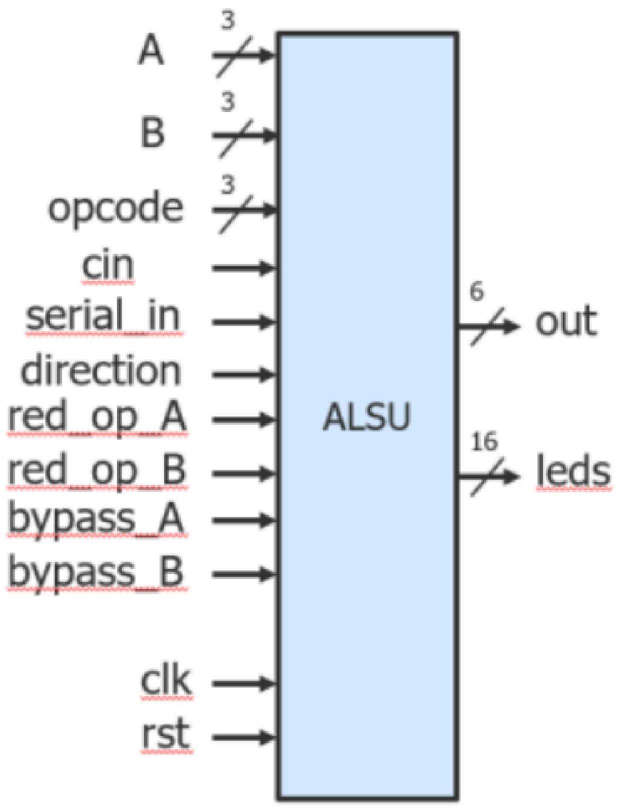
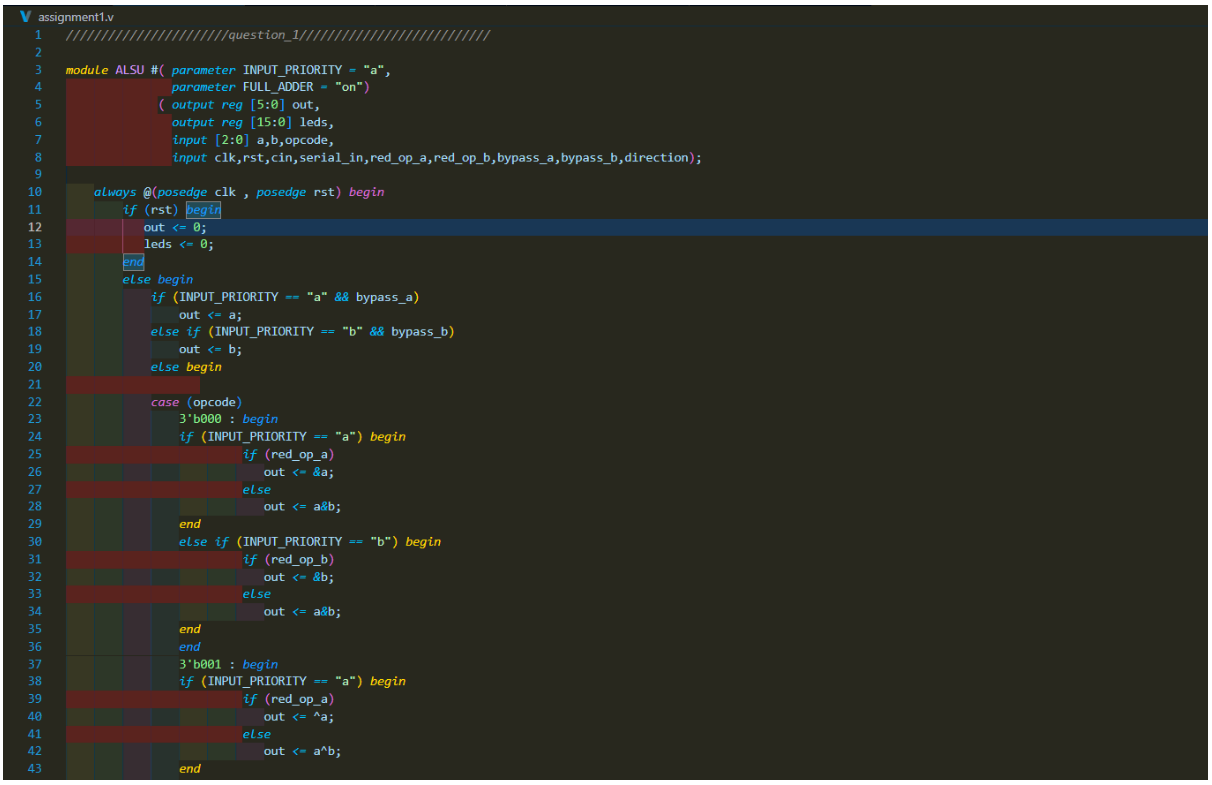
1) ALSU is a logic unit that can perform logical, arithmetic, and shift operations on input ports

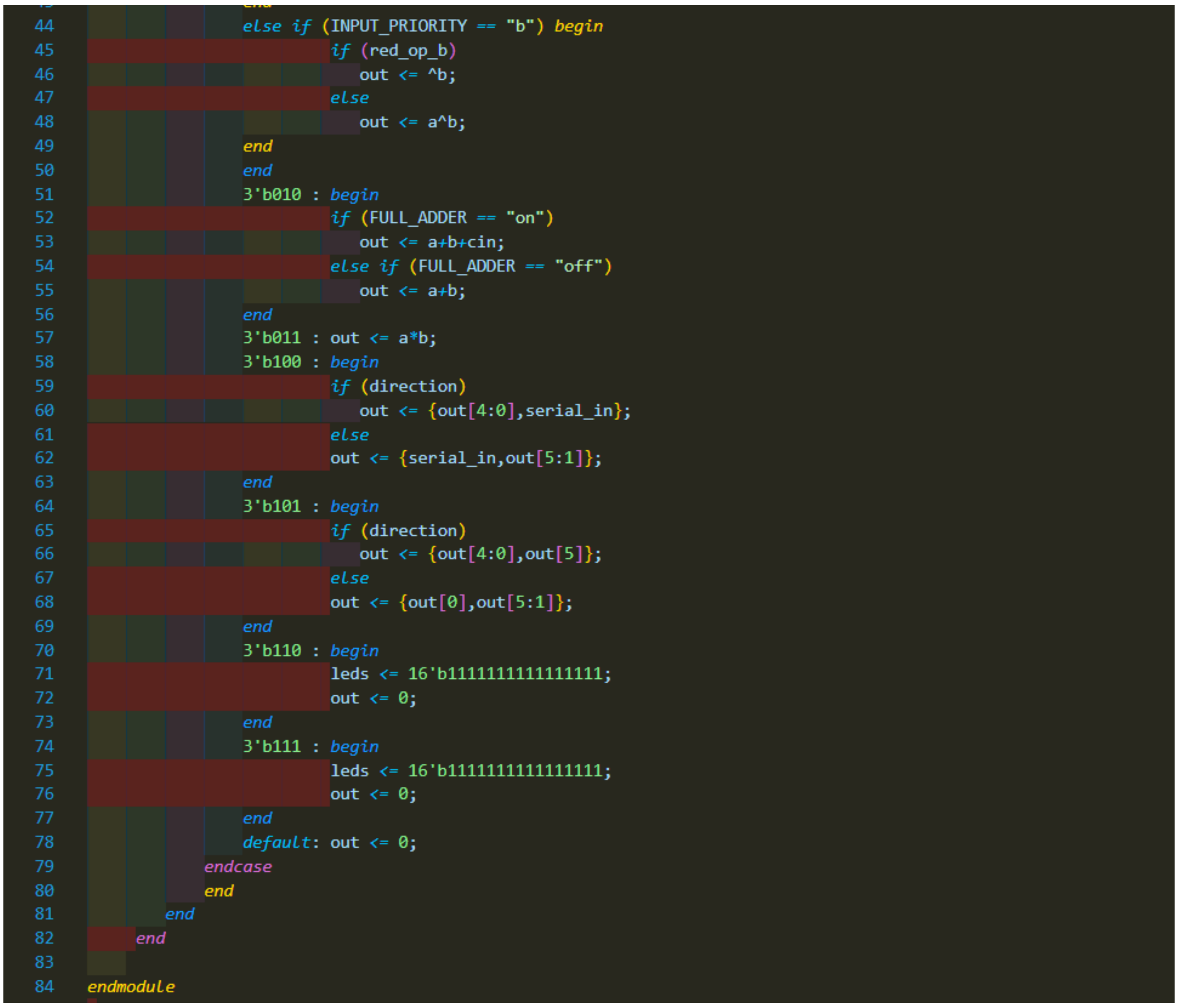
- Input ports A and B have various operations that can take place depending on the value of the opcode.
- Each input bit except for the clk and rst will be sampled at the rising edge before any processing so a D-FF is expected for each input bit at the design entry.
- The output of the ALSU is registered and is available at the rising edge of the clock.



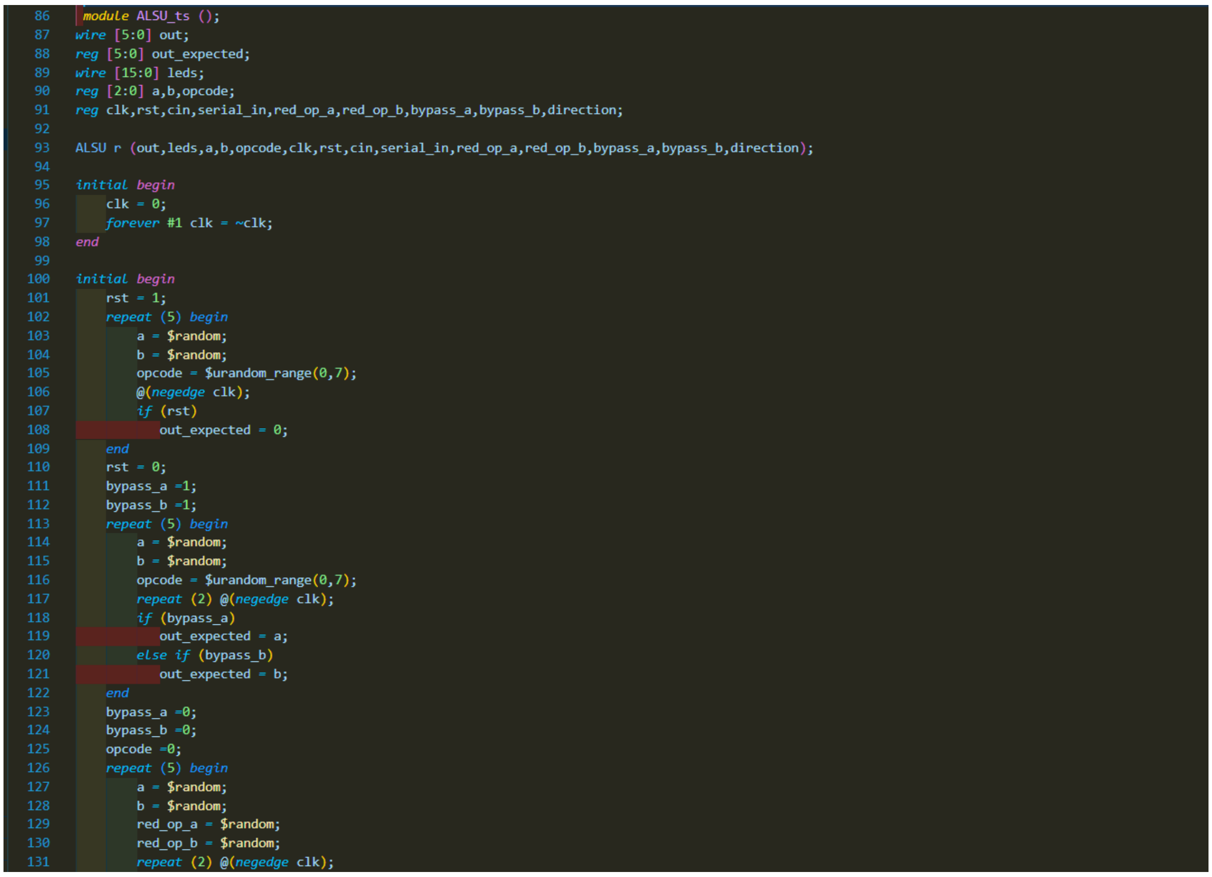


assignment1.v

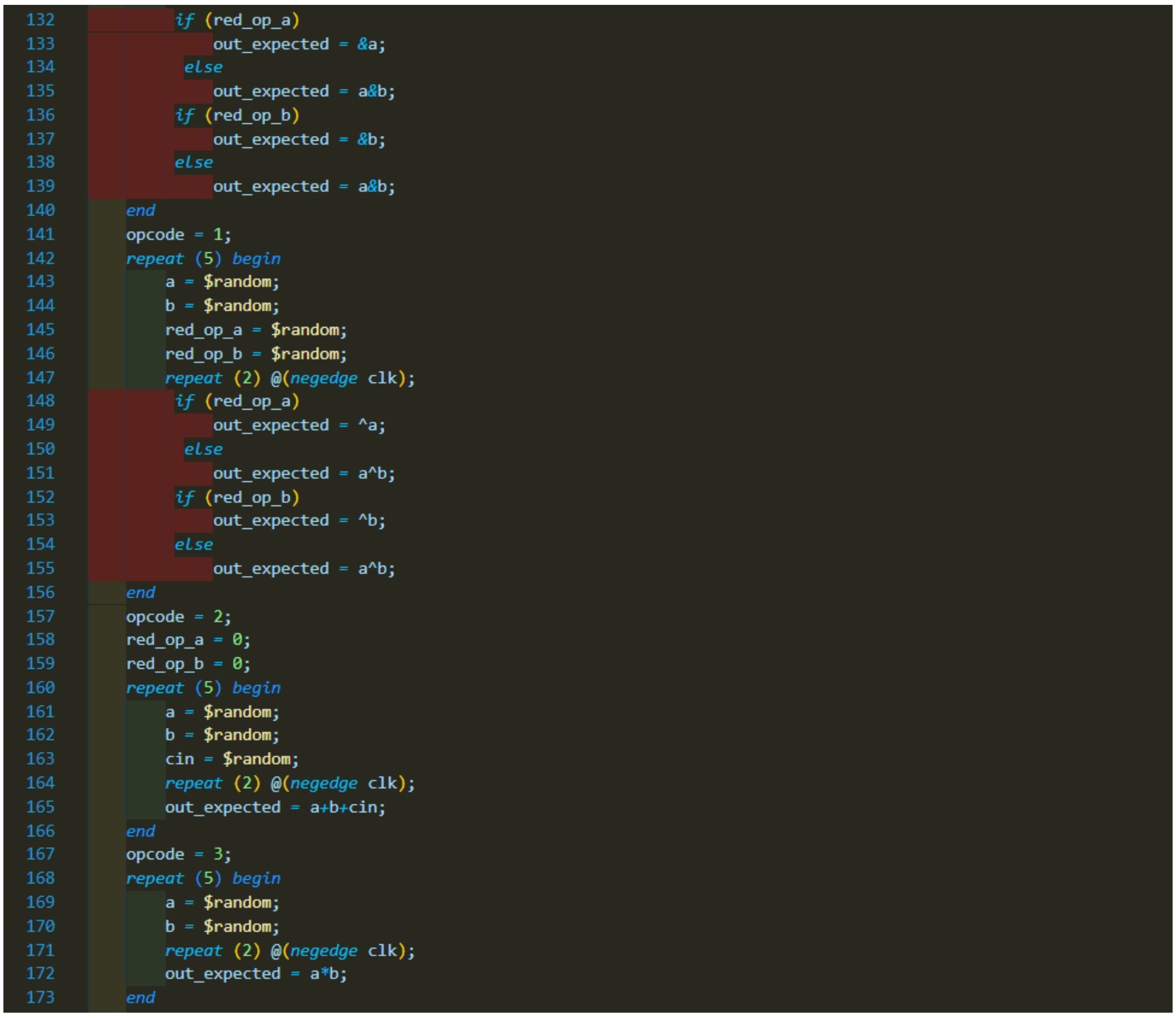
```verilog
//////////////////////question_1/////////////////////////

module ALSU #( parameter INPUT_PRIORITY = "a",
               parameter FULL_ADDER = "on")
              ( output reg [5:0] out,
               output reg [15:0] leds,
               input [2:0] a,b,opcode,
               input clk,rst,cin,serial_in,red_op_a,red_op_b,bypass_a,bypass_b,direction);

    always @(posedge clk , posedge rst) begin
        if (rst) begin
           out <= 0;
           leds <= 0;
        end
        else begin
            if (INPUT_PRIORITY == "a" && bypass_a)
                out <= a;
            else if (INPUT_PRIORITY == "b" && bypass_b)
                out <= b;
            else begin

            case (opcode)
                3'b000 : begin
                if (INPUT_PRIORITY == "a") begin
                         if (red_op_a)
                            out <= &a;
                         else
                            out <= a&b;
                end
                else if (INPUT_PRIORITY == "b") begin
                         if (red_op_b)
                            out <= &b;
                         else
                            out <= a&b;
                end
                end
                3'b001 : begin
                if (INPUT_PRIORITY == "a") begin
                         if (red_op_a)
                            out <= ^a;
                         else
                            out <= a^b;
                end
```



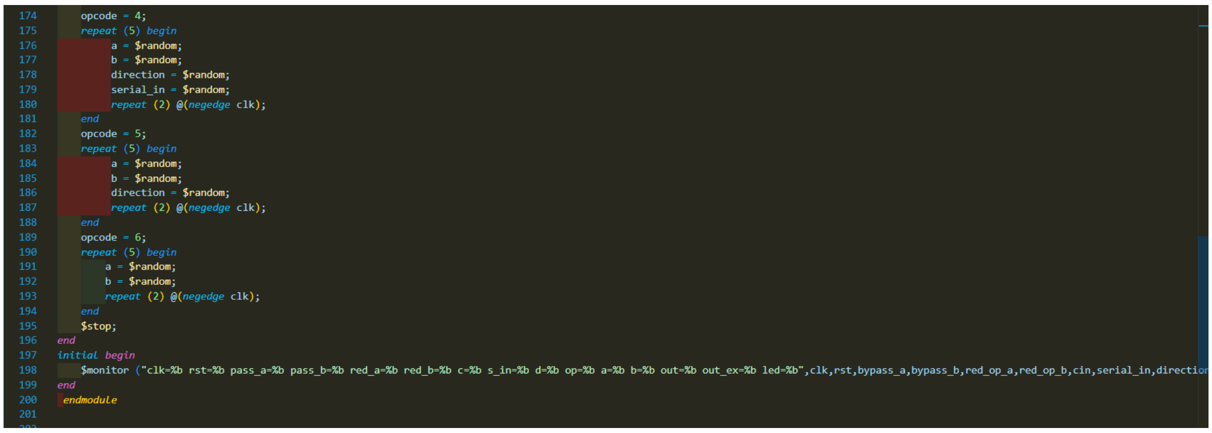
```verilog
                else if (INPUT_PRIORITY == "b") begin
                         if (red_op_b)
                            out <= ^b;
                         else
                            out <= a^b;
                end
                end
                3'b010 : begin
                         if (FULL_ADDER == "on")
                            out <= a+b+cin;
                         else if (FULL_ADDER == "off")
                            out <= a+b;
                end
                3'b011 : out <= a*b;
                3'b100 : begin
                         if (direction)
                            out <= {out[4:0],serial_in};
                         else
                         out <= {serial_in,out[5:1]};
                end
                3'b101 : begin
                         if (direction)
                            out <= {out[4:0],out[5]};
                         else
                         out <= {out[0],out[5:1]};
                end
                3'b110 : begin
                         leds <= 16'b1111111111111111;
                         out <= 0;
                end
                3'b111 : begin
                         leds <= 16'b1111111111111111;
                         out <= 0;
                end
                default: out <= 0;
            endcase
            end
        end
     end

endmodule
```



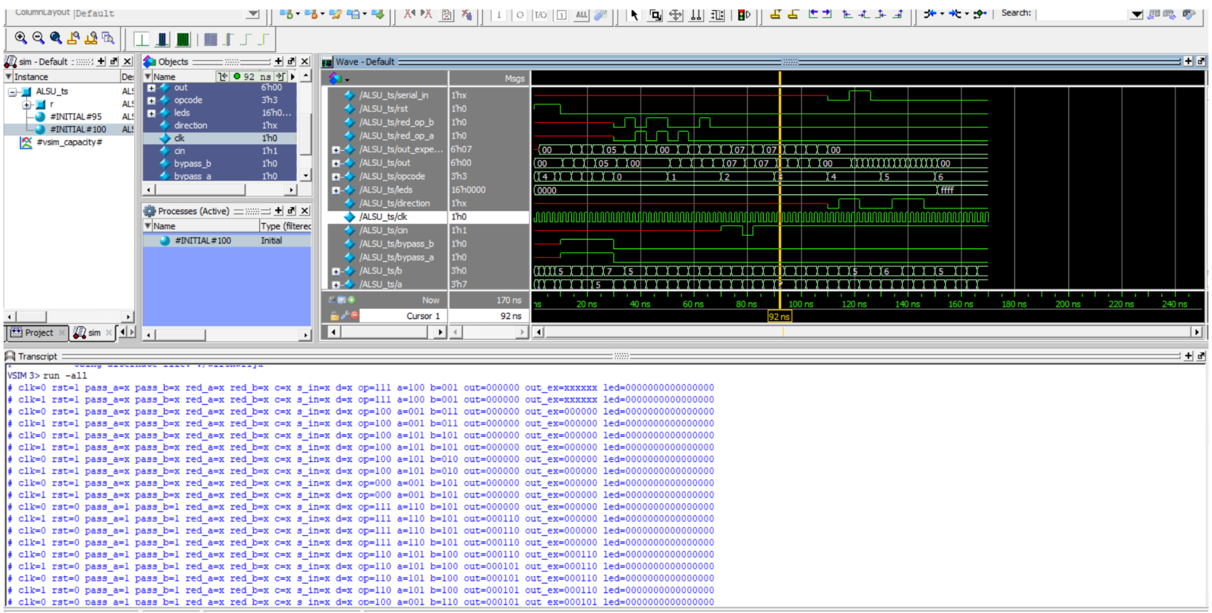
```verilog
module ALSU_ts ();
wire [5:0] out;
reg [5:0] out_expected;
wire [15:0] leds;
reg [2:0] a,b,opcode;
reg clk,rst,cin,serial_in,red_op_a,red_op_b,bypass_a,bypass_b,direction;

ALSU r (out,leds,a,b,opcode,clk,rst,cin,serial_in,red_op_a,red_op_b,bypass_a,bypass_b,direction);

initial begin
    clk = 0;
    forever #1 clk = ~clk;
end

initial begin
    rst = 1;
    repeat (5) begin
        a = $random;
        b = $random;
        opcode = $urandom_range(0,7);
        @(negedge clk);
        if (rst)
           out_expected = 0;
    end
    rst = 0;
    bypass_a =1;
    bypass_b =1;
    repeat (5) begin
        a = $random;
        b = $random;
        opcode = $urandom_range(0,7);
        repeat (2) @(negedge clk);
        if (bypass_a)
           out_expected = a;
        else if (bypass_b)
           out_expected = b;
    end
    bypass_a =0;
    bypass_b =0;
    opcode =0;
    repeat (5) begin
        a = $random;
        b = $random;
        red_op_a = $random;
        red_op_b = $random;
        repeat (2) @(negedge clk);
```



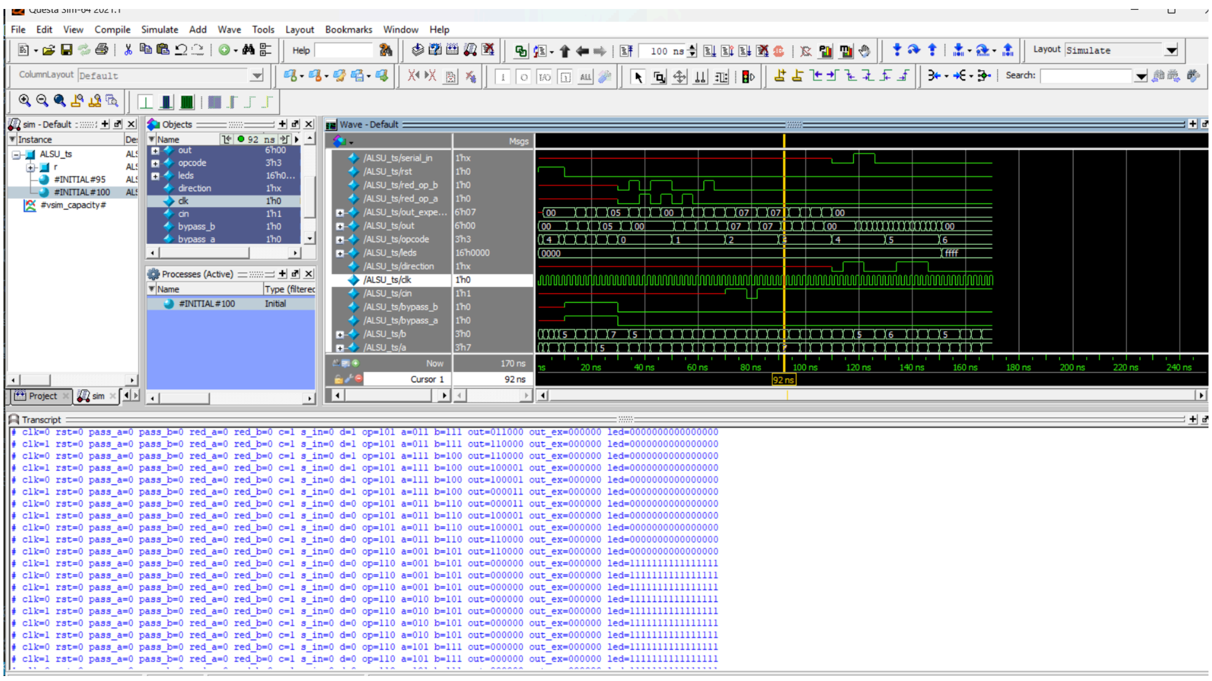
```
            if (red_op_a)
                out_expected = &a;
             else
                out_expected = a&b;
            if (red_op_b)
                out_expected = &b;
            else
                out_expected = a&b;
        end
        opcode = 1;
        repeat (5) begin
            a = $random;
            b = $random;
            red_op_a = $random;
            red_op_b = $random;
            repeat (2) @(negedge clk);
            if (red_op_a)
                out_expected = ^a;
             else
                out_expected = a^b;
            if (red_op_b)
                out_expected = ^b;
            else
                out_expected = a^b;
        end
        opcode = 2;
        red_op_a = 0;
        red_op_b = 0;
        repeat (5) begin
            a = $random;
            b = $random;
            cin = $random;
            repeat (2) @(negedge clk);
            out_expected = a+b+cin;
        end
        opcode = 3;
        repeat (5) begin
            a = $random;
            b = $random;
            repeat (2) @(negedge clk);
            out_expected = a*b;
        end
```



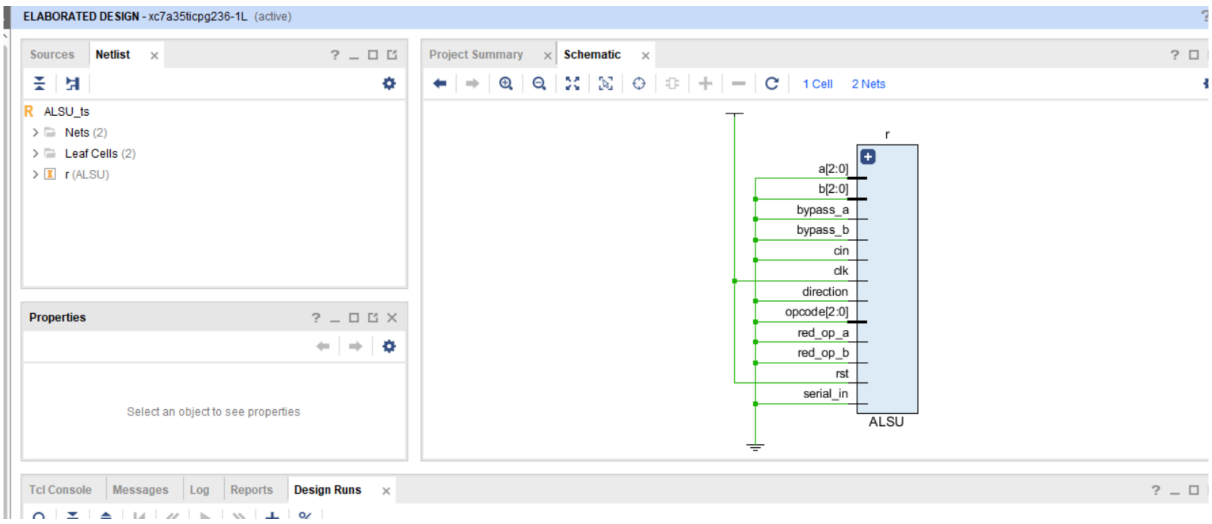
```
        opcode = 4;
        repeat (5) begin
            a = $random;
            b = $random;
            direction = $random;
            serial_in = $random;
            repeat (2) @(negedge clk);
        end
        opcode = 5;
        repeat (5) begin
            a = $random;
            b = $random;
            direction = $random;
            repeat (2) @(negedge clk);
        end
        opcode = 6;
        repeat (5) begin
           a = $random;
           b = $random;
           repeat (2) @(negedge clk);
        end
        $stop;
    end
    initial begin
        $monitor ("clk=%b rst=%b pass_a=%b pass_b=%b red_a=%b red_b=%b c=%b s_in=%b d=%b op=%b a=%b b=%b out=%b out_ex=%b led=%b",clk,rst,bypass_a,bypass_b,red_op_a,red_op_b,cin,serial_in,directio
    end
     endmodule
```

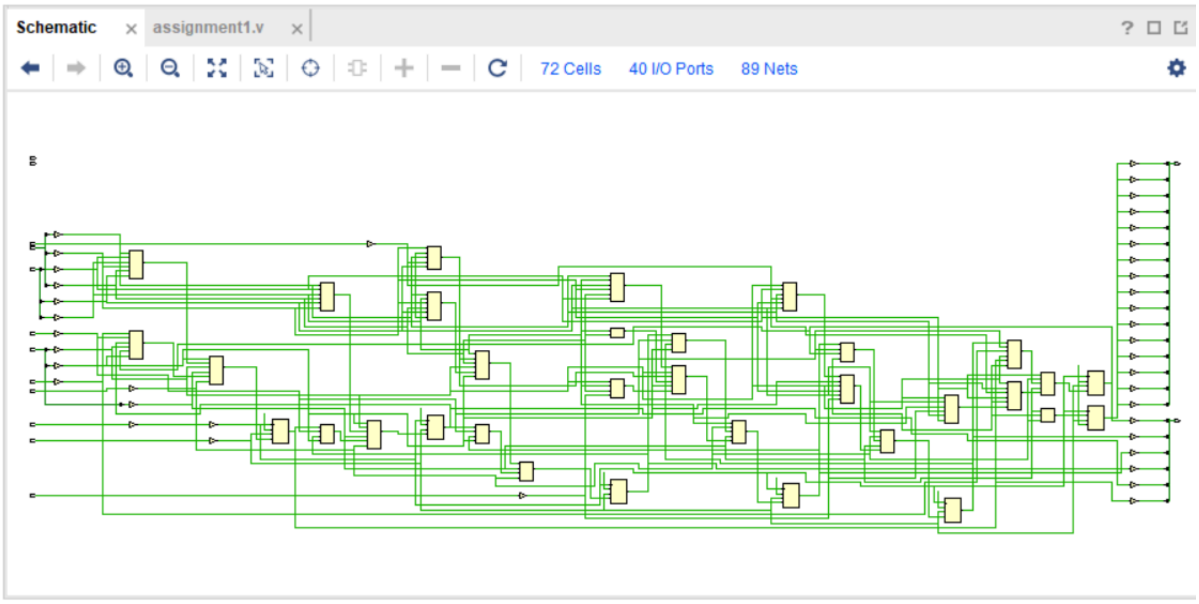


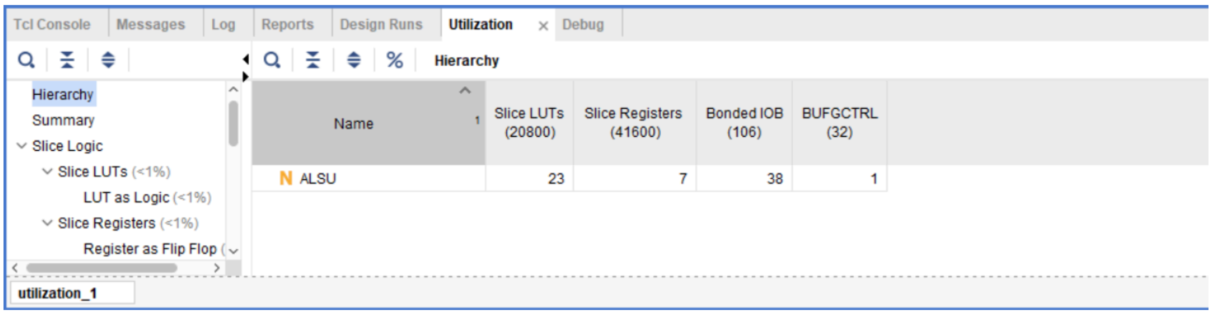
```
VSIM 3> run -all
# clk=0 rst=1 pass_a=x pass_b=x red_a=x red_b=x c=x s_in=x d=x op=111 a=100 b=001 out=000000 out_ex=xxxxxx led=0000000000000000
# clk=1 rst=1 pass_a=x pass_b=x red_a=x red_b=x c=x s_in=x d=x op=111 a=100 b=001 out=000000 out_ex=xxxxxx led=0000000000000000
# clk=0 rst=1 pass_a=x pass_b=x red_a=x red_b=x c=x s_in=x d=x op=100 a=001 b=011 out=000000 out_ex=000000 led=0000000000000000
# clk=1 rst=1 pass_a=x pass_b=x red_a=x red_b=x c=x s_in=x d=x op=100 a=001 b=011 out=000000 out_ex=000000 led=0000000000000000
# clk=0 rst=1 pass_a=x pass_b=x red_a=x red_b=x c=x s_in=x d=x op=100 a=101 b=101 out=000000 out_ex=000000 led=0000000000000000
# clk=1 rst=1 pass_a=x pass_b=x red_a=x red_b=x c=x s_in=x d=x op=100 a=101 b=101 out=000000 out_ex=000000 led=0000000000000000
# clk=0 rst=1 pass_a=x pass_b=x red_a=x red_b=x c=x s_in=x d=x op=100 a=101 b=010 out=000000 out_ex=000000 led=0000000000000000
# clk=1 rst=1 pass_a=x pass_b=x red_a=x red_b=x c=x s_in=x d=x op=100 a=101 b=010 out=000000 out_ex=000000 led=0000000000000000
# clk=0 rst=1 pass_a=x pass_b=x red_a=x red_b=x c=x s_in=x d=x op=000 a=001 b=101 out=000000 out_ex=000000 led=0000000000000000
# clk=1 rst=1 pass_a=x pass_b=x red_a=x red_b=x c=x s_in=x d=x op=000 a=001 b=101 out=000000 out_ex=000000 led=0000000000000000
# clk=0 rst=0 pass_a=1 pass_b=1 red_a=x red_b=x c=x s_in=x d=x op=111 a=110 b=101 out=000000 out_ex=000000 led=0000000000000000
# clk=1 rst=0 pass_a=1 pass_b=1 red_a=x red_b=x c=x s_in=x d=x op=111 a=110 b=101 out=000110 out_ex=000000 led=0000000000000000
# clk=0 rst=0 pass_a=1 pass_b=1 red_a=x red_b=x c=x s_in=x d=x op=111 a=110 b=101 out=000110 out_ex=000000 led=0000000000000000
# clk=1 rst=0 pass_a=1 pass_b=1 red_a=x red_b=x c=x s_in=x d=x op=111 a=110 b=101 out=000110 out_ex=000000 led=0000000000000000
# clk=0 rst=0 pass_a=1 pass_b=1 red_a=x red_b=x c=x s_in=x d=x op=110 a=101 b=100 out=000110 out_ex=000110 led=0000000000000000
# clk=1 rst=0 pass_a=1 pass_b=1 red_a=x red_b=x c=x s_in=x d=x op=110 a=101 b=100 out=000101 out_ex=000110 led=0000000000000000
# clk=0 rst=0 pass_a=1 pass_b=1 red_a=x red_b=x c=x s_in=x d=x op=110 a=101 b=100 out=000101 out_ex=000110 led=0000000000000000
# clk=1 rst=0 pass_a=1 pass_b=1 red_a=x red_b=x c=x s_in=x d=x op=110 a=101 b=100 out=000101 out_ex=000110 led=0000000000000000
# clk=0 rst=0 pass_a=1 pass_b=1 red_a=x red_b=x c=x s_in=x d=x op=100 a=001 b=110 out=000101 out_ex=000101 led=0000000000000000
```



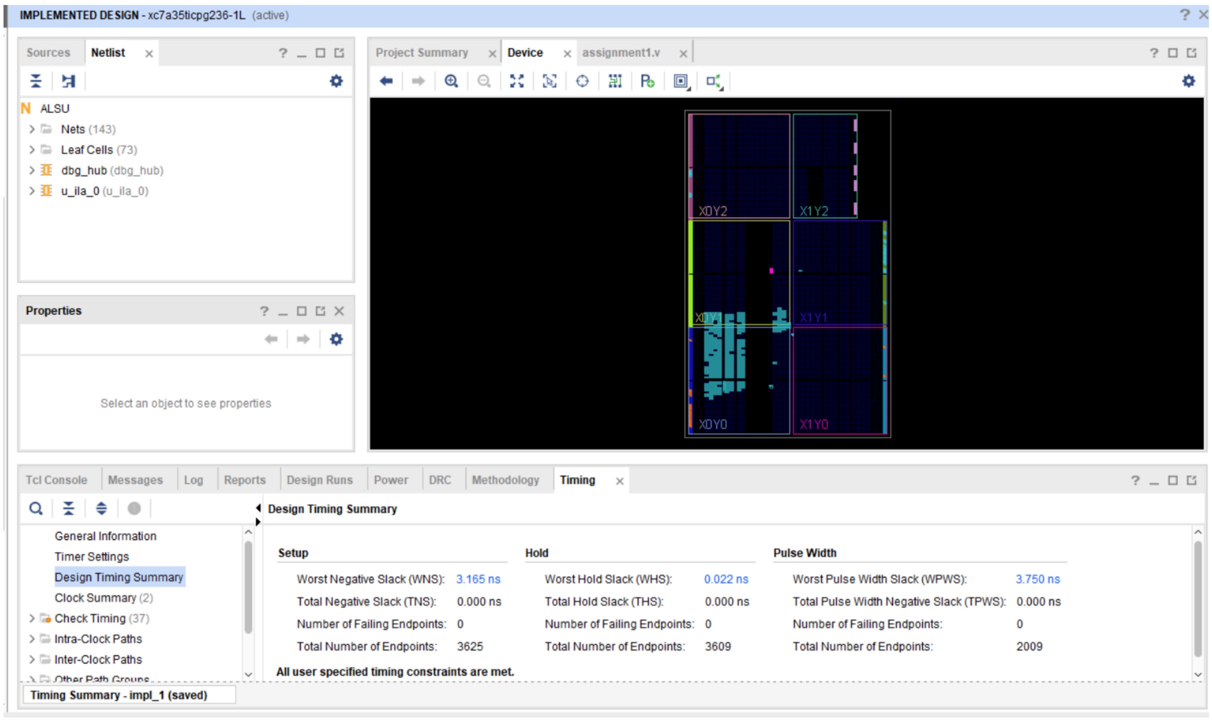
```
# clk=0 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=1 op=101 a=011 b=111 out=011000 out_ex=000000 led=0000000000000000
# clk=1 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=1 op=101 a=011 b=111 out=110000 out_ex=000000 led=0000000000000000
# clk=0 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=1 op=101 a=111 b=100 out=110000 out_ex=000000 led=0000000000000000
# clk=1 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=1 op=101 a=111 b=100 out=100001 out_ex=000000 led=0000000000000000
# clk=0 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=1 op=101 a=111 b=100 out=100001 out_ex=000000 led=0000000000000000
# clk=1 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=1 op=101 a=111 b=100 out=000011 out_ex=000000 led=0000000000000000
# clk=0 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=0 op=101 a=011 b=110 out=000011 out_ex=000000 led=0000000000000000
# clk=1 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=0 op=101 a=011 b=110 out=100001 out_ex=000000 led=0000000000000000
# clk=0 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=0 op=101 a=011 b=110 out=100001 out_ex=000000 led=0000000000000000
# clk=1 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=0 op=101 a=011 b=110 out=110000 out_ex=000000 led=0000000000000000
# clk=0 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=0 op=110 a=001 b=101 out=110000 out_ex=000000 led=0000000000000000
# clk=1 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=0 op=110 a=001 b=101 out=000000 out_ex=000000 led=1111111111111111
# clk=0 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=0 op=110 a=001 b=101 out=000000 out_ex=000000 led=1111111111111111
# clk=1 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=0 op=110 a=001 b=101 out=000000 out_ex=000000 led=1111111111111111
# clk=0 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=0 op=110 a=010 b=101 out=000000 out_ex=000000 led=1111111111111111
# clk=1 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=0 op=110 a=010 b=101 out=000000 out_ex=000000 led=1111111111111111
# clk=0 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=0 op=110 a=010 b=101 out=000000 out_ex=000000 led=1111111111111111
# clk=1 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=0 op=110 a=010 b=101 out=000000 out_ex=000000 led=1111111111111111
# clk=0 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=0 op=110 a=101 b=111 out=000000 out_ex=000000 led=1111111111111111
# clk=1 rst=0 pass_a=0 pass_b=0 red_a=0 red_b=0 c=1 s_in=0 d=0 op=110 a=101 b=111 out=000000 out_ex=000000 led=1111111111111111
```


ELABORATED DESIGN - xc7a35ticpg236-1L (active)





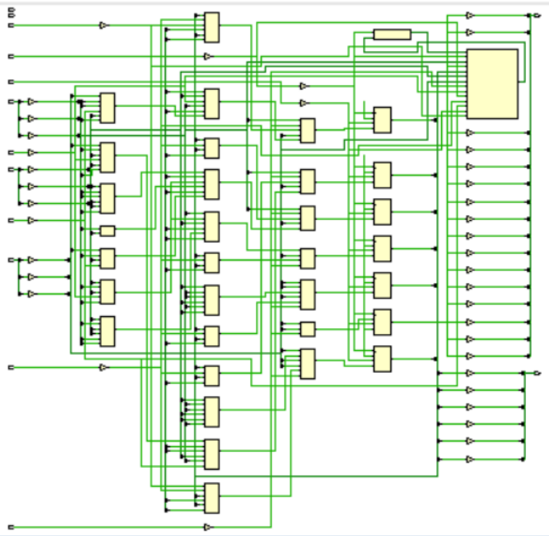
Name	Slice LUTs (20800)	Slice Registers (41600)	Bonded IOB (106)	BUFGCTRL (32)
ALSU	23	7	38	1



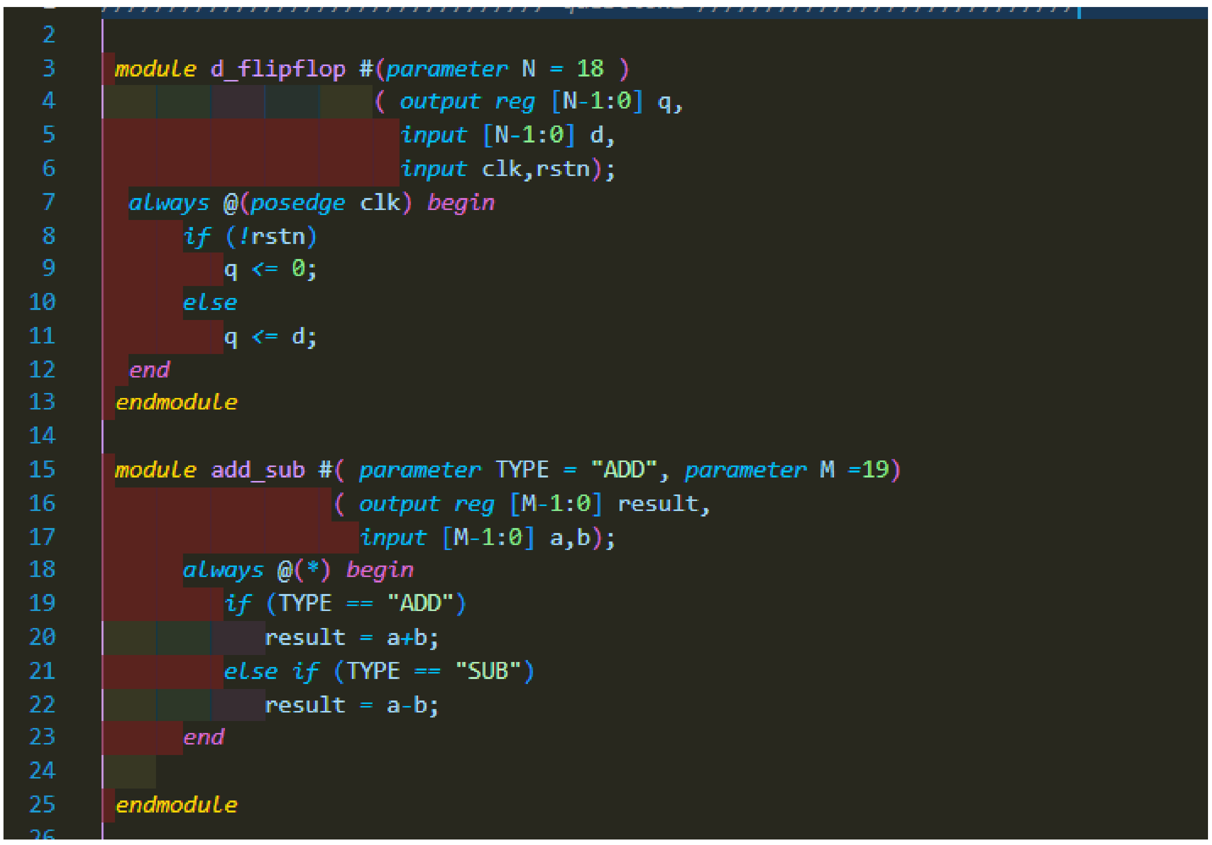
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.165 ns	Worst Hold Slack (WHS):	0.022 ns	Worst Pulse Width Slack (WPWS):	3.750 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	3625	Total Number of Endpoints:	3609	Total Number of Endpoints:	2009

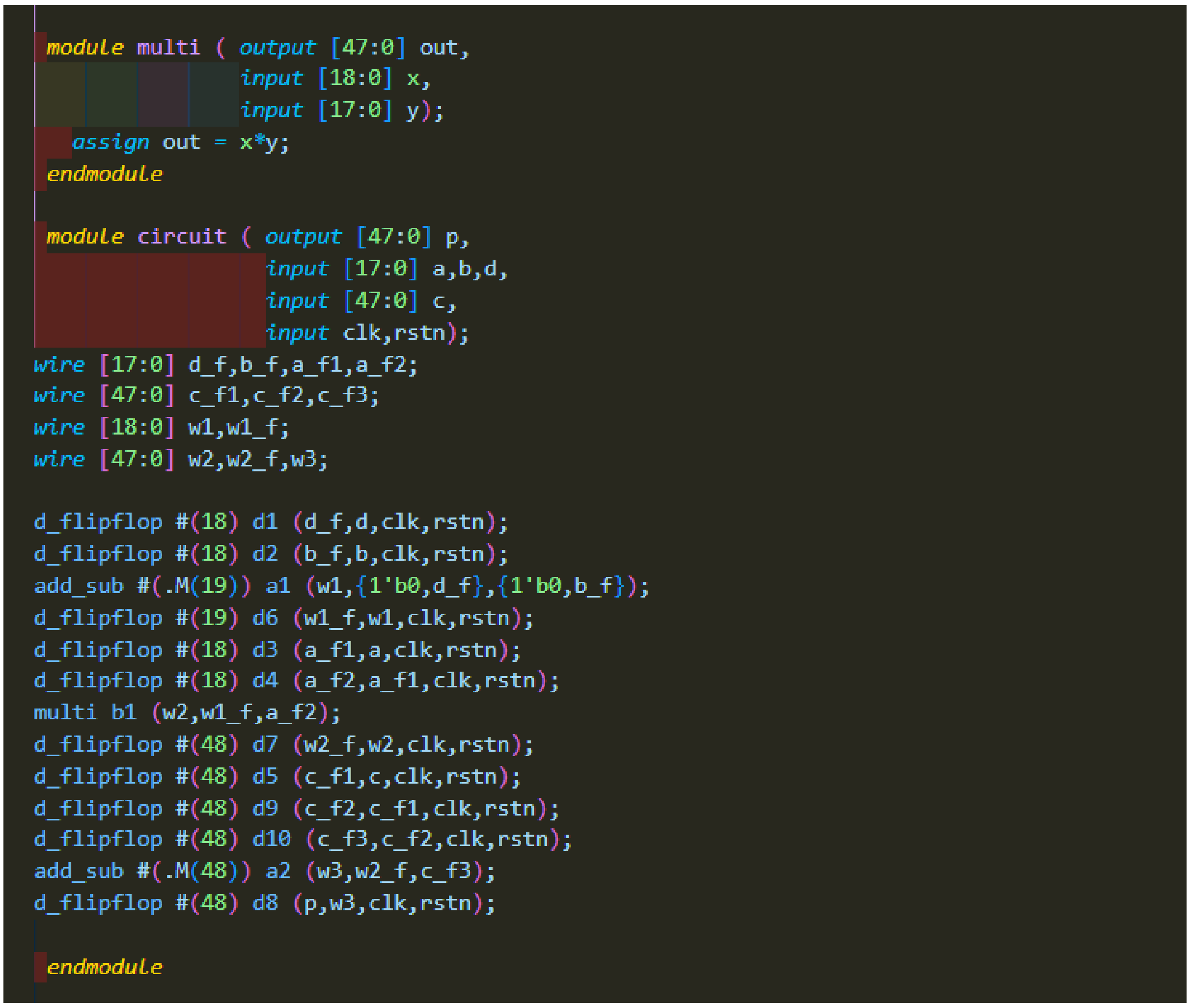
All user specified timing constraints are met.



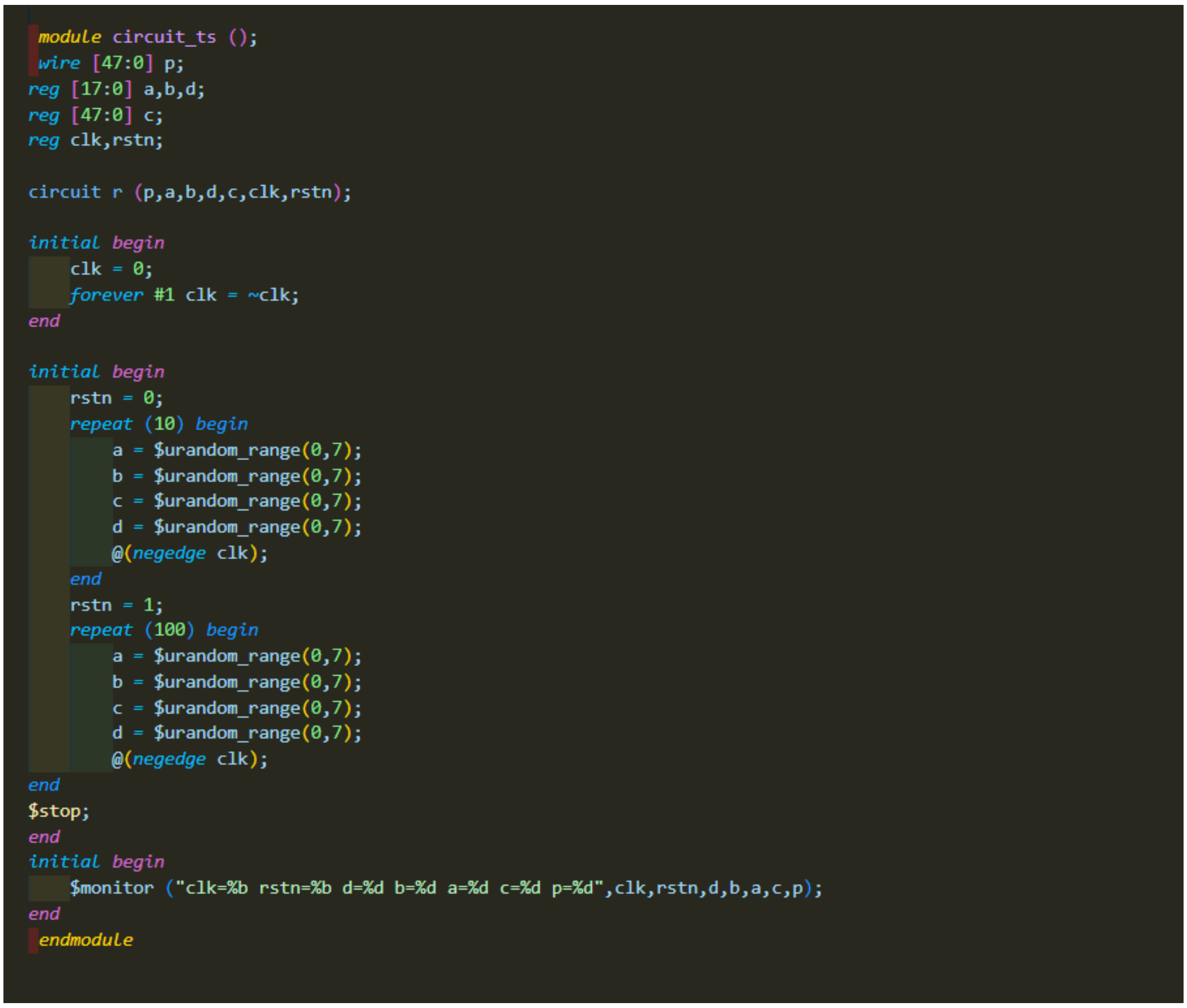
q2



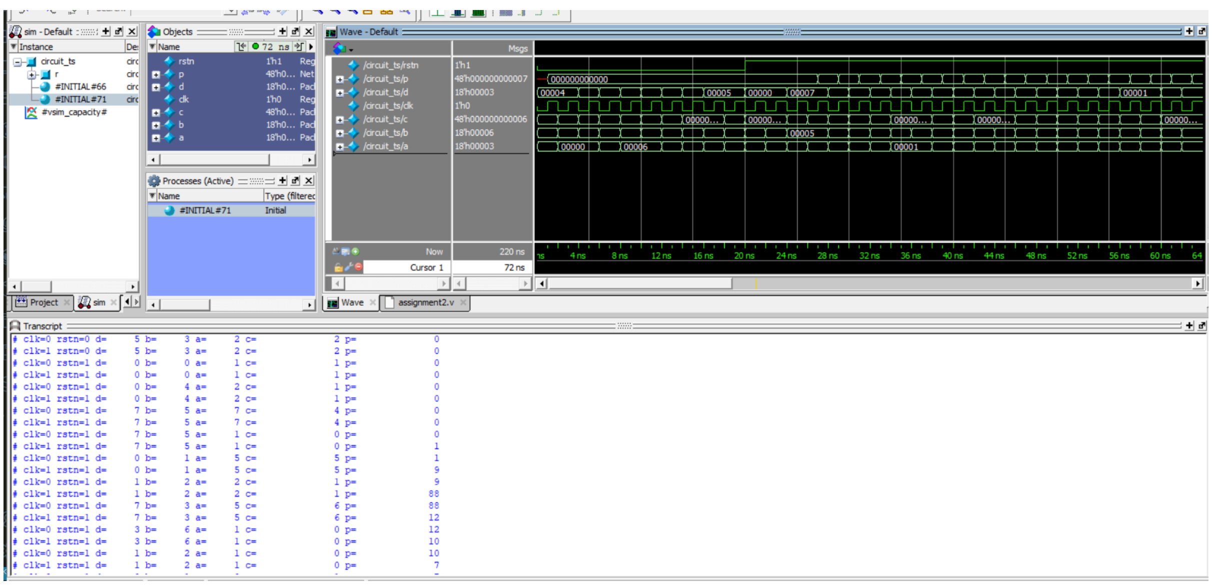
```
module d_flipflop #(parameter N = 18 )
                    ( output reg [N-1:0] q,
                      input [N-1:0] d,
                      input clk,rstn);
  always @(posedge clk) begin
      if (!rstn)
         q <= 0;
      else
         q <= d;
  end
endmodule

module add_sub #( parameter TYPE = "ADD", parameter M =19)
                 ( output reg [M-1:0] result,
                   input [M-1:0] a,b);
      always @(*) begin
         if (TYPE == "ADD")
            result = a+b;
         else if (TYPE == "SUB")
            result = a-b;
      end

endmodule
```



```
module multi ( output [47:0] out,
               input [18:0] x,
               input [17:0] y);
  assign out = x*y;
endmodule

module circuit ( output [47:0] p,
                 input [17:0] a,b,d,
                 input [47:0] c,
                 input clk,rstn);
wire [17:0] d_f,b_f,a_f1,a_f2;
wire [47:0] c_f1,c_f2,c_f3;
wire [18:0] w1,w1_f;
wire [47:0] w2,w2_f,w3;

d_flipflop #(18) d1 (d_f,d,clk,rstn);
d_flipflop #(18) d2 (b_f,b,clk,rstn);
add_sub #(.M(19)) a1 (w1,{1'b0,d_f},{1'b0,b_f});
d_flipflop #(19) d6 (w1_f,w1,clk,rstn);
d_flipflop #(18) d3 (a_f1,a,clk,rstn);
d_flipflop #(18) d4 (a_f2,a_f1,clk,rstn);
multi b1 (w2,w1_f,a_f2);
d_flipflop #(48) d7 (w2_f,w2,clk,rstn);
d_flipflop #(48) d5 (c_f1,c,clk,rstn);
d_flipflop #(48) d9 (c_f2,c_f1,clk,rstn);
d_flipflop #(48) d10 (c_f3,c_f2,clk,rstn);
add_sub #(.M(48)) a2 (w3,w2_f,c_f3);
d_flipflop #(48) d8 (p,w3,clk,rstn);

endmodule
```



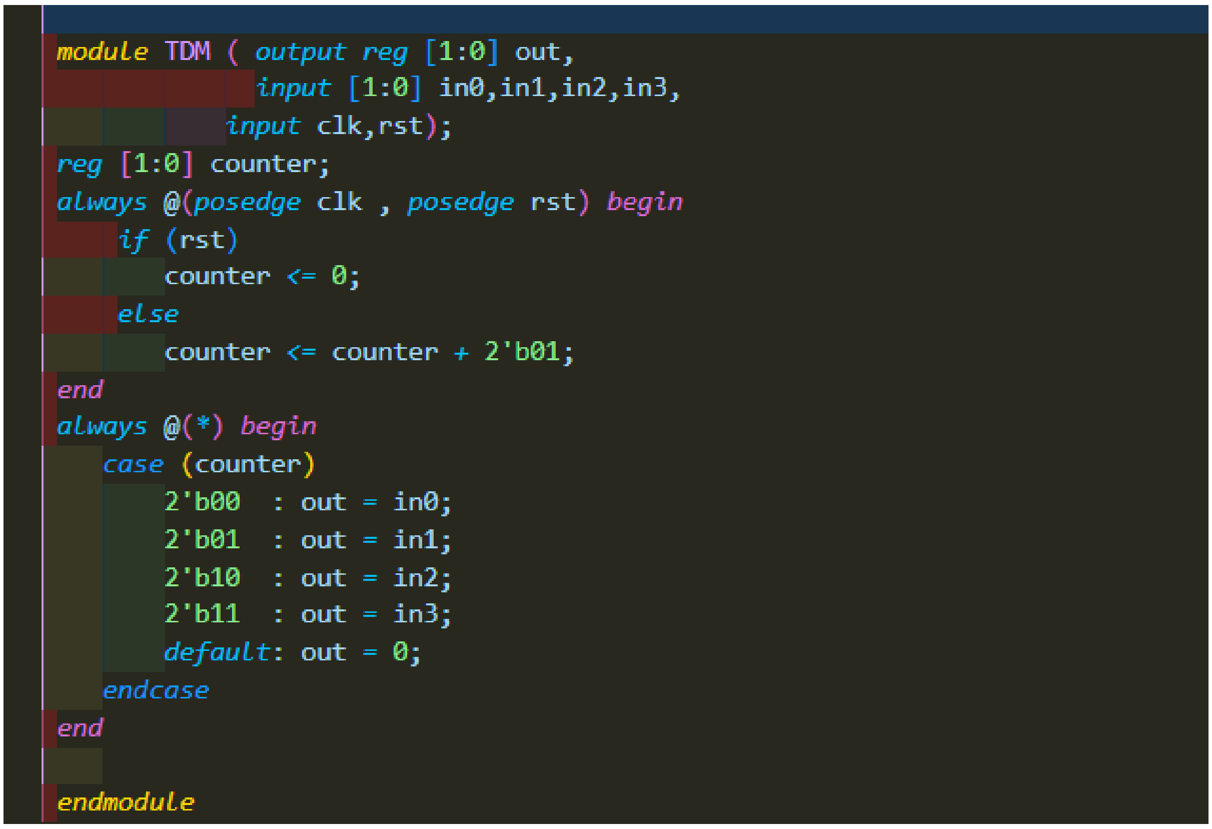
```verilog
module circuit_ts ();
wire [47:0] p;
reg [17:0] a,b,d;
reg [47:0] c;
reg clk,rstn;

circuit r (p,a,b,d,c,clk,rstn);

initial begin
    clk = 0;
    forever #1 clk = ~clk;
end

initial begin
    rstn = 0;
    repeat (10) begin
        a = $urandom_range(0,7);
        b = $urandom_range(0,7);
        c = $urandom_range(0,7);
        d = $urandom_range(0,7);
        @(negedge clk);
    end
    rstn = 1;
    repeat (100) begin
        a = $urandom_range(0,7);
        b = $urandom_range(0,7);
        c = $urandom_range(0,7);
        d = $urandom_range(0,7);
        @(negedge clk);
end
$stop;
end
initial begin
    $monitor ("clk=%b rstn=%b d=%d b=%d a=%d c=%d p=%d",clk,rstn,d,b,a,c,p);
end
endmodule
```



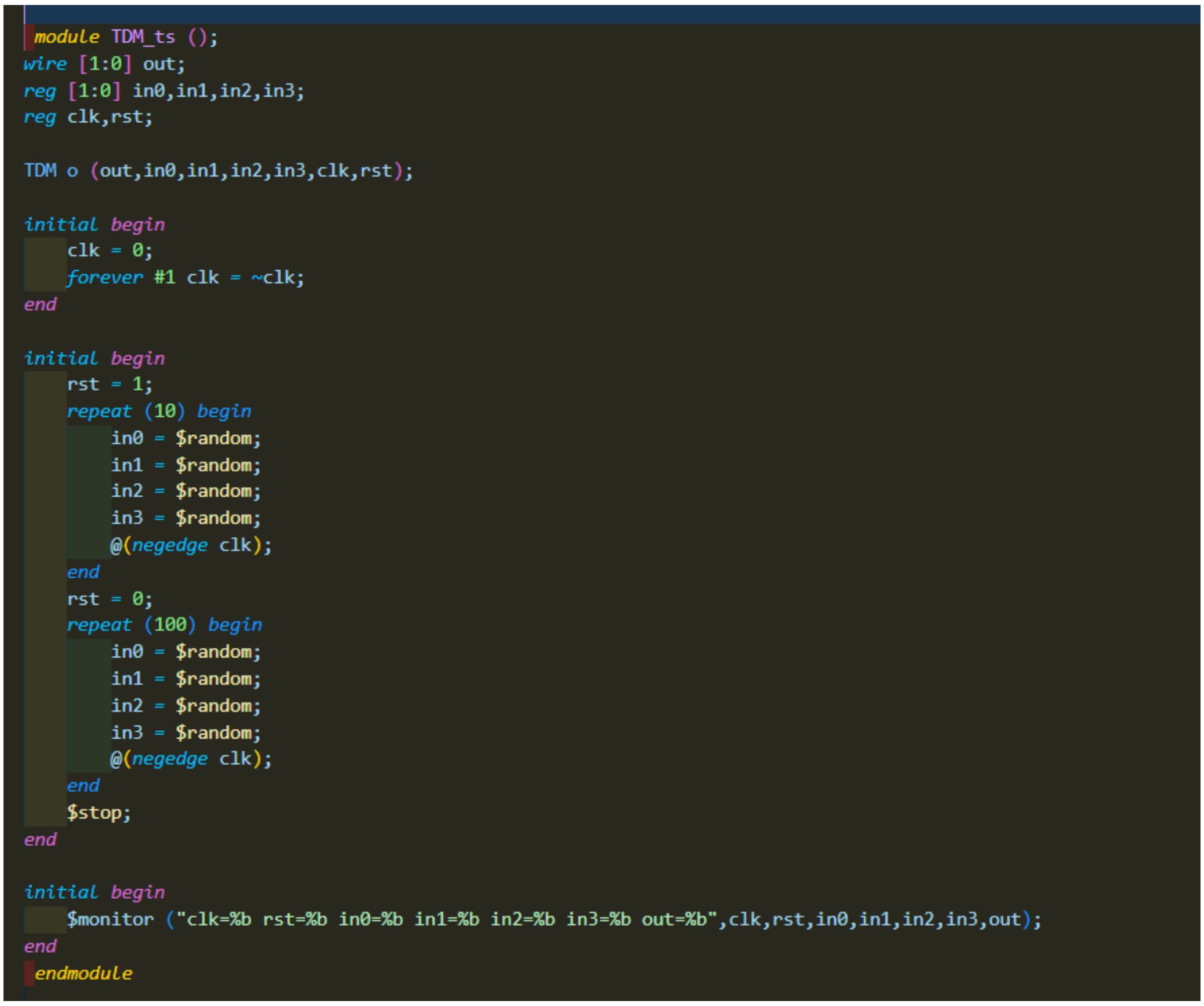
```
/circuit_ts/rstn   1'h1
/circuit_ts/p      48'h000000000007
/circuit_ts/d      18'h00003
/circuit_ts/clk    1'h0
/circuit_ts/c      48'h000000000006
/circuit_ts/b      18'h00006
/circuit_ts/a      18'h00003
Now       220 ns
Cursor 1  72 ns
```

```
Transcript
# clk=0 rstn=0 d=     5 b=     3 a=     2 c=              2 p=              0
# clk=1 rstn=0 d=     5 b=     3 a=     2 c=              2 p=              0
# clk=0 rstn=1 d=     0 b=     0 a=     1 c=              1 p=              0
# clk=1 rstn=1 d=     0 b=     0 a=     1 c=              1 p=              0
# clk=0 rstn=1 d=     0 b=     4 a=     2 c=              1 p=              0
# clk=1 rstn=1 d=     0 b=     4 a=     2 c=              1 p=              0
# clk=0 rstn=1 d=     7 b=     5 a=     7 c=              4 p=              0
# clk=1 rstn=1 d=     7 b=     5 a=     7 c=              4 p=              0
# clk=0 rstn=1 d=     7 b=     5 a=     1 c=              0 p=              0
# clk=1 rstn=1 d=     7 b=     5 a=     1 c=              0 p=              1
# clk=0 rstn=1 d=     0 b=     1 a=     5 c=              5 p=              1
# clk=1 rstn=1 d=     0 b=     1 a=     5 c=              5 p=              9
# clk=0 rstn=1 d=     1 b=     2 a=     2 c=              1 p=              9
# clk=1 rstn=1 d=     1 b=     2 a=     2 c=              1 p=             88
# clk=0 rstn=1 d=     7 b=     3 a=     5 c=              6 p=             88
# clk=1 rstn=1 d=     7 b=     3 a=     5 c=              6 p=             12
# clk=0 rstn=1 d=     3 b=     6 a=     1 c=              0 p=             12
# clk=1 rstn=1 d=     3 b=     6 a=     1 c=              0 p=             10
# clk=0 rstn=1 d=     1 b=     2 a=     1 c=              0 p=             10
# clk=1 rstn=1 d=     1 b=     2 a=     1 c=              0 p=              7
```

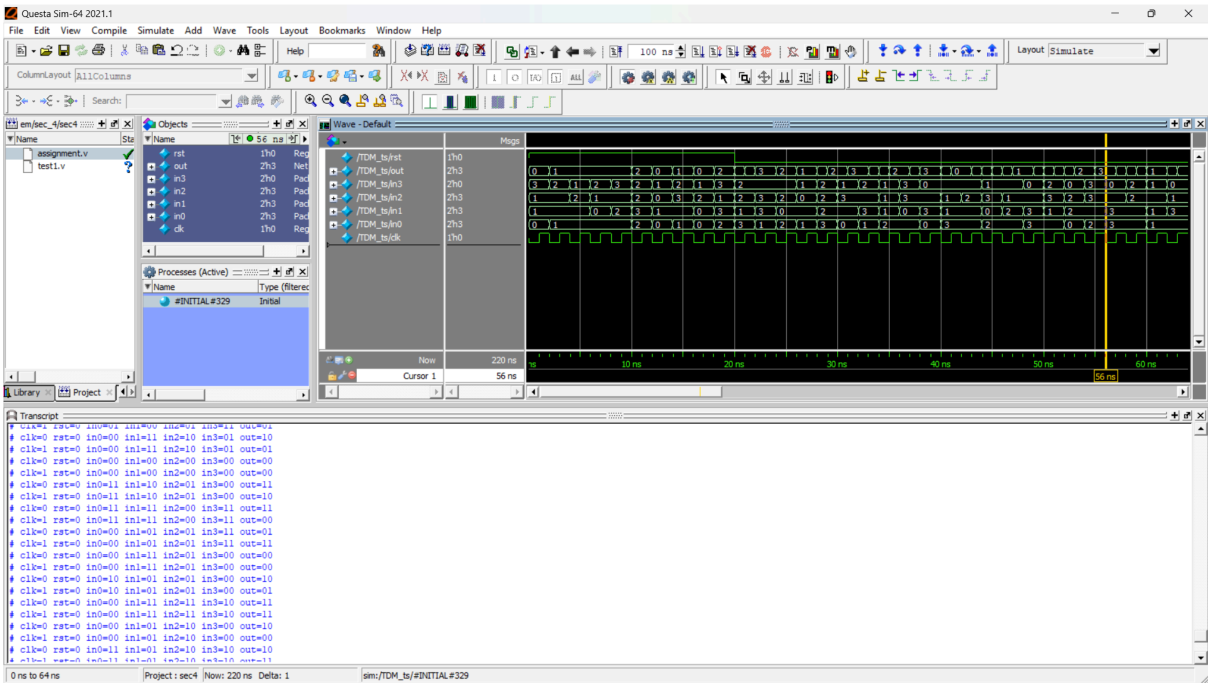
q3



```verilog
module TDM ( output reg [1:0] out,
                 input [1:0] in0,in1,in2,in3,
                 input clk,rst);
reg [1:0] counter;
always @(posedge clk , posedge rst) begin
    if (rst)
        counter <= 0;
    else
        counter <= counter + 2'b01;
end
always @(*) begin
    case (counter)
        2'b00  : out = in0;
        2'b01  : out = in1;
        2'b10  : out = in2;
        2'b11  : out = in3;
        default: out = 0;
    endcase
end

endmodule
```

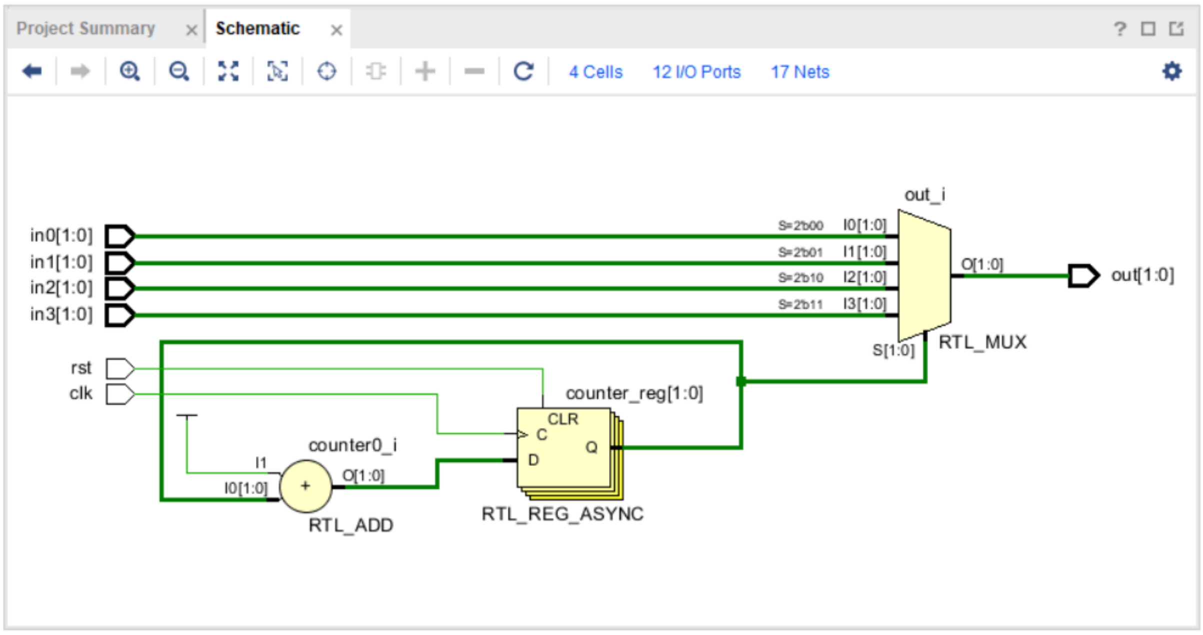


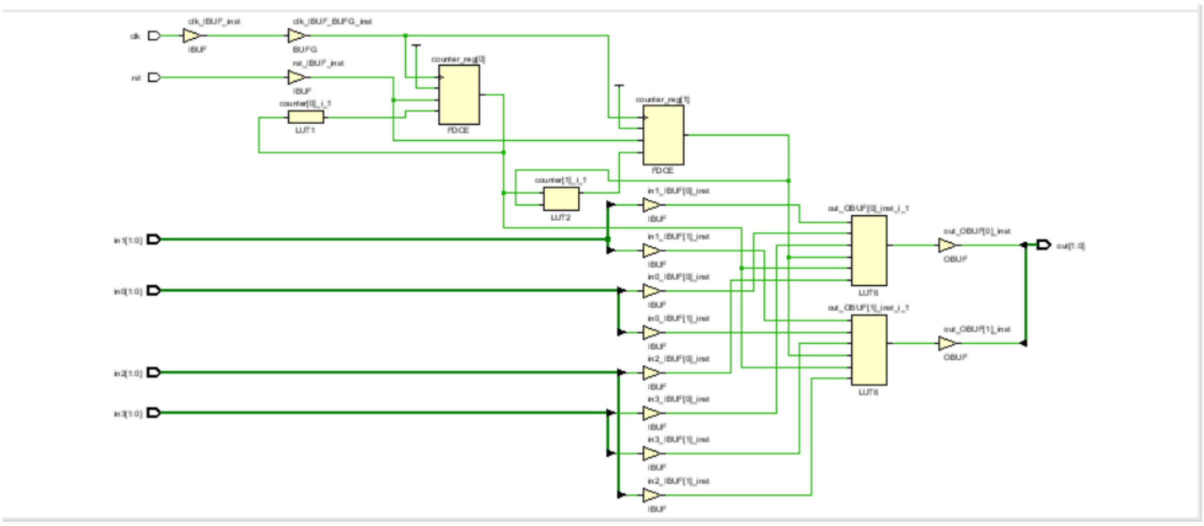
```verilog
module TDM_ts ();
wire [1:0] out;
reg [1:0] in0,in1,in2,in3;
reg clk,rst;

TDM o (out,in0,in1,in2,in3,clk,rst);

initial begin
    clk = 0;
    forever #1 clk = ~clk;
end

initial begin
    rst = 1;
    repeat (10) begin
        in0 = $random;
        in1 = $random;
        in2 = $random;
        in3 = $random;
        @(negedge clk);
    end
    rst = 0;
    repeat (100) begin
        in0 = $random;
        in1 = $random;
        in2 = $random;
        in3 = $random;
        @(negedge clk);
    end
    $stop;
end

initial begin
    $monitor ("clk=%b rst=%b in0=%b in1=%b in2=%b in3=%b out=%b",clk,rst,in0,in1,in2,in3,out);
end
endmodule
```

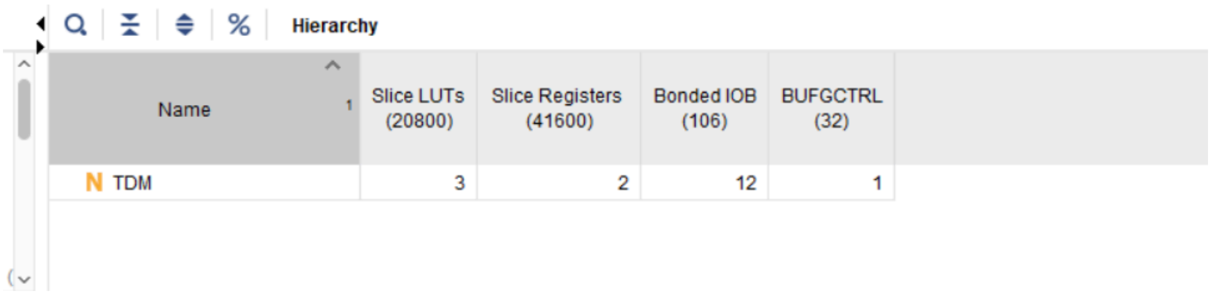


```
# clk=1 rst=0 in0=01 in1=00 in2=01 in3=11 out=01
# clk=0 rst=0 in0=00 in1=11 in2=10 in3=01 out=10
# clk=1 rst=0 in0=00 in1=11 in2=10 in3=01 out=01
# clk=0 rst=0 in0=00 in1=00 in2=00 in3=00 out=00
# clk=1 rst=0 in0=00 in1=00 in2=00 in3=00 out=00
# clk=0 rst=0 in0=11 in1=10 in2=01 in3=00 out=11
# clk=1 rst=0 in0=11 in1=10 in2=01 in3=00 out=10
# clk=0 rst=0 in0=11 in1=11 in2=00 in3=11 out=11
# clk=1 rst=0 in0=11 in1=11 in2=00 in3=11 out=00
# clk=0 rst=0 in0=00 in1=01 in2=01 in3=11 out=01
# clk=1 rst=0 in0=00 in1=01 in2=01 in3=11 out=11
# clk=0 rst=0 in0=00 in1=11 in2=01 in3=00 out=00
# clk=1 rst=0 in0=00 in1=11 in2=01 in3=00 out=00
# clk=0 rst=0 in0=10 in1=01 in2=01 in3=00 out=10
# clk=1 rst=0 in0=10 in1=01 in2=01 in3=00 out=01
# clk=0 rst=0 in0=00 in1=11 in2=11 in3=10 out=11
# clk=1 rst=0 in0=00 in1=11 in2=11 in3=10 out=11
# clk=0 rst=0 in0=00 in1=01 in2=10 in3=00 out=10
# clk=1 rst=0 in0=00 in1=01 in2=10 in3=00 out=00
# clk=0 rst=0 in0=11 in1=01 in2=10 in3=10 out=10
# clk=1 rst=0 in0=11 in1=01 in2=10 in3=10 out=11
```

0 ns to 64 ns | Project : sec4 | Now: 220 ns Delta: 1 | sim:/TDM_ts/#INITIAL#329

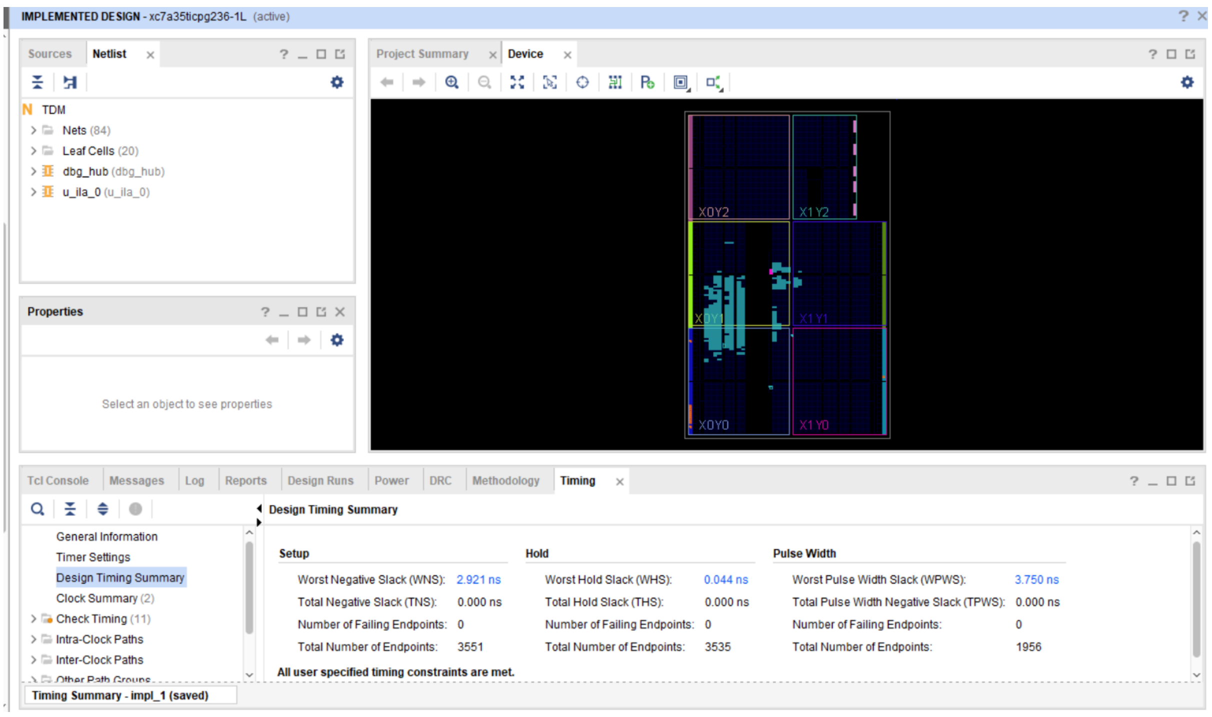






Hierarchy

Name	Slice LUTs (20800)	Slice Registers (41600)	Bonded IOB (106)	BUFGCTRL (32)
N TDM	3	2	12	1



IMPLEMENTED DESIGN - xc7a35ticpg236-1L (active)

N TDM
- Nets (84)
- Leaf Cells (20)
- dbg_hub (dbg_hub)
- u_ila_0 (u_ila_0)

Select an object to see properties

- General Information
- Timer Settings
- Design Timing Summary
- Clock Summary (2)
- Check Timing (11)
- Intra-Clock Paths
- Inter-Clock Paths
- Other Path Groups

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.921 ns	Worst Hold Slack (WHS):	0.044 ns	Worst Pulse Width Slack (WPWS):	3.750 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	3551	Total Number of Endpoints:	3535	Total Number of Endpoints:	1956

All user specified timing constraints are met.

Timing Summary - impl_1 (saved)