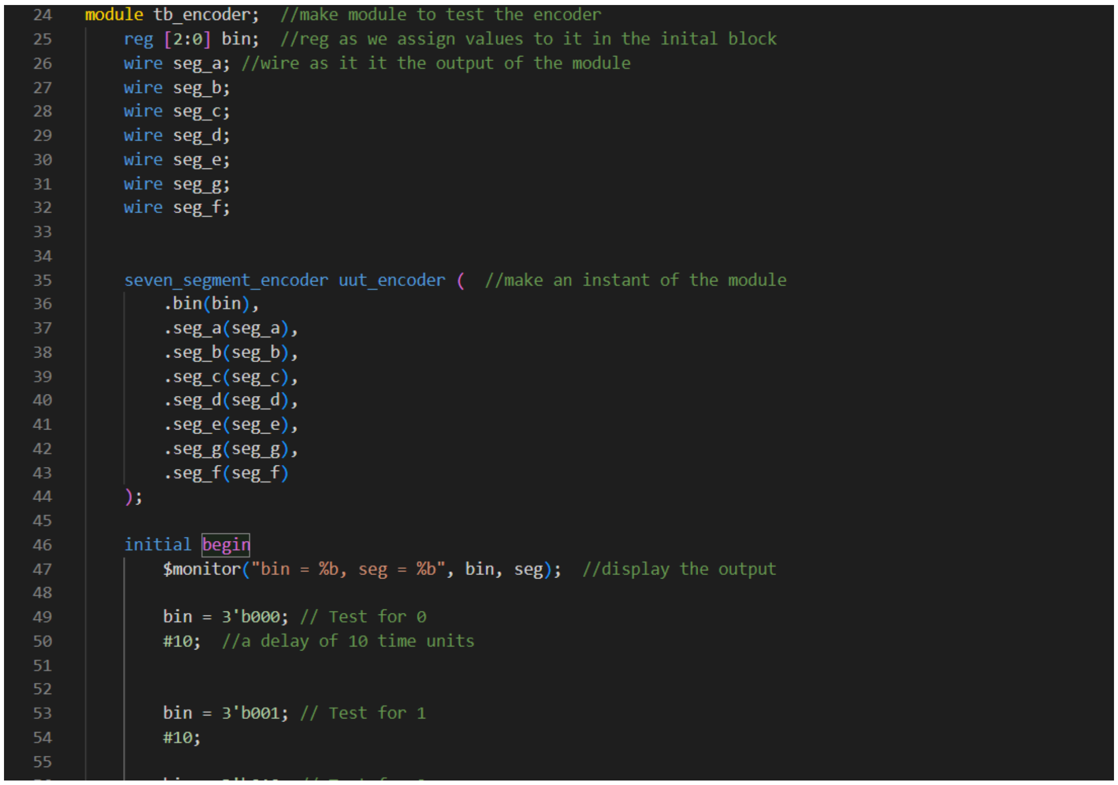
- Write Verilog code to perform encoder as following table:



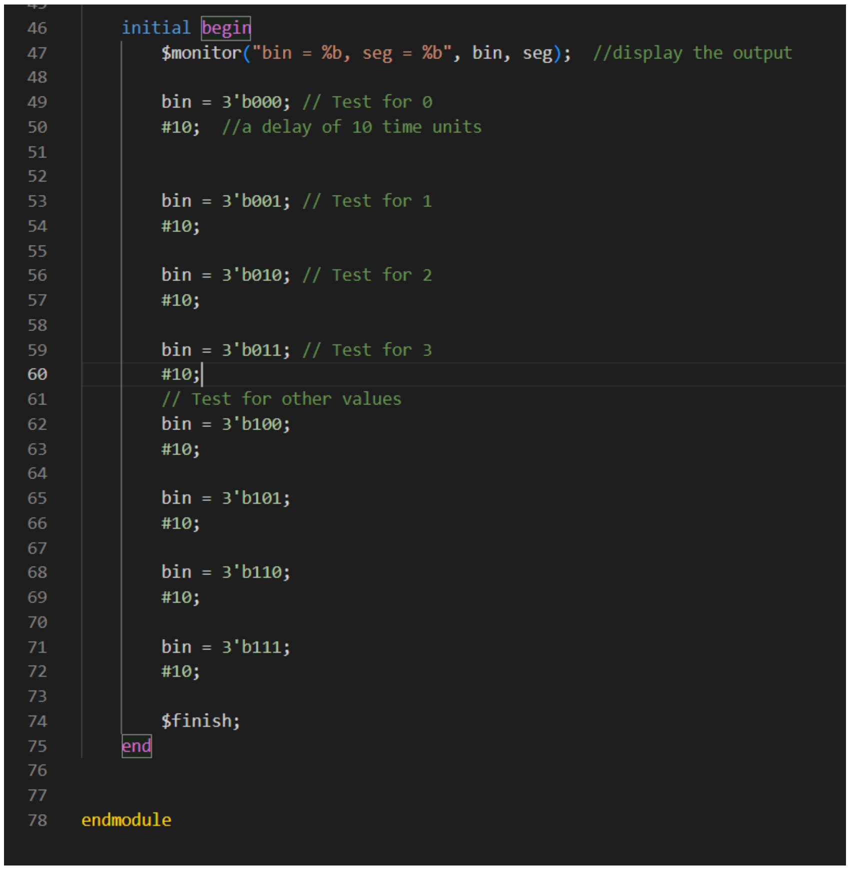
```verilog
module seven_segment_encoder(
    input [2:0] bin,
    output reg seg_a,
    output reg seg_b,
    output reg seg_c,
    output reg seg_d,
    output reg seg_e,
    output reg seg_f,
    output reg seg_g
);

always @(bin)
begin
    case(bin)
        3'b000: begin seg_a = 1'b1; seg_b = 1'b1; seg_c = 1'b1; seg_d = 1'b1; seg_e = 1'b1; seg_f = 1'b1; seg_g = 1'b0; end // 0
        3'b001: begin seg_a = 1'b0; seg_b = 1'b1; seg_c = 1'b1; seg_d = 1'b0; seg_e = 1'b0; seg_f = 1'b0; seg_g = 1'b0; end // 1
        3'b010: begin seg_a = 1'b1; seg_b = 1'b1; seg_c = 1'b0; seg_d = 1'b1; seg_e = 1'b1; seg_f = 1'b0; seg_g = 1'b1; end // 2
        3'b011: begin seg_a = 1'b1; seg_b = 1'b1; seg_c = 1'b1; seg_d = 1'b1; seg_e = 1'b0; seg_f = 1'b0; seg_g = 1'b1; end // 3
        default: begin seg_a = 1'b1; seg_b = 1'b0; seg_c = 1'b0; seg_d = 1'b1; seg_e = 1'b1; seg_f = 1'b1; seg_g = 1'b1; end // E
    endcase
end

endmodule
```

Group 1

Mohamed Mohamed tarek Mohamed salah Abdalla Elgohary waleed Emad



```verilog
module tb_encoder;  //make module to test the encoder
    reg [2:0] bin;  //reg as we assign values to it in the inital block
    wire seg_a; //wire as it it the output of the module
    wire seg_b;
    wire seg_c;
    wire seg_d;
    wire seg_e;
    wire seg_g;
    wire seg_f;


    seven_segment_encoder uut_encoder (  //make an instant of the module
        .bin(bin),
        .seg_a(seg_a),
        .seg_b(seg_b),
        .seg_c(seg_c),
        .seg_d(seg_d),
        .seg_e(seg_e),
        .seg_g(seg_g),
        .seg_f(seg_f)
    );

    initial begin
        $monitor("bin = %b, seg = %b", bin, seg);  //display the output

        bin = 3'b000; // Test for 0
        #10;  //a delay of 10 time units


        bin = 3'b001; // Test for 1
        #10;

```

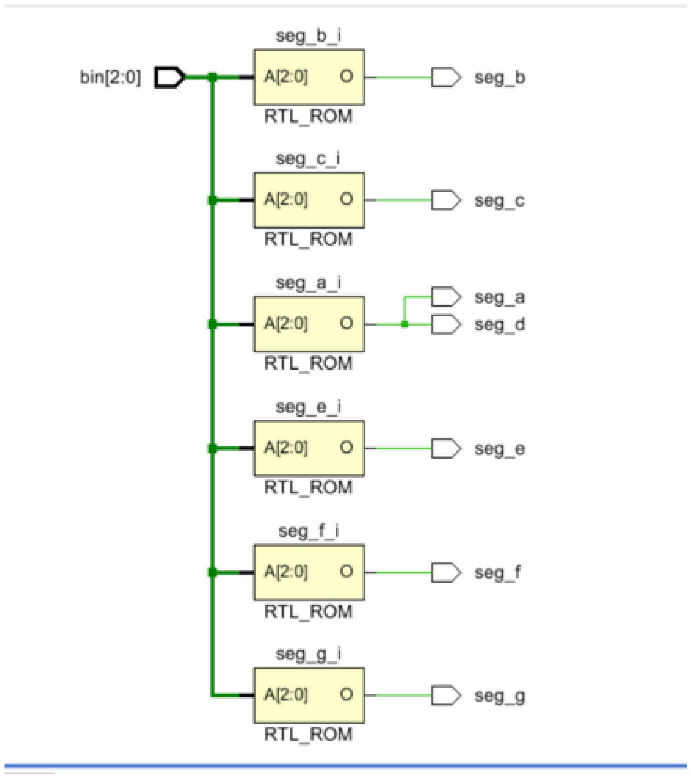


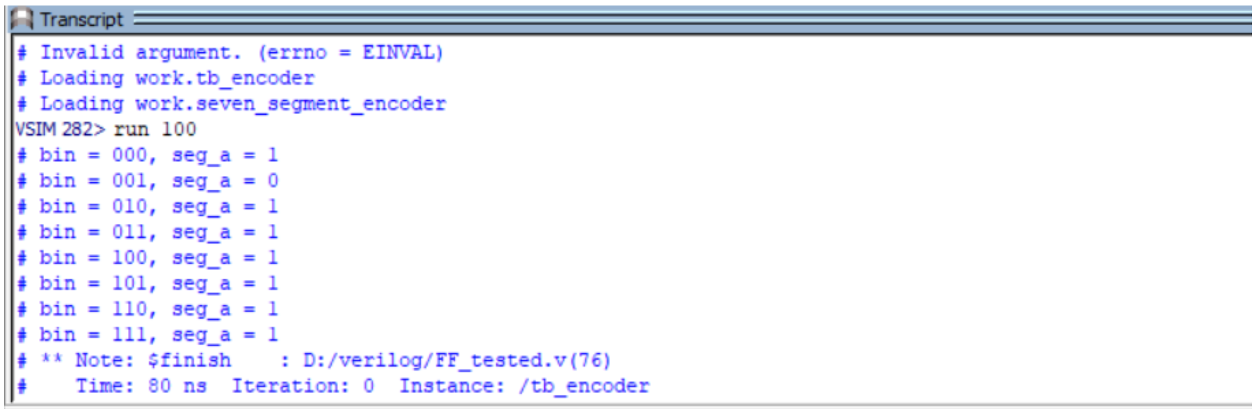
```verilog
    initial begin
        $monitor("bin = %b, seg = %b", bin, seg);  //display the output

        bin = 3'b000; // Test for 0
        #10;  //a delay of 10 time units


        bin = 3'b001; // Test for 1
        #10;

```

```verilog
        bin = 3'b010; // Test for 2
        #10;

        bin = 3'b011; // Test for 3
        #10;
        // Test for other values
        bin = 3'b100;
        #10;

        bin = 3'b101;
        #10;

        bin = 3'b110;
        #10;

        bin = 3'b111;
        #10;

        $finish;
    end


endmodule
```

Group 1

Mohamed Mohamed tarek Mohamed salah Abdalla Elgohary waleed Emad



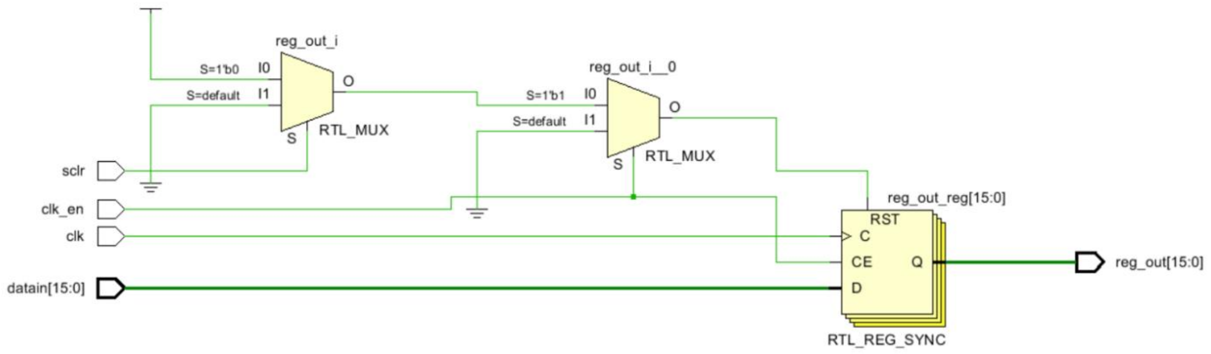


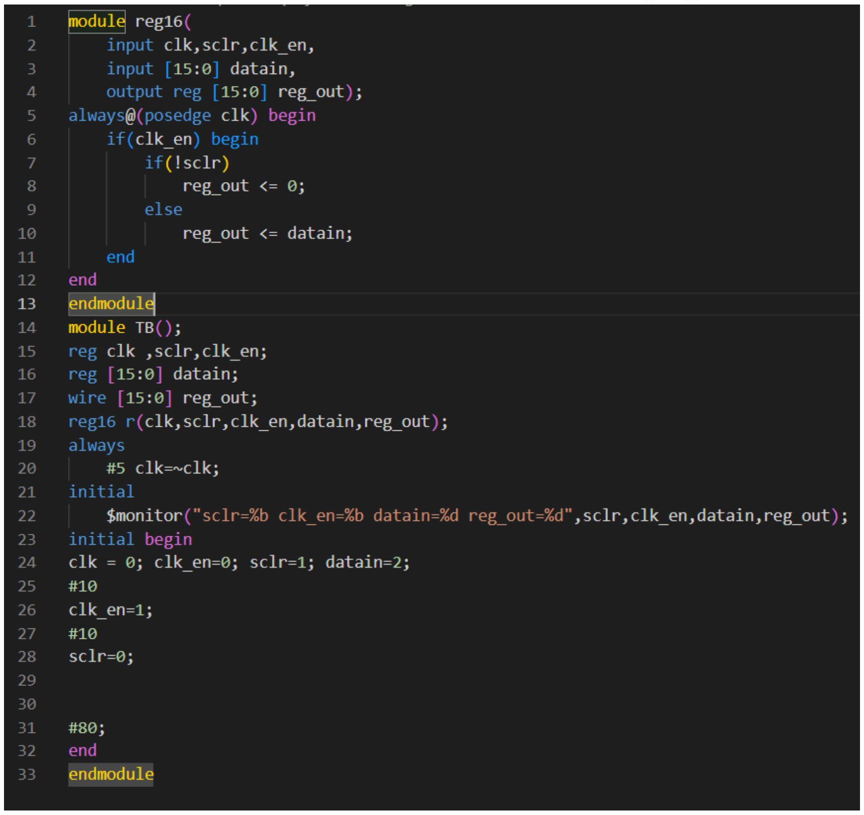
```
# Invalid argument. (errno = EINVAL)
# Loading work.tb_encoder
# Loading work.seven_segment_encoder
VSIM 282> run 100
# bin = 000, seg_a = 1
# bin = 001, seg_a = 0
# bin = 010, seg_a = 1
# bin = 011, seg_a = 1
# bin = 100, seg_a = 1
# bin = 101, seg_a = 1
# bin = 110, seg_a = 1
# bin = 111, seg_a = 1
# ** Note: $finish    : D:/verilog/FF_tested.v(76)
#    Time: 80 ns  Iteration: 0  Instance: /tb_encoder
```

Group 1

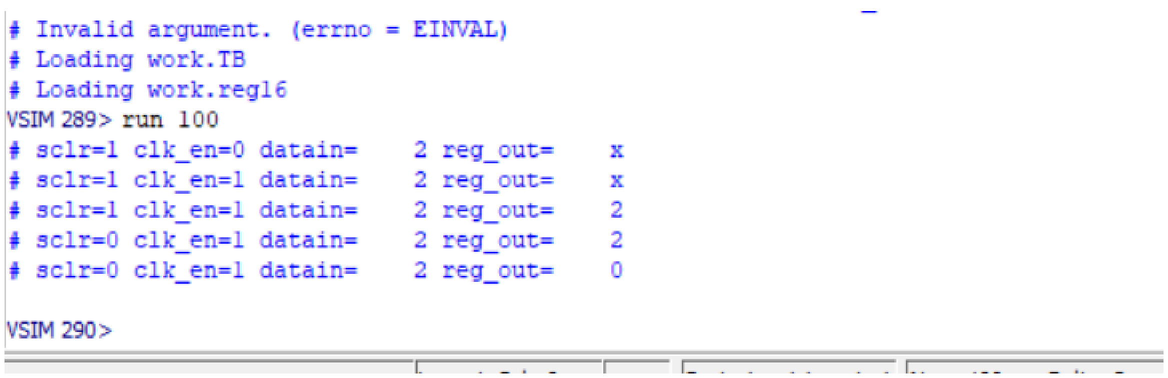
Mohamed Mohamed tarek   Mohamed salah   Abdalla Elgohary   waleed Emad

reg16





```
module reg16(
    input clk,sclr,clk_en,
    input [15:0] datain,
    output reg [15:0] reg_out);
always@(posedge clk) begin
    if(clk_en) begin
        if(!sclr)
            reg_out <= 0;
        else
            reg_out <= datain;
    end
end
endmodule
module TB();
reg clk ,sclr,clk_en;
reg [15:0] datain;
wire [15:0] reg_out;
reg16 r(clk,sclr,clk_en,datain,reg_out);
always
    #5 clk=~clk;
initial
    $monitor("sclr=%b clk_en=%b datain=%d reg_out=%d",sclr,clk_en,datain,reg_out);
initial begin
clk = 0; clk_en=0; sclr=1; datain=2;
#10
clk_en=1;
#10
sclr=0;


#80;
end
endmodule
```

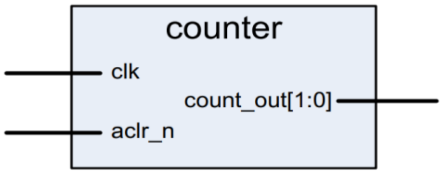


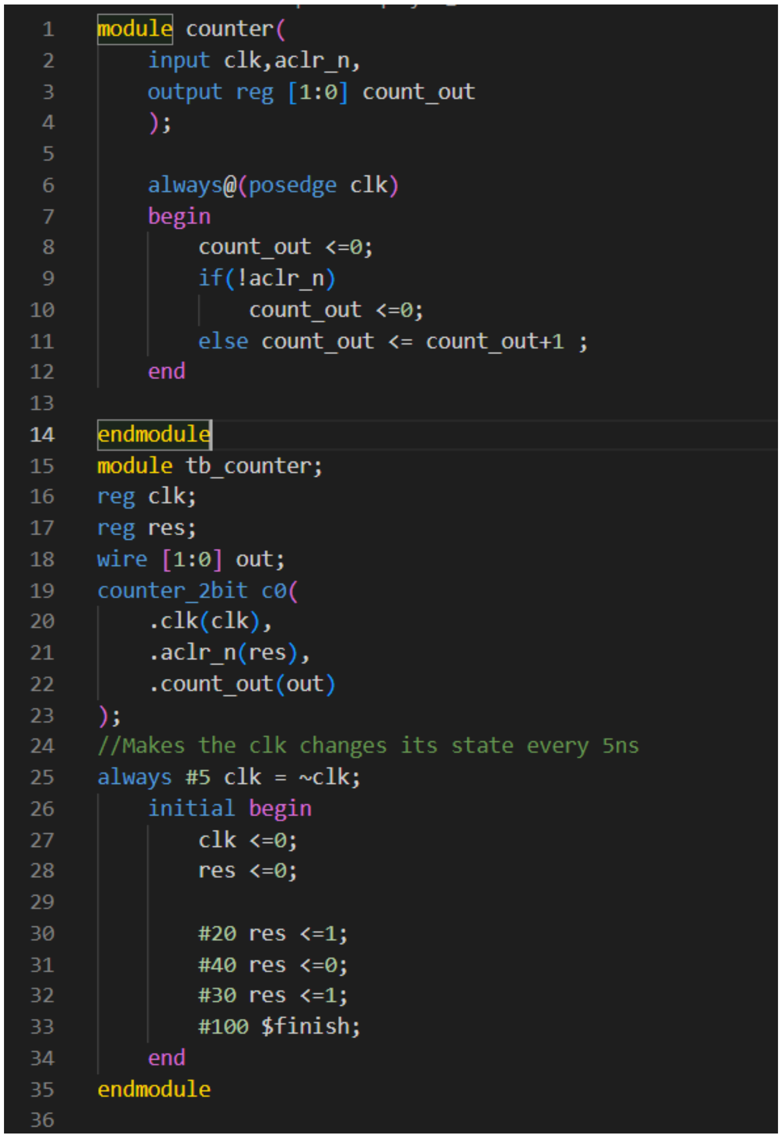
```
# Invalid argument. (errno = EINVAL)
# Loading work.TB
# Loading work.reg16
VSIM 289> run 100
# sclr=1 clk_en=0 datain=    2 reg_out=    x
# sclr=1 clk_en=1 datain=    2 reg_out=    x
# sclr=1 clk_en=1 datain=    2 reg_out=    2
# sclr=0 clk_en=1 datain=    2 reg_out=    2
# sclr=0 clk_en=1 datain=    2 reg_out=    0

VSIM 290>
```

Group 1

Mohamed Mohamed tarek Mohamed salah Abdalla Elgohary waleed Emad

2-bit Counter with asynchronous control:

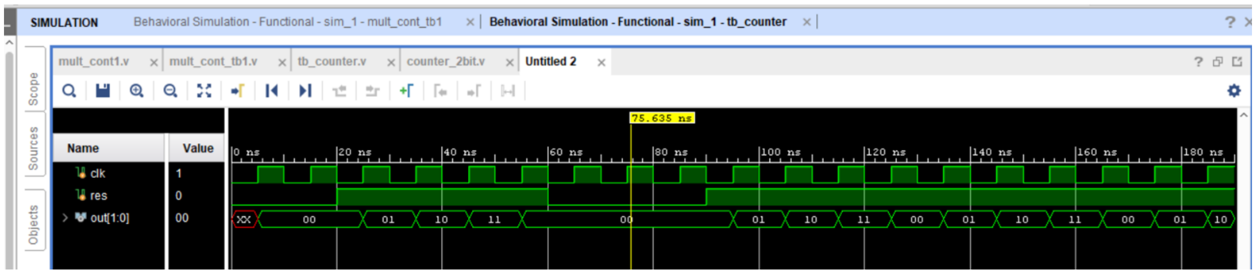


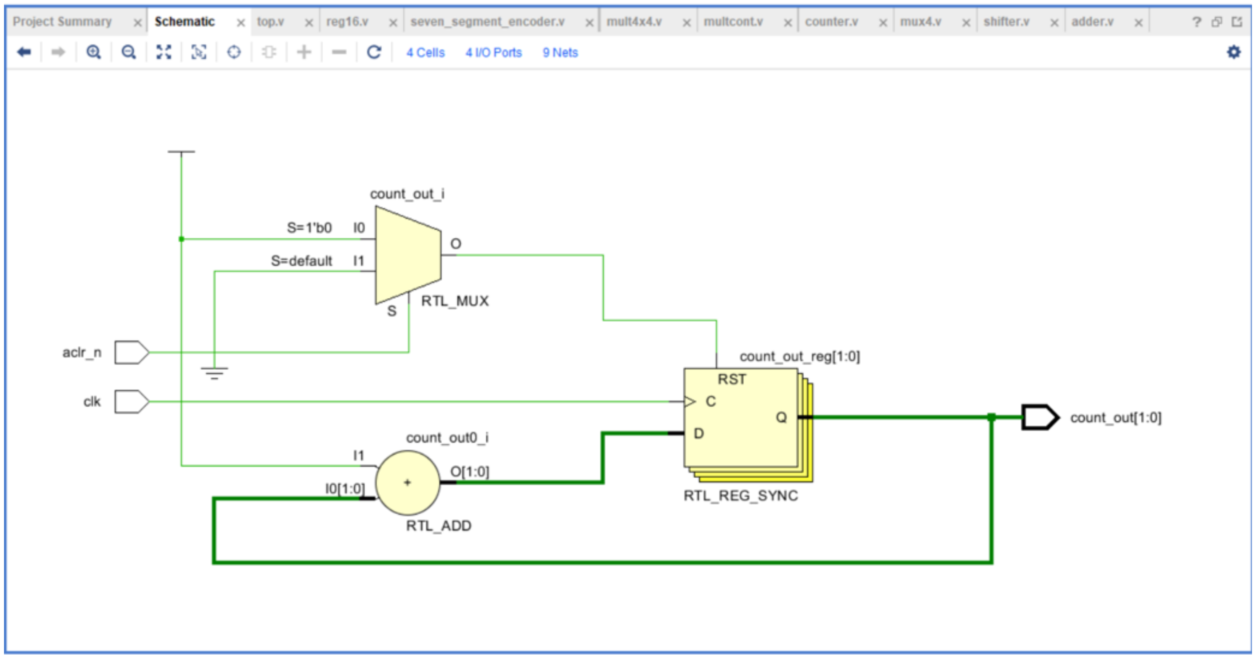


```verilog
module counter(
    input clk,aclr_n,
    output reg [1:0] count_out
    );

    always@(posedge clk)
    begin
        count_out <=0;
        if(!aclr_n)
            count_out <=0;
        else count_out <= count_out+1 ;
    end

endmodule
module tb_counter;
reg clk;
reg res;
wire [1:0] out;
counter_2bit c0(
    .clk(clk),
    .aclr_n(res),
    .count_out(out)
);
//Makes the clk changes its state every 5ns
always #5 clk = ~clk;
    initial begin
        clk <=0;
        res <=0;

        #20 res <=1;
        #40 res <=0;
        #30 res <=1;
        #100 $finish;
    end
endmodule
```

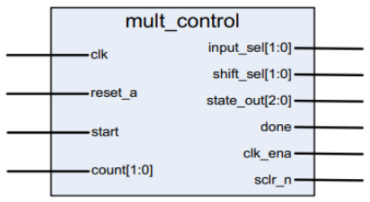
Group 1

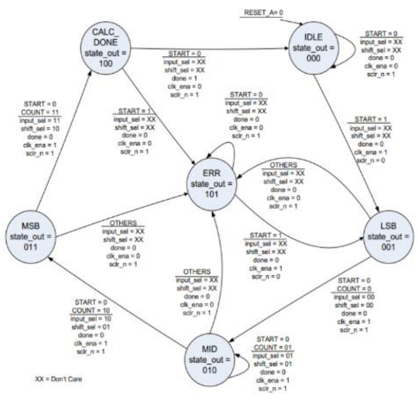
Mohamed Mohamed tarek Mohamed salah Abdalla Elgohary waleed Emad





[Multiplier controller]

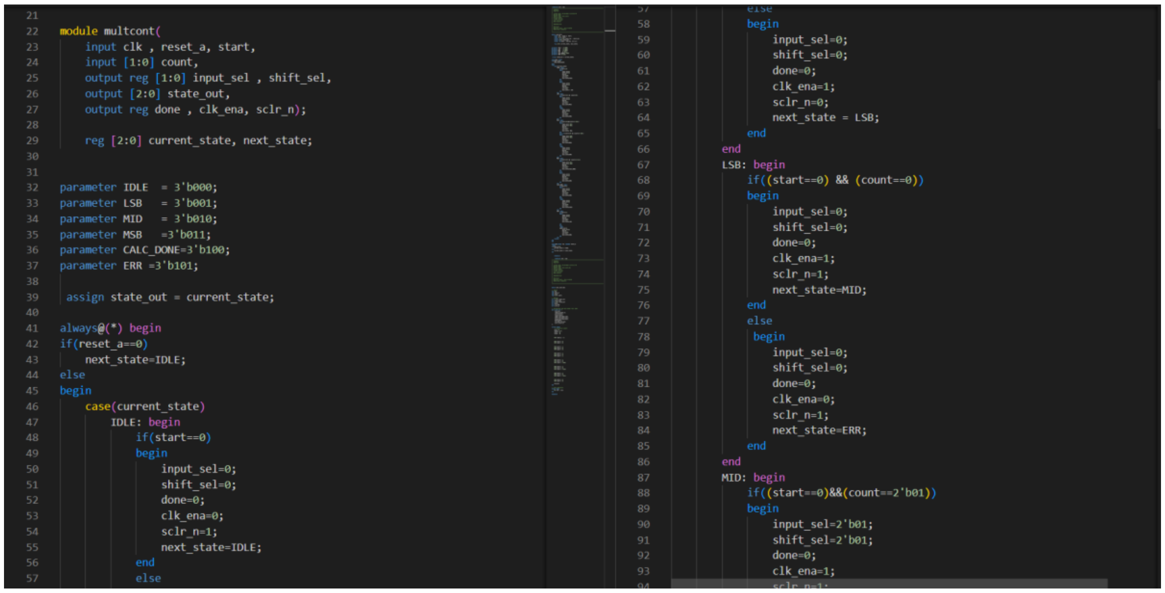



RESET_A= 0
CALC_DONE state_out = 100
IDLE state_out = 000
ERR state_out = 101
MSB state_out = 011
LSB state_out = 001
MID state_out = 010
START = 0 / input_sel = XX / shift_sel = XX / done = 1 / clk_ena = 0 / sclr_n = 1
START = 0 / input_sel = XX / shift_sel = XX / done = 0 / clk_ena = 0 / sclr_n = 1
START = 0 / COUNT = 11 / input_sel = 11 / shift_sel = 10 / done = 0 / clk_ena = 1 / sclr_n = 1
START = 1 / input_sel = XX / shift_sel = XX / done = 0 / clk_ena = 0 / sclr_n = 1
START = 0 / input_sel = XX / shift_sel = XX / done = 0 / clk_ena = 0 / sclr_n = 1
START = 1 / input_sel = XX / shift_sel = XX / done = 0 / clk_ena = 1 / sclr_n = 0
OTHERS / input_sel = XX / shift_sel = XX / done = 0 / clk_ena = 0 / sclr_n = 1
OTHERS / input_sel = XX / shift_sel = XX / done = 0 / clk_ena = 0 / sclr_n = 1
START = 1 / input_sel = XX / shift_sel = XX / done = 0 / clk_ena = 1 / sclr_n = 0
OTHERS / input_sel = XX / shift_sel = XX / done = 0 / clk_ena = 0 / sclr_n = 1
START = 0 / COUNT = 0 / input_sel = 00 / shift_sel = 00 / done = 0 / clk_ena = 1 / sclr_n = 1
START = 0 / COUNT = 10 / input_sel = 10 / shift_sel = 01 / done = 0 / clk_ena = 1 / sclr_n = 1
START = 0 / COUNT = 01 / input_sel = 01 / shift_sel = 01 / done = 0 / clk_ena = 1 / sclr_n = 1
XX = Don't Care

The outputs of mult_control state machine control the other blocks in the design

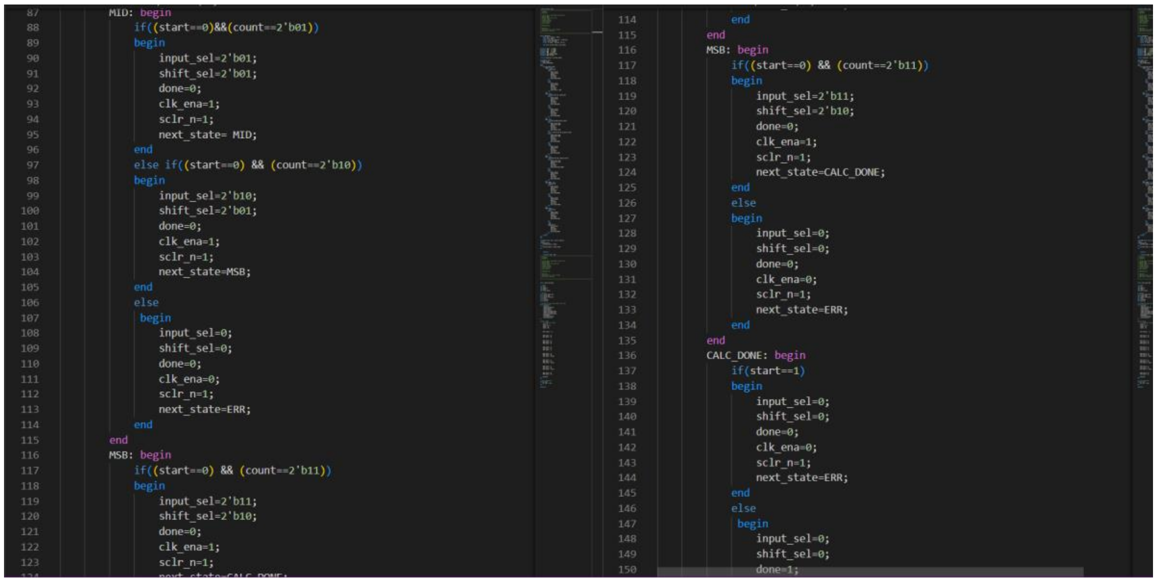
Group 1

Mohamed Mohamed tarek   Mohamed salah   Abdalla Elgohary   waleed Emad



```verilog

module multcont(
    input clk , reset_a, start,
    input [1:0] count,
    output reg [1:0] input_sel , shift_sel,
    output [2:0] state_out,
    output reg done , clk_ena, sclr_n);

    reg [2:0] current_state, next_state;


parameter IDLE  = 3'b000;
parameter LSB   = 3'b001;
parameter MID   = 3'b010;
parameter MSB   =3'b011;
parameter CALC_DONE=3'b100;
parameter ERR =3'b101;

 assign state_out = current_state;

always@(*) begin
if(reset_a==0)
    next_state=IDLE;
else
begin
    case(current_state)
        IDLE: begin
            if(start==0)
            begin
                input_sel=0;
                shift_sel=0;
                done=0;
                clk_ena=0;
                sclr_n=1;
                next_state=IDLE;
            end
            else
```

```verilog
            begin
                input_sel=0;
                shift_sel=0;
                done=0;
                clk_ena=1;
                sclr_n=0;
                next_state = LSB;
            end
        end
        LSB: begin
            if((start==0) && (count==0))
            begin
                input_sel=0;
                shift_sel=0;
                done=0;
                clk_ena=1;
                sclr_n=1;
                next_state=MID;
            end
            else
             begin
                input_sel=0;
                shift_sel=0;
                done=0;
                clk_ena=0;
                sclr_n=1;
                next_state=ERR;
            end
        end
        MID: begin
            if((start==0)&&(count==2'b01))
            begin
                input_sel=2'b01;
                shift_sel=2'b01;
                done=0;
                clk_ena=1;
                sclr_n=1;
```

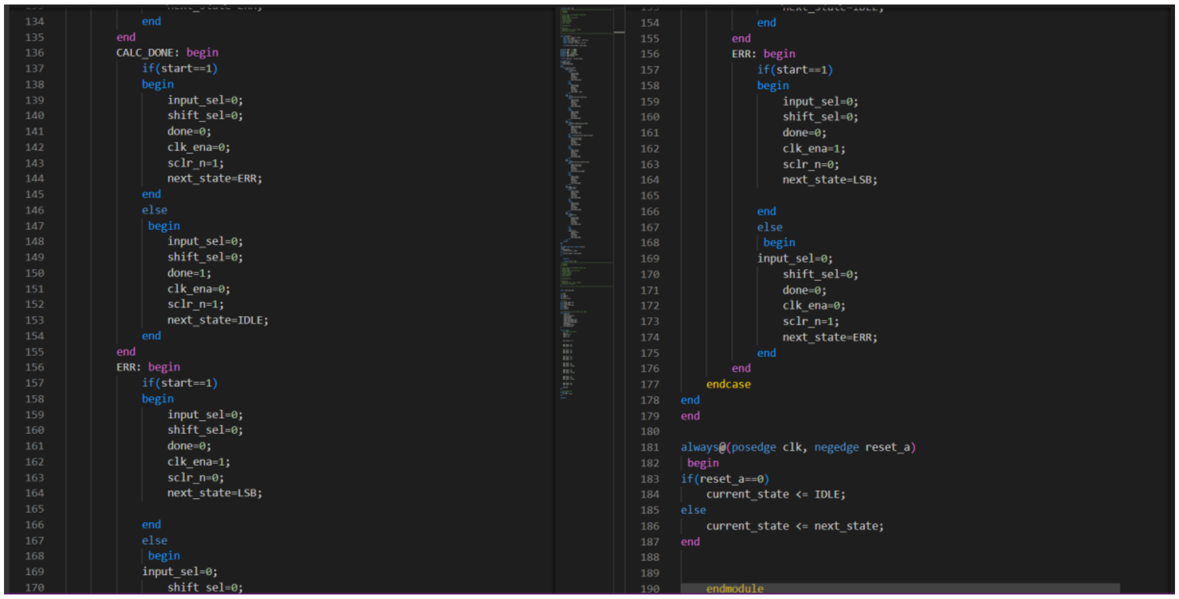


```verilog
        MID: begin
            if((start==0)&&(count==2'b01))
            begin
                input_sel=2'b01;
                shift_sel=2'b01;
                done=0;
                clk_ena=1;
                sclr_n=1;
                next_state= MID;
            end
            else if((start==0) && (count==2'b10))
            begin
                input_sel=2'b10;
                shift_sel=2'b01;
                done=0;
                clk_ena=1;
                sclr_n=1;
                next_state=MSB;
            end
            else
             begin
                input_sel=0;
                shift_sel=0;
                done=0;
                clk_ena=0;
                sclr_n=1;
                next_state=ERR;
            end
        end
        MSB: begin
            if((start==0) && (count==2'b11))
            begin
                input_sel=2'b11;
                shift_sel=2'b10;
                done=0;
                clk_ena=1;
                sclr_n=1;
                next_state=CALC_DONE;
```

```verilog
            end
        end
        MSB: begin
            if((start==0) && (count==2'b11))
            begin
                input_sel=2'b11;
                shift_sel=2'b10;
                done=0;
                clk_ena=1;
                sclr_n=1;
                next_state=CALC_DONE;
            end
            else
            begin
                input_sel=0;
                shift_sel=0;
                done=0;
                clk_ena=0;
                sclr_n=1;
                next_state=ERR;
            end
        end
        CALC_DONE: begin
            if(start==1)
            begin
                input_sel=0;
                shift_sel=0;
                done=0;
                clk_ena=0;
                sclr_n=1;
                next_state=ERR;
            end
            else
             begin
                input_sel=0;
                shift_sel=0;
                done=1;
```

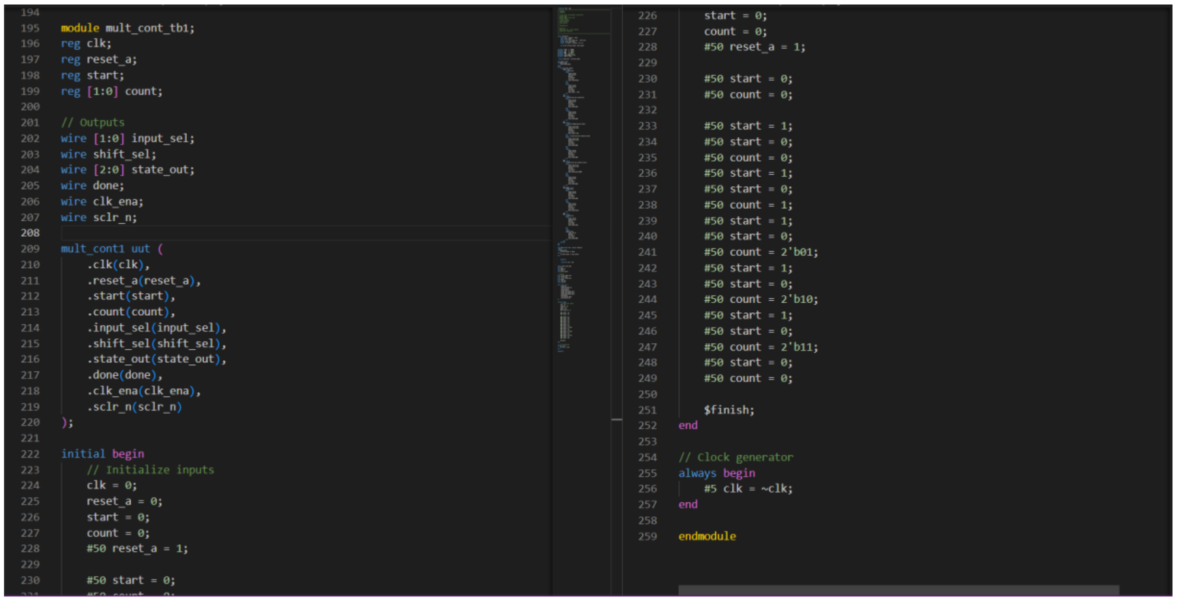
Group 1

Mohamed Mohamed tarek   Mohamed salah   Abdalla Elgohary   waleed Emad



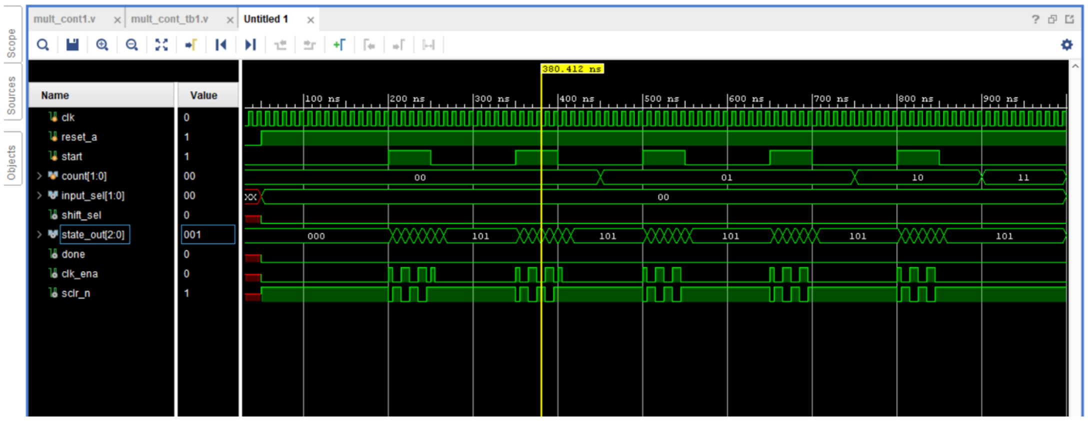
```
                end
            end
            CALC_DONE: begin
                if(start==1)
                begin
                    input_sel=0;
                    shift_sel=0;
                    done=0;
                    clk_ena=0;
                    sclr_n=1;
                    next_state=ERR;
                end
                else
                 begin
                    input_sel=0;
                    shift_sel=0;
                    done=1;
                    clk_ena=0;
                    sclr_n=1;
                    next_state=IDLE;
                end
            end
            ERR: begin
                if(start==1)
                begin
                    input_sel=0;
                    shift_sel=0;
                    done=0;
                    clk_ena=1;
                    sclr_n=0;
                    next_state=LSB;

                end
                else
                 begin
                input_sel=0;
                    shift_sel=0;
```

```
                end
            end
            ERR: begin
                if(start==1)
                begin
                    input_sel=0;
                    shift_sel=0;
                    done=0;
                    clk_ena=1;
                    sclr_n=0;
                    next_state=LSB;

                end
                else
                 begin
                input_sel=0;
                    shift_sel=0;
                    done=0;
                    clk_ena=0;
                    sclr_n=1;
                    next_state=ERR;
                end
            end
        endcase
end
end

always@(posedge clk, negedge reset_a)
 begin
if(reset_a==0)
    current_state <= IDLE;
else
    current_state <= next_state;
end


    endmodule
```



```

module mult_cont_tb1;
reg clk;
reg reset_a;
reg start;
reg [1:0] count;

// Outputs
wire [1:0] input_sel;
wire shift_sel;
wire [2:0] state_out;
wire done;
wire clk_ena;
wire sclr_n;

mult_cont1 uut (
    .clk(clk),
    .reset_a(reset_a),
    .start(start),
    .count(count),
    .input_sel(input_sel),
    .shift_sel(shift_sel),
    .state_out(state_out),
    .done(done),
    .clk_ena(clk_ena),
    .sclr_n(sclr_n)
);

initial begin
    // Initialize inputs
    clk = 0;
    reset_a = 0;
    start = 0;
    count = 0;
    #50 reset_a = 1;

    #50 start = 0;
```

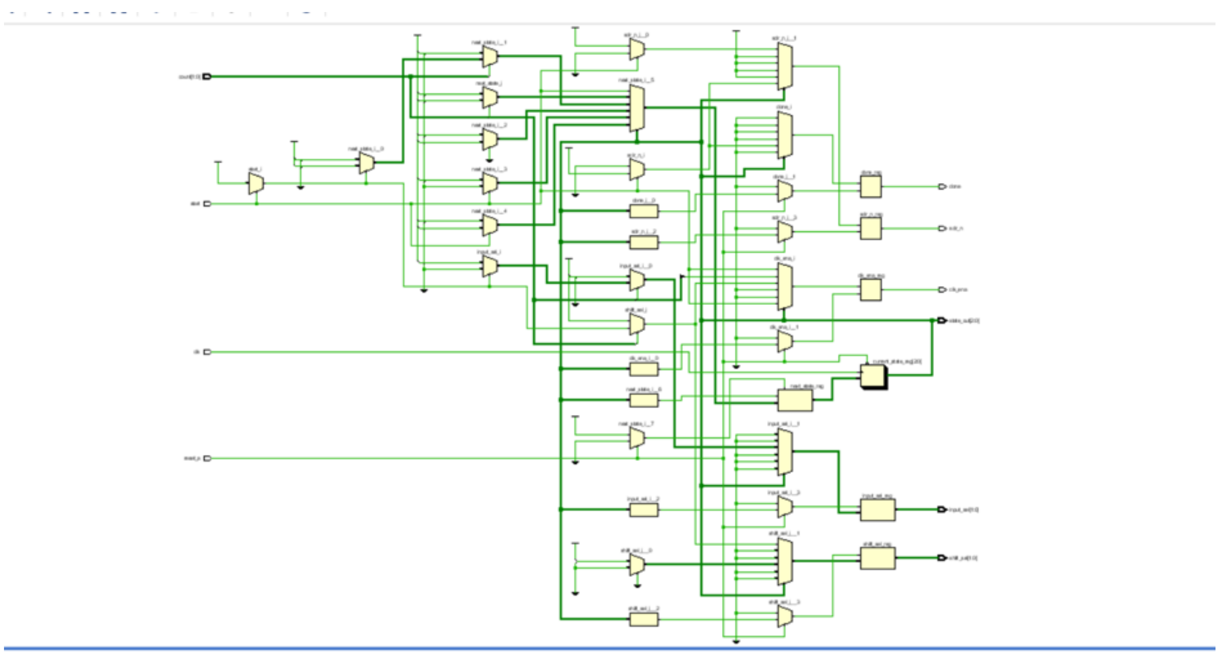
```
    start = 0;
    count = 0;
    #50 reset_a = 1;

    #50 start = 0;
    #50 count = 0;

    #50 start = 1;
    #50 start = 0;
    #50 count = 0;
    #50 start = 1;
    #50 start = 0;
    #50 count = 1;
    #50 start = 1;
    #50 start = 0;
    #50 count = 2'b01;
    #50 start = 1;
    #50 start = 0;
    #50 count = 2'b10;
    #50 start = 1;
    #50 start = 0;
    #50 count = 2'b11;
    #50 start = 0;
    #50 count = 0;

    $finish;
end

// Clock generator
always begin
    #5 clk = ~clk;
end

endmodule
```



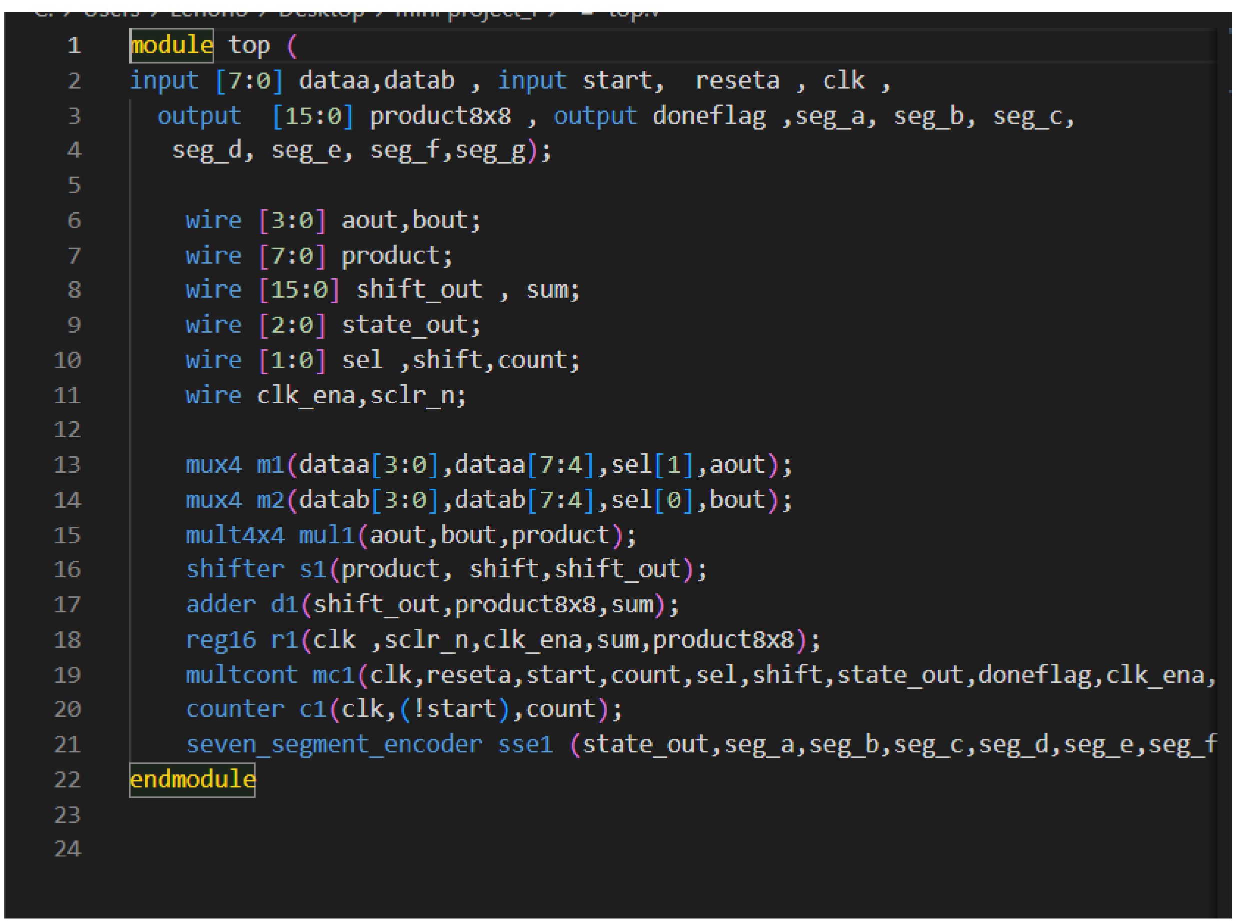
Name	Value
clk	0
reset_a	1
start	1
count[1:0]	00
input_sel[1:0]	00
shift_sel	0
state_out[2:0]	001
done	0
clk_ena	0
sclr_n	1

Group 1

Mohamed Mohamed tarek Mohamed salah Abdalla Elgohary waleed Emad



Top module :

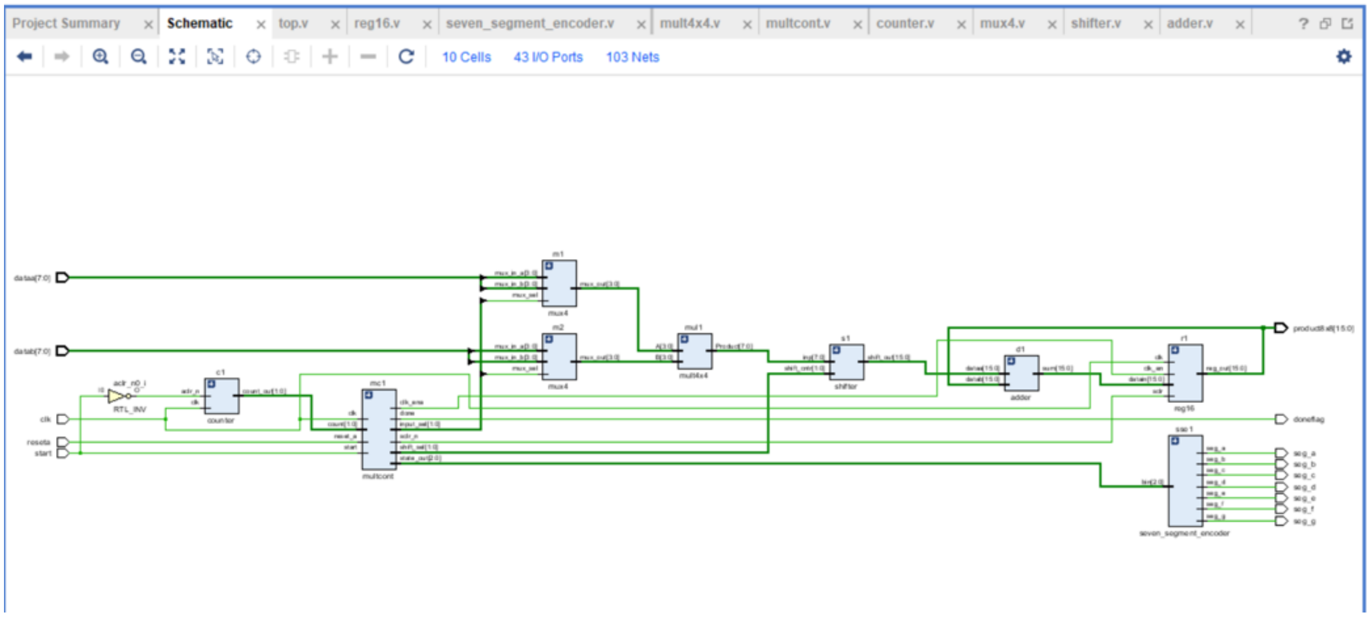


```verilog
module top (
input [7:0] dataa,datab , input start,  reseta , clk ,
  output  [15:0] product8x8 , output doneflag ,seg_a, seg_b, seg_c,
   seg_d, seg_e, seg_f,seg_g);

    wire [3:0] aout,bout;
    wire [7:0] product;
    wire [15:0] shift_out , sum;
    wire [2:0] state_out;
    wire [1:0] sel ,shift,count;
    wire clk_ena,sclr_n;

    mux4 m1(dataa[3:0],dataa[7:4],sel[1],aout);
    mux4 m2(datab[3:0],datab[7:4],sel[0],bout);
    mult4x4 mul1(aout,bout,product);
    shifter s1(product, shift,shift_out);
    adder d1(shift_out,product8x8,sum);
    reg16 r1(clk ,sclr_n,clk_ena,sum,product8x8);
    multcont mc1(clk,reseta,start,count,sel,shift,state_out,doneflag,clk_ena,
    counter c1(clk,(!start),count);
    seven_segment_encoder sse1 (state_out,seg_a,seg_b,seg_c,seg_d,seg_e,seg_f
endmodule
```

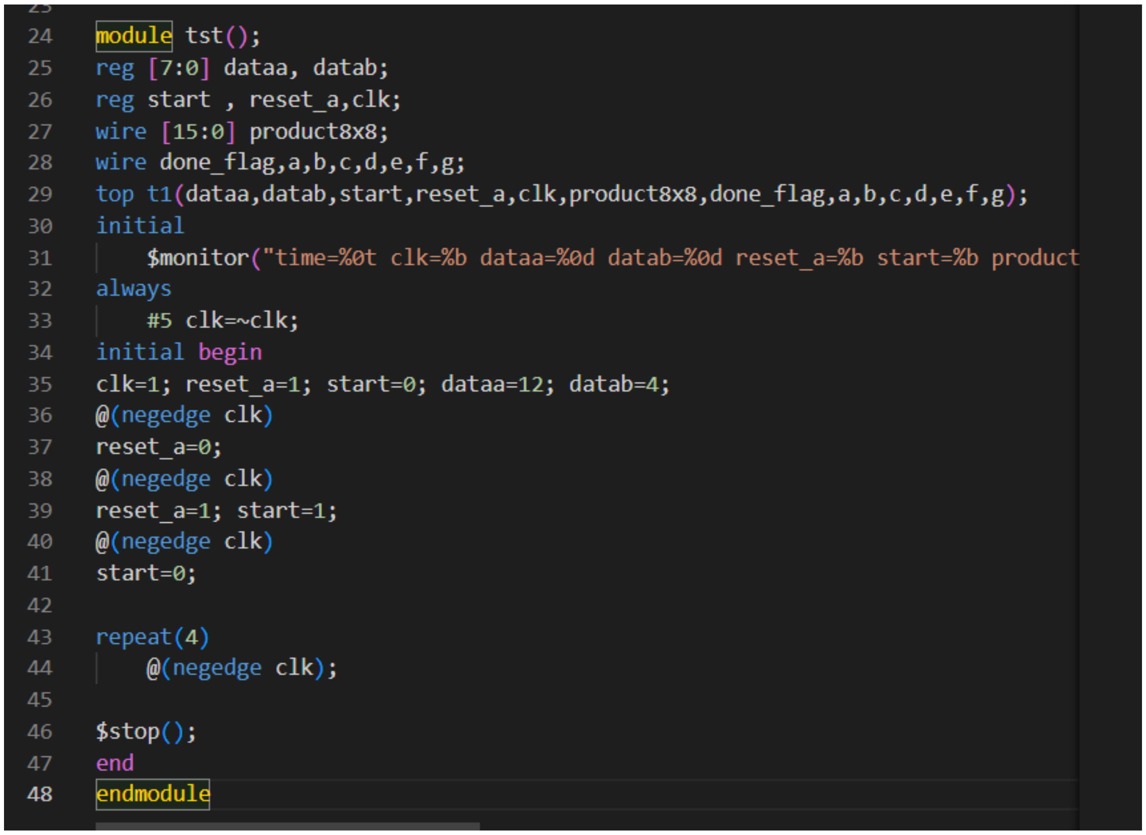
Group 1

Mohamed Mohamed tarek Mohamed salah Abdalla Elgohary waleed Emad

Top schematic



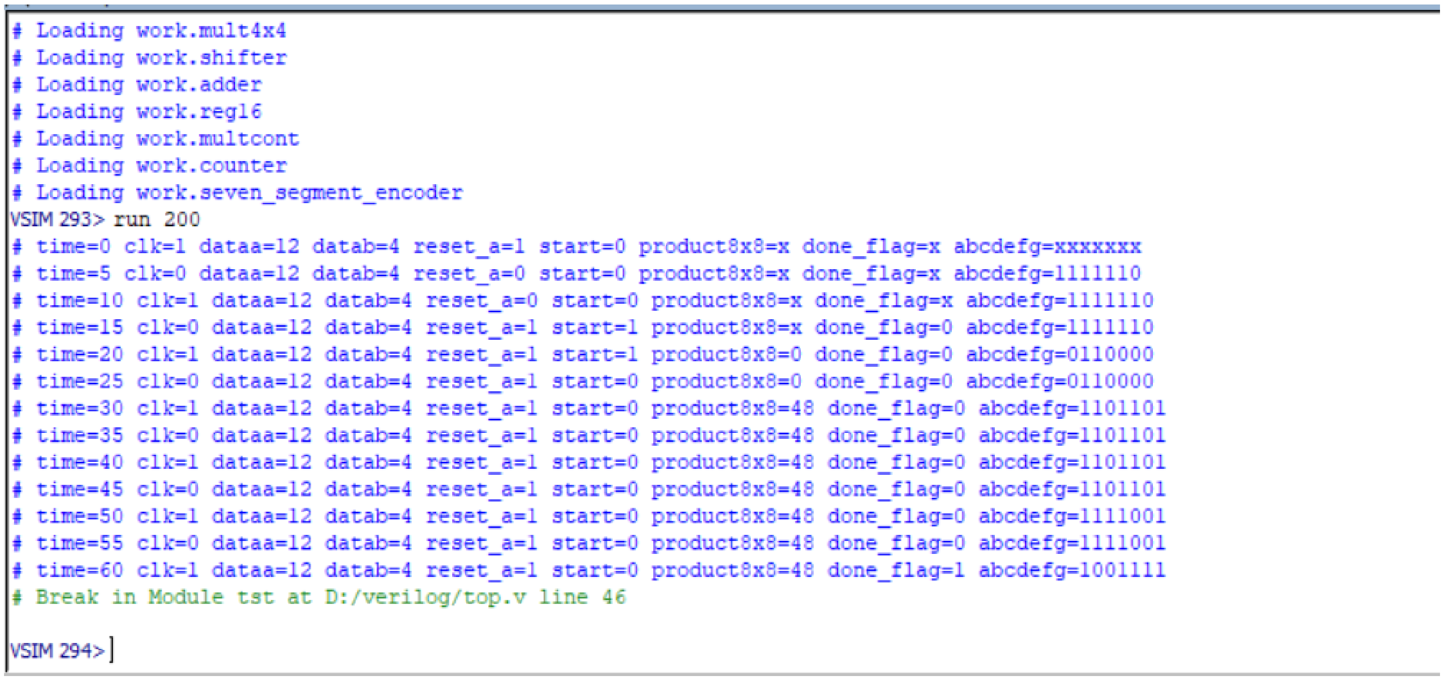
Top Test bench :



```
module tst();
reg [7:0] dataa, datab;
reg start , reset_a,clk;
wire [15:0] product8x8;
wire done_flag,a,b,c,d,e,f,g;
top t1(dataa,datab,start,reset_a,clk,product8x8,done_flag,a,b,c,d,e,f,g);
initial
    $monitor("time=%0t clk=%b dataa=%0d datab=%0d reset_a=%b start=%b product
always
    #5 clk=~clk;
initial begin
clk=1; reset_a=1; start=0; dataa=12; datab=4;
@(negedge clk)
reset_a=0;
@(negedge clk)
reset_a=1; start=1;
@(negedge clk)
start=0;

repeat(4)
    @(negedge clk);

$stop();
end
endmodule
```

Group 1

Mohamed Mohamed tarek   Mohamed salah   Abdalla Elgohary   waleed Emad



```
# Loading work.mult4x4
# Loading work.shifter
# Loading work.adder
# Loading work.reg16
# Loading work.multcont
# Loading work.counter
# Loading work.seven_segment_encoder
VSIM 293> run 200
# time=0 clk=1 dataa=12 datab=4 reset_a=1 start=0 product8x8=x done_flag=x abcdefg=xxxxxxx
# time=5 clk=0 dataa=12 datab=4 reset_a=0 start=0 product8x8=x done_flag=x abcdefg=1111110
# time=10 clk=1 dataa=12 datab=4 reset_a=0 start=0 product8x8=x done_flag=x abcdefg=1111110
# time=15 clk=0 dataa=12 datab=4 reset_a=1 start=1 product8x8=x done_flag=0 abcdefg=1111110
# time=20 clk=1 dataa=12 datab=4 reset_a=1 start=1 product8x8=0 done_flag=0 abcdefg=0110000
# time=25 clk=0 dataa=12 datab=4 reset_a=1 start=0 product8x8=0 done_flag=0 abcdefg=0110000
# time=30 clk=1 dataa=12 datab=4 reset_a=1 start=0 product8x8=48 done_flag=0 abcdefg=1101101
# time=35 clk=0 dataa=12 datab=4 reset_a=1 start=0 product8x8=48 done_flag=0 abcdefg=1101101
# time=40 clk=1 dataa=12 datab=4 reset_a=1 start=0 product8x8=48 done_flag=0 abcdefg=1101101
# time=45 clk=0 dataa=12 datab=4 reset_a=1 start=0 product8x8=48 done_flag=0 abcdefg=1101101
# time=50 clk=1 dataa=12 datab=4 reset_a=1 start=0 product8x8=48 done_flag=0 abcdefg=1111001
# time=55 clk=0 dataa=12 datab=4 reset_a=1 start=0 product8x8=48 done_flag=0 abcdefg=1111001
# time=60 clk=1 dataa=12 datab=4 reset_a=1 start=0 product8x8=48 done_flag=1 abcdefg=1001111
# Break in Module tst at D:/verilog/top.v line 46

VSIM 294>
```