يوسف أحمد محمد ابراهيم

Assignment 4

Module 1 :

- **RTL code :**

```
module mod1_alsu#(parameter FULL_ADDER = "ON", INPUT_PRIORITY = "A")
                 (input [2:0] a, b, op,
                 input cin, serial_in, dir, red_op_a,
                 input red_op_b, bypass_a, bypass_b,
                 input clk, rst,
                 output reg [5:0]out,
                 output reg [15:0]leds);

    reg [2:0] a_reg, b_reg, op_reg;
    reg cin_reg, serial_in_reg, dir_reg,
    reg red_op_a_reg, red_op_b_reg, bypass_a_reg, bypass_b_reg;
    reg rotate;
    reg [5:0]out_reg;

    always @(posedge clk, posedge rst) begin
        if(rst) begin
            a_reg <= 0;
            b_reg <= 0;
            op_reg <= 0;
            cin_reg <= 0;
            serial_in_reg <= 0;
            dir_reg <= 0;
            red_op_a_reg <= 0;
            red_op_b_reg <= 0;
            bypass_a_reg <= 0;
            bypass_b_reg <= 0;
            out_reg = 0;
        end
        else begin
            a_reg <= a;
            b_reg <= b;
            op_reg <= op;
            cin_reg <= cin;
            serial_in_reg <= serial_in;
            dir_reg <= dir;
            red_op_a_reg <= red_op_a;
            red_op_b_reg <= red_op_b;
            bypass_a_reg <= bypass_a;
            bypass_b_reg <= bypass_b;
        end
    end
```

```
always @(posedge clk, posedge rst) begin
    if(rst) begin
        out <= 0;
        leds <= 0;
    end
    else begin
        if(bypass_a_reg) begin
            if(bypass_b_reg) begin
                if(INPUT_PRIORITY == "A")
                    out <= a_reg;
                else
                    out <= b_reg;
            end
            else
                out <= a_reg;
        end
        else if(bypass_b_reg)
            out <= b;
        else begin
            if(red_op_a_reg) begin
                if(red_op_b_reg) begin
                    if(INPUT_PRIORITY == "A") begin
                        case (op)
                            3'b000 : out <= &a_reg;
                            3'b001 : out <= ^a_reg;
                            default : begin
                                out <= 0;
                                leds <= ~leds;
                            end
                        endcase
                    end
                    else if(INPUT_PRIORITY == "B") begin
                        case (op)
                            3'b000 : out <= &b_reg;
                            3'b001 : out <= ^b_reg;
                            default : begin
                                out <= 0;
                                leds <= ~leds;
                            end
                        endcase
                    end
                end
```

```
    else begin
        case (op)
            3'b000 : out <= &a_reg;
            3'b001 : out <= ^a_reg;
            default : begin
                out <= 0;
                leds <= ~leds;
            end
        endcase
    end
end
else begin
    if(red_op_b) begin
        case (op)
            3'b000 : out <= &b_reg;
            3'b001 : out <= ^b_reg;
            default : begin
                out <= 0;
                leds <= ~leds;
            end
            endcase
        end
        else begin
            case (op)
            3'b000 : out <= a_reg & b_reg;
            3'b001 : out <= a_reg ^ b_reg;
            3'b010 : begin
                if(FULL_ADDER == "ON")
                    out <= a_reg + b_reg + cin_reg;
                else
                    out <= a_reg + b_reg;
            end
            3'b011 : out <= a_reg * b_reg;
            3'b100 : begin
                if(dir_reg) begin
                    out_reg <= out;
                    out[5:1] <= out_reg[4:0];
                    out[0] <= serial_in_reg;
                end
                else begin
                    out_reg <= out;
                    out[4:0] <= out_reg[5:1];
                    out[5] <= serial_in_reg;
                end
```

```
                3'b101 : begin
                    if(dir_reg) begin
                        rotate = out[5];
                        out[5:1] <= out[4:0];
                        out[0] <= rotate;
                    end
                    else begin
                        rotate = out[0];
                        out[4:0] <= out[5:1];
                        out[5] <= rotate;
                    end
                end
                default : begin
                    out <= 0;
                    leds <= ~leds;
                end
            endcase
        end
    end
  end
 end
endmodule
```

- **testbench code :**

```
module mod1_alsu_tb();

    // stimuls instantiation and clk generation
    reg cin, serial_in, dir, red_op_a, red_op_b,
    reg bypass_a, bypass_b, clk, rst;
    reg [2:0] a, b, op;
    reg [5:0]out_expected;
    reg [15:0]leds_expected;
    wire [5:0]out_dut;
    wire [15:0]leds_dut;

    mod1_alsu dut(a, b, op, cin, serial_in, dir, red_op_a, red_op_b,
            bypass_a, bypass_b, clk, rst, out_dut, leds_dut);

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

```
    //1 check rst
    rst = 1;
    cin = 0;
    serial_in = 0; dir = 0;
    red_op_a = 0;
    red_op_b = 0;
    bypass_a = 0;
    bypass_b = 0;
    a = 0;
    b = 0;
    op = 0;
    leds_expected = 0;
    out_expected = 0;
    repeat(10) begin
        @(negedge clk);
        if(out_dut != out_expected || leds_dut != leds_expected)
            begin
                $display("ERROR");
                $stop;
            end
    end
    $stop;
```

```
//2 check bypass
rst = 1;
cin = 0;
serial_in = 0; dir = 0;
red_op_a = 0;
red_op_b = 0;
bypass_a = 1;
bypass_b = 1;
@(negedge clk);
rst = 0;
repeat(10) begin
    a = $random;
    b = $random;
    op = $urandom_range(5);
    @(negedge clk);
    @(negedge clk);
    leds_expected = 0;
    out_expected = a;
    if(out_dut != out_expected || leds_dut != leds_expected)
        begin
            $display("ERROR");
            $stop;
        end
end
$stop;
```

```
//3_check op = 0
rst = 1;
cin = 0;
serial_in = 0; dir = 0;
bypass_a = 0;
bypass_b = 0;
op = 0;
leds_expected = 0;
@(negedge clk);
rst = 0;
repeat(10) begin
    red_op_a = $random;
    red_op_b = $random;
    a = $random;
    b = $random;
    if(red_op_a)
        out_expected = &a;
    else if (red_op_b)
        out_expected = &b;
    else
        out_expected = a & b;
    @(negedge clk);
    @(negedge clk);
    if(out_dut != out_expected || leds_dut != leds_expected)
        begin
            $display("ERROR");
            $stop;
        end
end
$stop;
```

```
// 4_check op = 1
rst = 1;
cin = 0;
serial_in = 0; dir = 0;
bypass_a = 0;
bypass_b = 0;
op = 1;
leds_expected = 0;
@(negedge clk);
rst = 0;
repeat(10) begin
    red_op_a = $random;
    red_op_b = $random;
    a = $random;
    b = $random;
    if(red_op_a)
        out_expected = ^a;
    else if (red_op_b)
        out_expected = ^b;
    else
        out_expected = a ^ b;
    @(negedge clk);
    @(negedge clk);
    if(out_dut != out_expected || leds_dut != leds_expected)
        begin
            $display("ERROR");
            $stop;
        end
end
$stop;
```

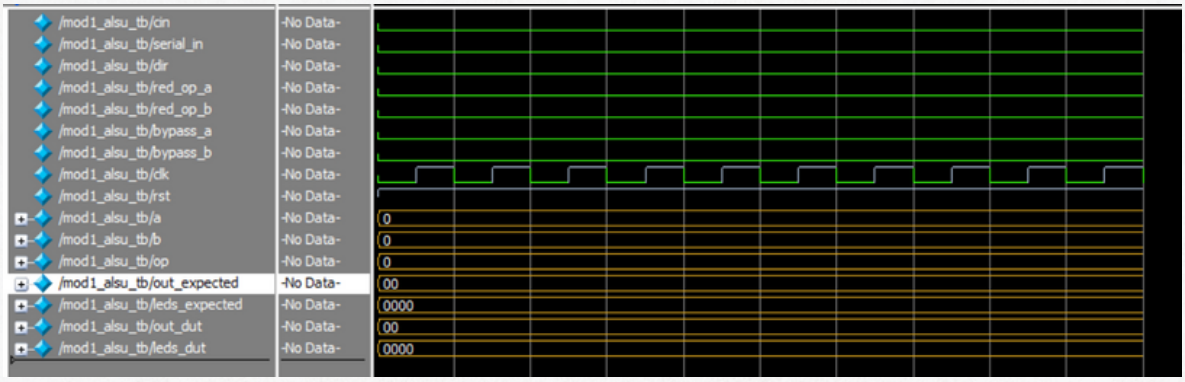
```
//5_check op = 2
rst = 1;
serial_in = 0;
dir = 0;
bypass_a = 0;
bypass_b = 0;
op = 2;
red_op_a = 0;
red_op_b = 0;
leds_expected = 0;
@(negedge clk);
rst = 0;
repeat(10) begin
    a = $random;
    b = $random;
    cin = $random;
    out_expected = a + b + cin;
    @(negedge clk);
    @(negedge clk);
    if(out_dut != out_expected || leds_dut != leds_expected)
        begin
            $display("ERROR");
            $stop;
        end
end
$stop;
```

```
//6_check op = 3
rst = 1;
serial_in = 0;
dir = 0;
bypass_a = 0;
bypass_b = 0;
op = 3;
red_op_a = 0;
red_op_b = 0;
leds_expected = 0;
cin = 0;
@(negedge clk);
rst = 0;
repeat(10) begin
    a = $random;
    b = $random;
    out_expected = a * b;
    @(negedge clk);
    @(negedge clk);
    if(out_dut != out_expected || leds_dut != leds_expected)
        begin
            $display("ERROR");
            $stop;
        end
end
$stop;
```

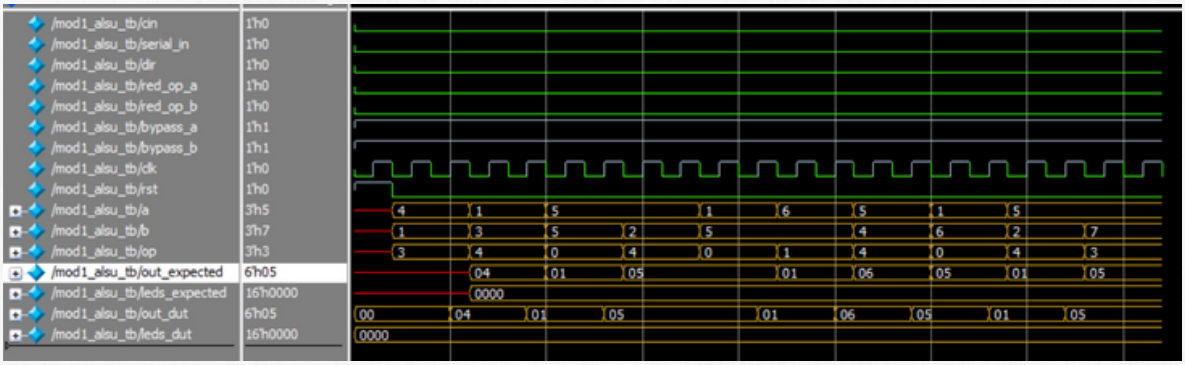
```
//7_check op = 4
rst = 1;
bypass_a = 0;
bypass_b = 0;
op = 4;
red_op_a = 0;
red_op_b = 0;
leds_expected = 0;
cin = 0;
out_expected = 0;
@(negedge clk);
rst = 0;
repeat(10) begin
    serial_in = $random;
    dir = $random;
    a = $random;
    b = $random;
    @(negedge clk);
    @(negedge clk);
    if(dir == 1) begin
        out_expected[5:1] = out_expected[4:0];
        out_expected[0] = serial_in;
    end
    else begin
        out_expected[4:0] = out_expected[5:1];
        out_expected[5] = serial_in;
    end
    if(out_dut != out_expected || leds_dut != leds_expected)
        begin
            $display("ERROR");
            $stop;
        end
end
$stop;
```

```
        //8_check_ op = 5;
        rst = 1;
        bypass_a = 0;
        bypass_b = 0;
        op = 5;
        red_op_a = 0;
        red_op_b = 0;
        leds_expected = 0;
        cin = 0;
        out_expected = 0;
        @(negedge clk);
        rst = 0;
        repeat(10) begin
            serial_in = $random;
            dir = $random;
            a = $random;
            b = $random;
            @(negedge clk);
            @(negedge clk);
            if(dir == 1) begin
                out_expected[5:1] = out_expected[4:0];
                out_expected[0] = out_expected[5];
            end
            else begin
                out_expected[4:0] = out_expected[5:1];
                out_expected[5] = out_expected[0];
            end
            if(out_dut != out_expected || leds_dut != leds_expected)
                begin
                    $display("ERROR");
                    $stop;
                end
        end
        $stop;
    end
endmodule
```

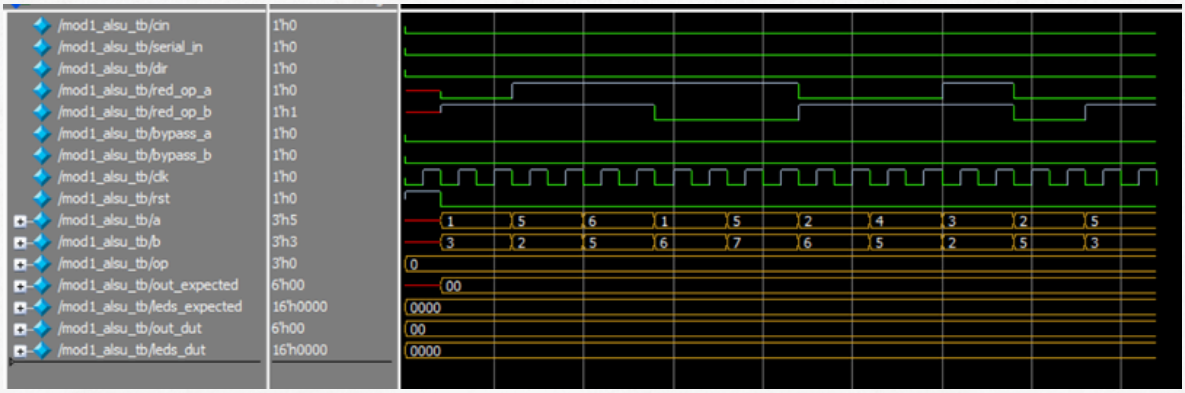
- **waveforms (questasim)**
  - test 1 (reset functionality)



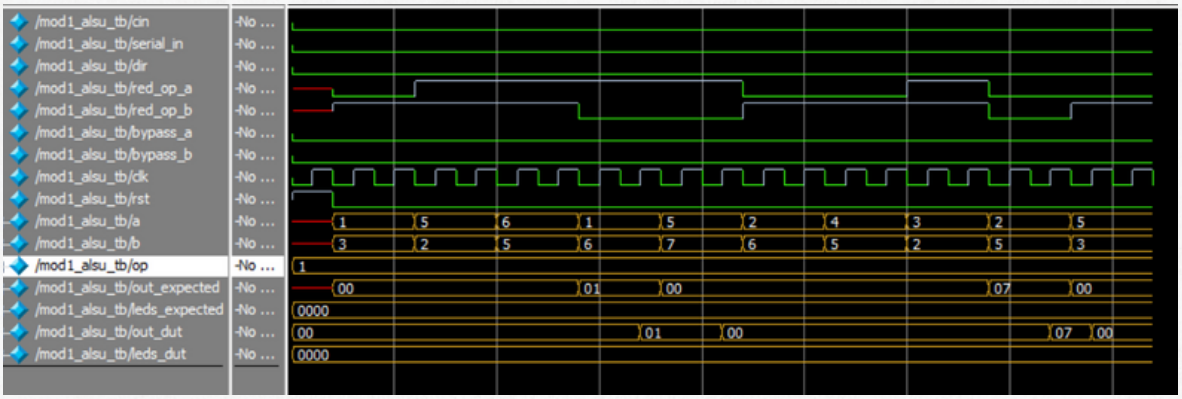
  - test 2 (bypass functionality)



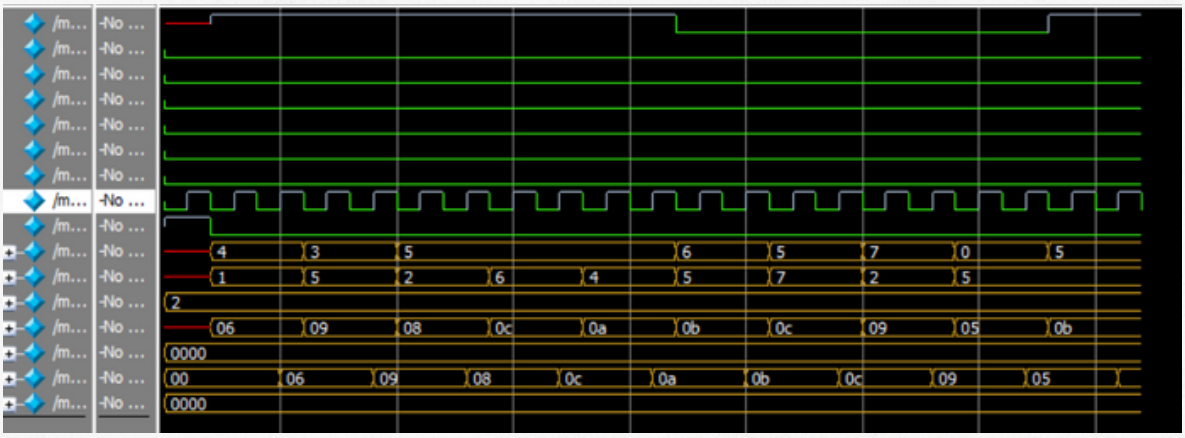
  - test 3 (op = 0 functionality)



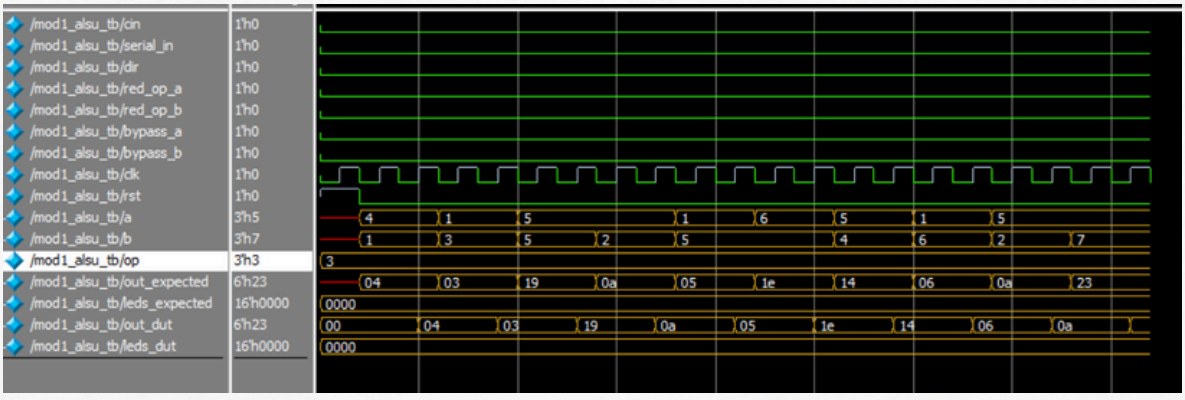
- test 4 (op = 1 functionality)



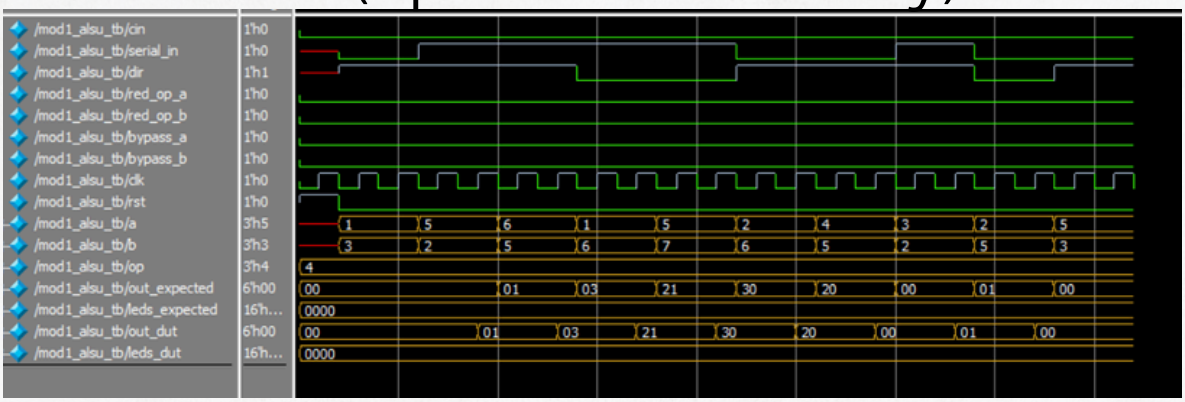
- test 5(op = 2 functionality)



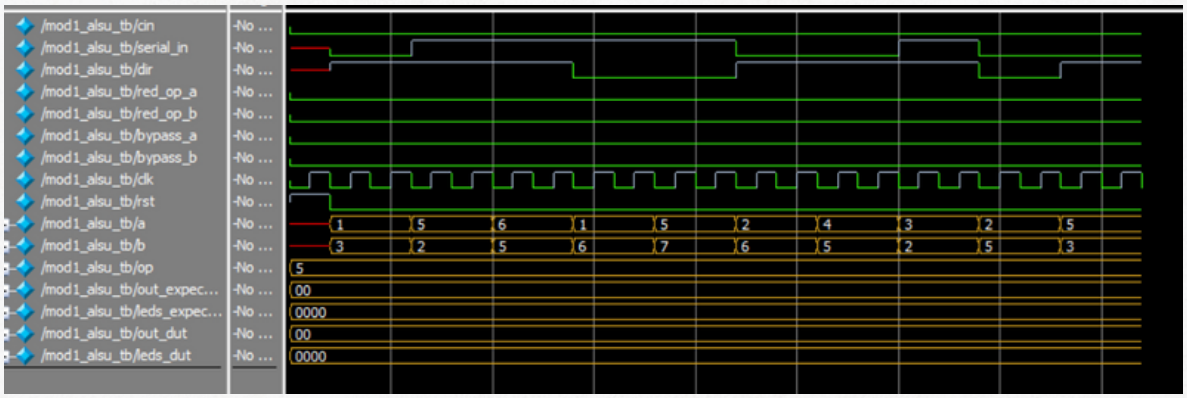
- test 6 (op = 3 functionality)



- test 7 (op = 4 functionality)



- test 8 (op = 5 functionality)



- **Do file :**

```
vlib work
vlog 1.v 1_tb.v
vsim -voptargs=+acc work.mod1_alsu_tb
add wave *

run -all

#quit -sim
```

- **Constraint file (vivado)**

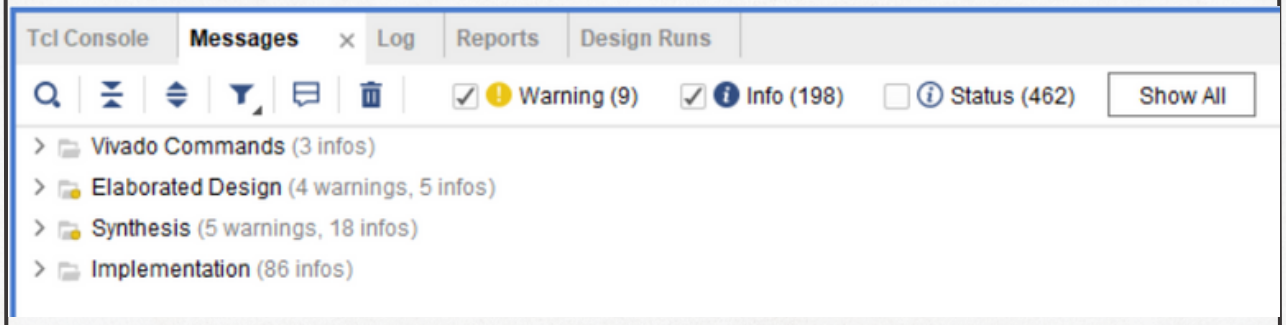
```
##Clock signal
set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]


## Switches
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {a[0]}]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {a[1]}]
set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports {a[2]}]
set_property -dict { PACKAGE_PIN W17   IOSTANDARD LVCMOS33 } [get_ports {b[0]}]
set_property -dict { PACKAGE_PIN W15   IOSTANDARD LVCMOS33 } [get_ports {b[1]}]
set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports {b[2]}]
set_property -dict { PACKAGE_PIN W14   IOSTANDARD LVCMOS33 } [get_ports {op[0]}]
set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports {op[1]}]
set_property -dict { PACKAGE_PIN V2    IOSTANDARD LVCMOS33 } [get_ports {op[2]}]
set_property -dict { PACKAGE_PIN T3    IOSTANDARD LVCMOS33 } [get_ports {cin}]
set_property -dict { PACKAGE_PIN T2    IOSTANDARD LVCMOS33 } [get_ports {serial_in}]
set_property -dict { PACKAGE_PIN R3    IOSTANDARD LVCMOS33 } [get_ports {dir}]
set_property -dict { PACKAGE_PIN W2    IOSTANDARD LVCMOS33 } [get_ports {red_op_a}]
set_property -dict { PACKAGE_PIN U1    IOSTANDARD LVCMOS33 } [get_ports {red_op_b}]
set_property -dict { PACKAGE_PIN T1    IOSTANDARD LVCMOS33 } [get_ports {bypass_a}]
set_property -dict { PACKAGE_PIN R2    IOSTANDARD LVCMOS33 } [get_ports {bypass_b}]
```

```
## LEDs
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {leds[0]}]
set_property -dict { PACKAGE_PIN E19   IOSTANDARD LVCMOS33 } [get_ports {leds[1]}]
set_property -dict { PACKAGE_PIN U19   IOSTANDARD LVCMOS33 } [get_ports {leds[2]}]
set_property -dict { PACKAGE_PIN V19   IOSTANDARD LVCMOS33 } [get_ports {leds[3]}]
set_property -dict { PACKAGE_PIN W18   IOSTANDARD LVCMOS33 } [get_ports {leds[4]}]
set_property -dict { PACKAGE_PIN U15   IOSTANDARD LVCMOS33 } [get_ports {leds[5]}]
set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports {leds[6]}]
set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 } [get_ports {leds[7]}]
set_property -dict { PACKAGE_PIN V13   IOSTANDARD LVCMOS33 } [get_ports {leds[8]}]
set_property -dict { PACKAGE_PIN V3    IOSTANDARD LVCMOS33 } [get_ports {leds[9]}]
set_property -dict { PACKAGE_PIN W3    IOSTANDARD LVCMOS33 } [get_ports {leds[10]}]
set_property -dict { PACKAGE_PIN U3    IOSTANDARD LVCMOS33 } [get_ports {leds[11]}]
set_property -dict { PACKAGE_PIN P3    IOSTANDARD LVCMOS33 } [get_ports {leds[12]}]
set_property -dict { PACKAGE_PIN N3    IOSTANDARD LVCMOS33 } [get_ports {leds[13]}]
set_property -dict { PACKAGE_PIN P1    IOSTANDARD LVCMOS33 } [get_ports {leds[14]}]
set_property -dict { PACKAGE_PIN L1    IOSTANDARD LVCMOS33 } [get_ports {leds[15]}]
```

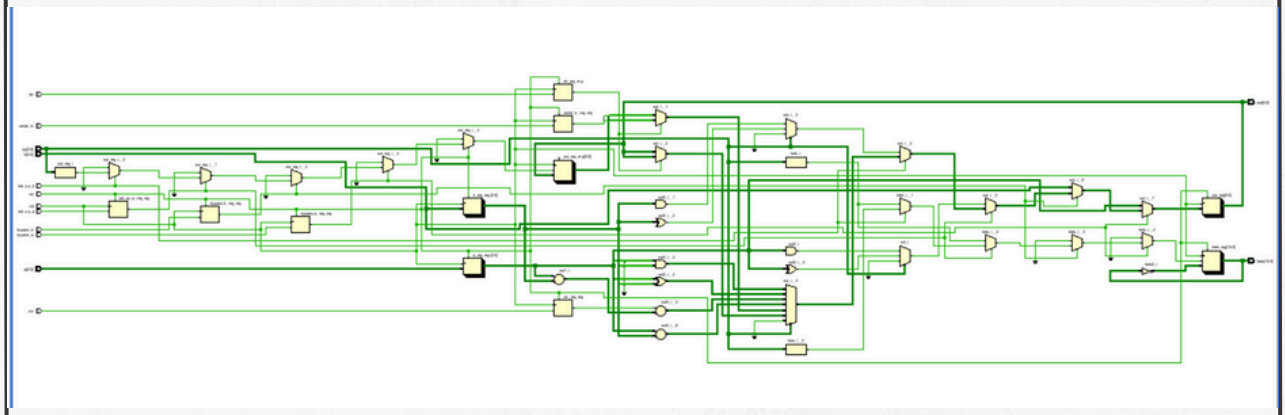
```
##Buttons
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports rst]
#set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports btnU]
#set_property -dict { PACKAGE_PIN W19   IOSTANDARD LVCMOS33 } [get_ports btnL]
#set_property -dict { PACKAGE_PIN T17   IOSTANDARD LVCMOS33 } [get_ports btnR]
#set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports btnD]
```

- **Messages : (Elaporation)**

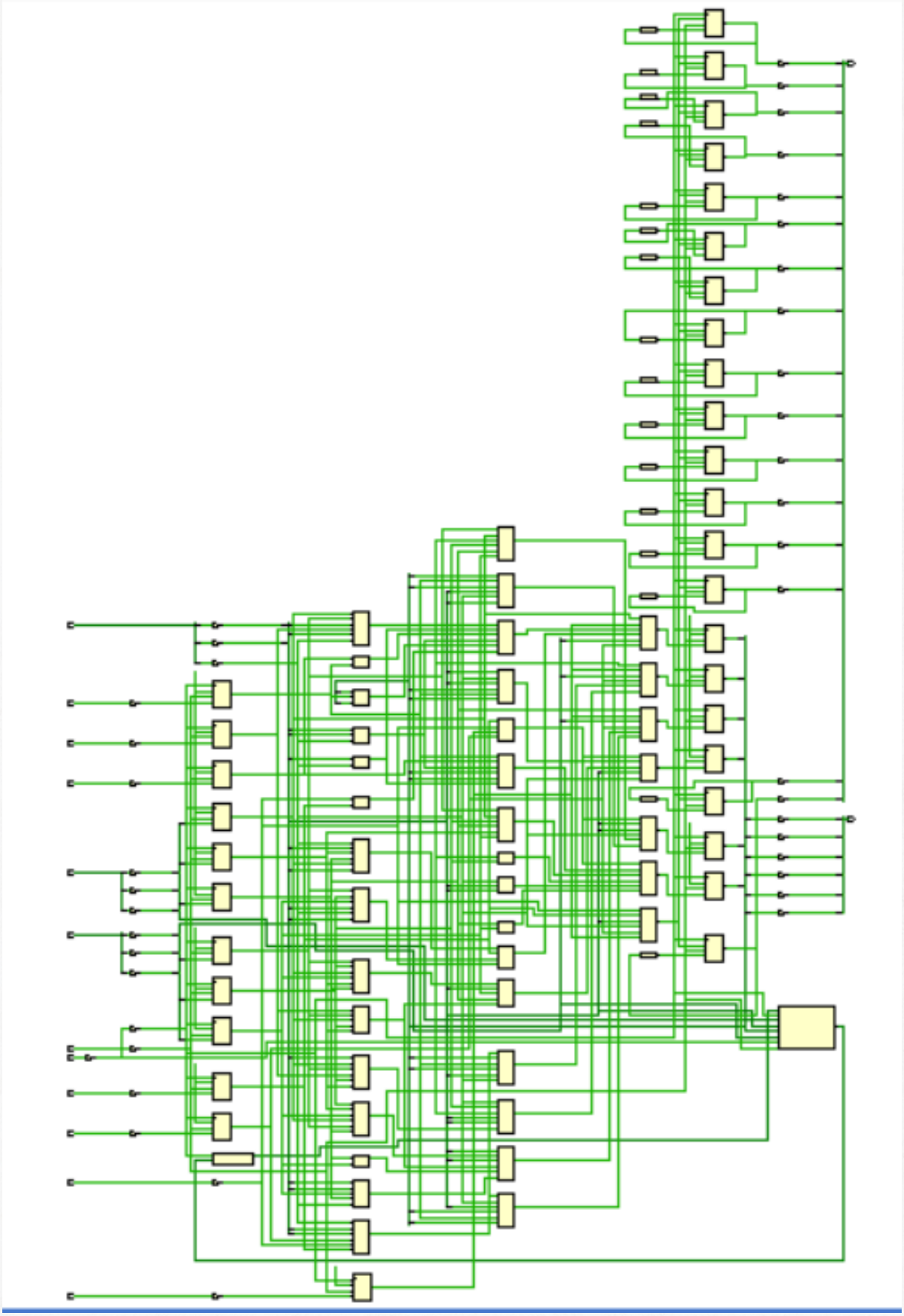
I already ran all othem



- **RTL schematic : (Elaporation)**

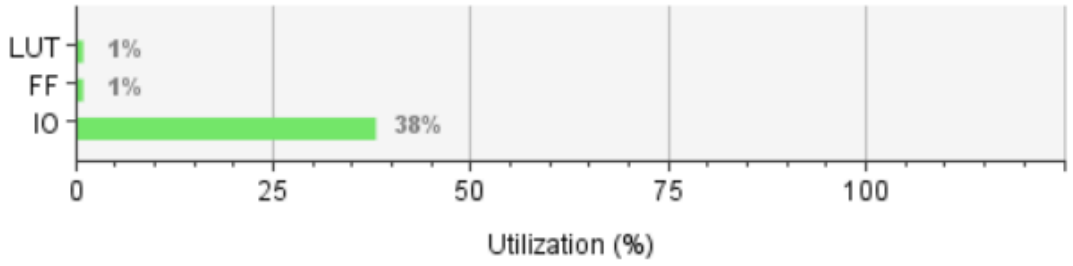


- **Technology schematic : (Synthesis)**



- **Utilization report: (Synthesis)**

Resource	Utilization	Available	Utilization %
LUT	48	20800	0.23
FF	34	41600	0.08
IO	40	106	37.74

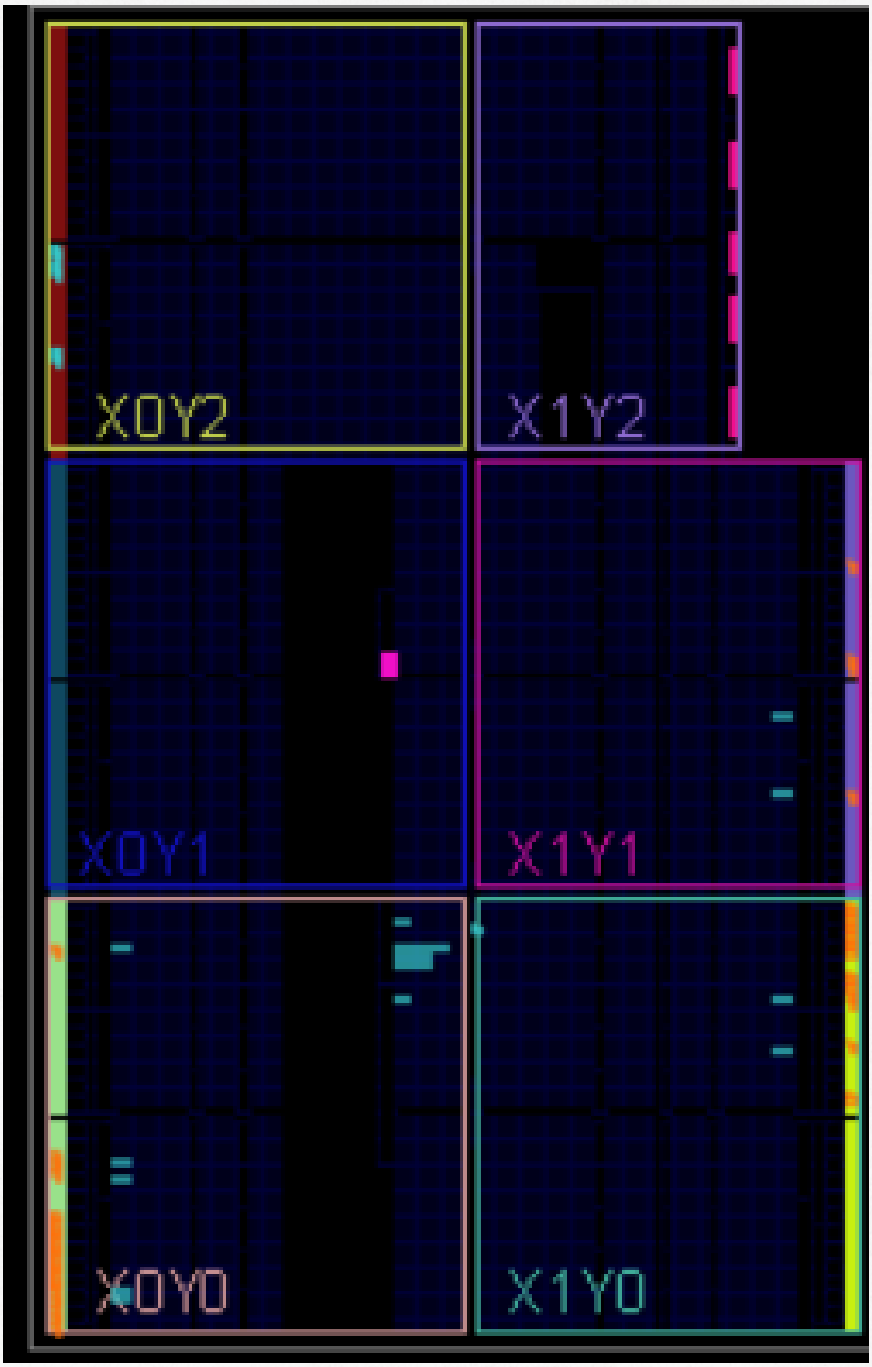


- **Timing report : (Synthesis)**

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.461 ns	Worst Hold Slack (WHS):	0.139 ns	Worst Pulse Width Slack (WPWS):	4.500 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:	38	Total Number of Endpoints:	38	Total Number of Endpoints:	35

All user specified timing constraints are met.

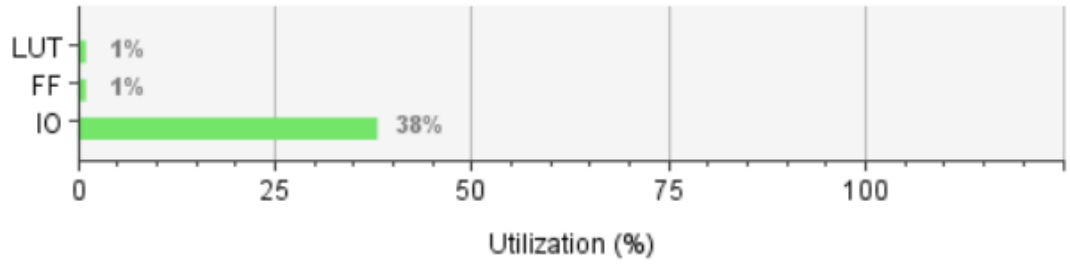
- **Device Snippets : (implementation)**



- **Utilization report: (Implementation)**

Summary

Resource	Utilization	Available	Utilization %
LUT	48	20800	0.23
FF	46	41600	0.11
IO	40	106	37.74



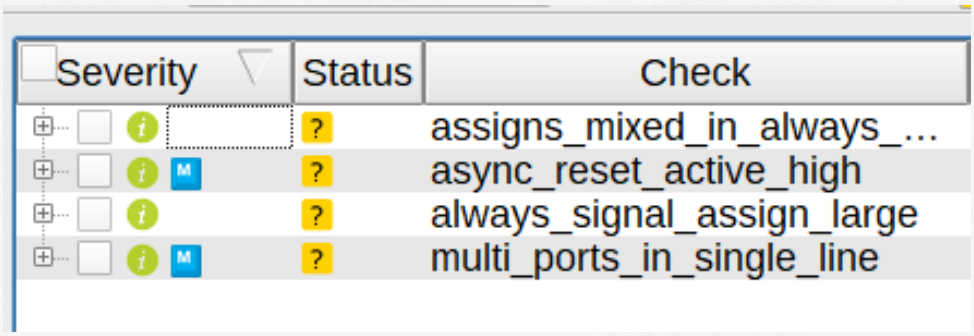
- **Timing report : (Implementation)**

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.467 ns	Worst Hold Slack (WHS):	0.178 ns	Worst Pulse Width Slack (WPWS):	4.500 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:	56	Total Number of Endpoints:	56	Total Number of Endpoints:	47

All user specified timing constraints are met.

- **lint tool messages:**



Severity	Status	Check
	?	assigns_mixed_in_always_...
	?	async_reset_active_high
	?	always_signal_assign_large
	?	multi_ports_in_single_line

Module 2 :

- **RTL code :**

```
module mod2_dsp#(parameter OPERATION = "ADD")
                (input [17:0] a, b, d,
                input [47:0] c,
                input clk, rst_n,
                output reg [47:0] p);

    reg [17:0] a_reg, b_reg, d_reg;
    reg [18:0] first_arth_out;
    reg[47:0] multiplier_out;
    reg [47:0] c_reg;

    always @(posedge clk) begin
        if(!rst_n) begin
            a_reg <= 0;
            b_reg <= 0;
            c_reg <= 0;
            d_reg <= 0;
        end
        else begin
            a_reg <= a;
            b_reg <= b;
            c_reg <= c;
            d_reg <= d;
        end
    end
```

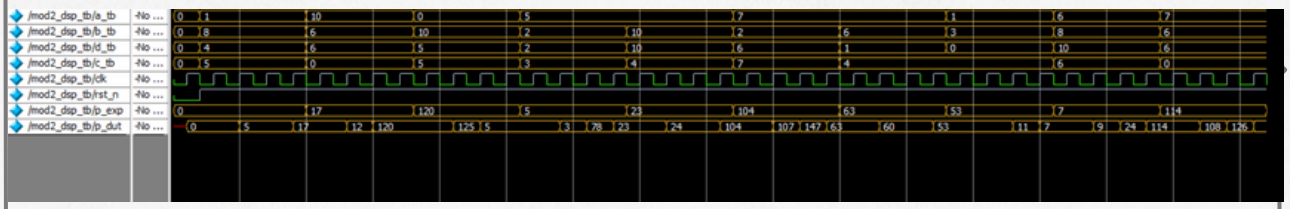
- **RTL code :**

```
27      always @(posedge clk) begin
28          if(!rst_n) begin
29              first_arth_out <= 0;
30              multiplier_out <= 0;
31              p <= 0;
32          end
33          else begin
34              if(OPERATION == "ADD") begin
35                  first_arth_out <= b_reg + d_reg;
36                  multiplier_out <= a_reg * first_arth_out;
37                  p <= c_reg + multiplier_out;
38              end
39              else if(OPERATION == "SUB") begin
40                  first_arth_out <= b_reg - d_reg;
41                  multiplier_out <= a_reg * first_arth_out;
42                  p <= c_reg - multiplier_out;
43              end
44          end
45      end
46  endmodule
```

- **Testbench :**

```
module mod2_dsp_tb();
    reg [17:0] a_tb, b_tb, d_tb;
    reg [47:0] c_tb;
    reg clk, rst_n;

    reg [47:0] p_exp;
    wire [47:0] p_dut;

    mod2_dsp dut(a_tb, b_tb, d_tb, c_tb, clk, rst_n, p_dut);
    initial begin
        clk = 0;
        forever
            #1 clk = ~clk;
    end
    initial begin
        rst_n = 0;
        a_tb = 0;
        b_tb = 0;
        c_tb = 0;
        d_tb = 0;
        p_exp = 0;
        @(negedge clk);
        rst_n = 1;
        repeat(10) begin
            a_tb = $random%10;
            b_tb = $random%10;
            c_tb = $random%10;
            d_tb = $random%10;
            repeat(4)
                @(negedge clk);
            p_exp = ((b_tb + d_tb)*a_tb)+c_tb;
            // if(p_dut == p_exp) begin
            //     $display("ERROR");
            //     $stop;
            // end

        end
            $stop;
    end
endmodule
```

- **Waveforms : (questasim)**



- **Do file :**

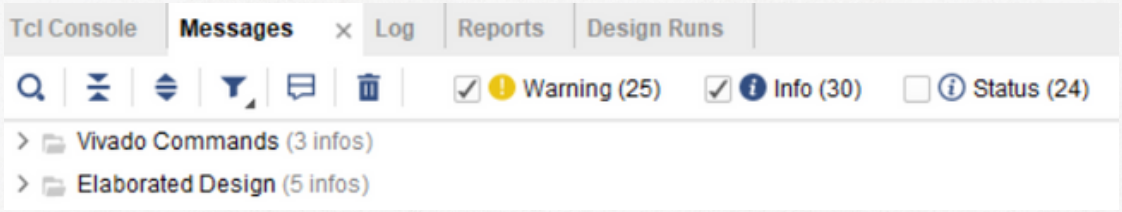
```
1   vlib work
2   vlog 2.v 2_tb.v
3   vsim -voptargs=+acc work.mod2_dsp_tb
4   add wave *
5
6   run -all
7
8   #quit -sim
```

- **Constraint file (vivado)**

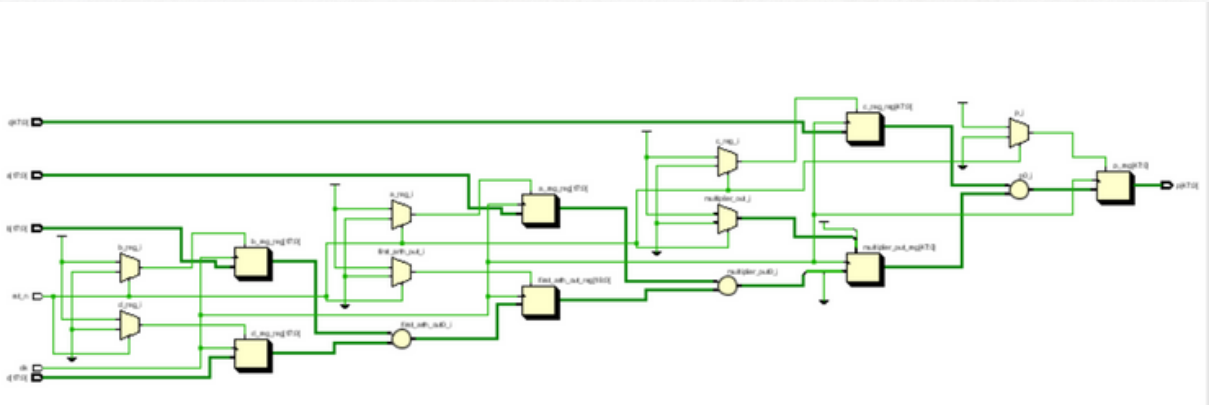
timing only

```
6   ##Clock signal
7   set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
8   create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
9
```

- **Messages : (Elaporation)**

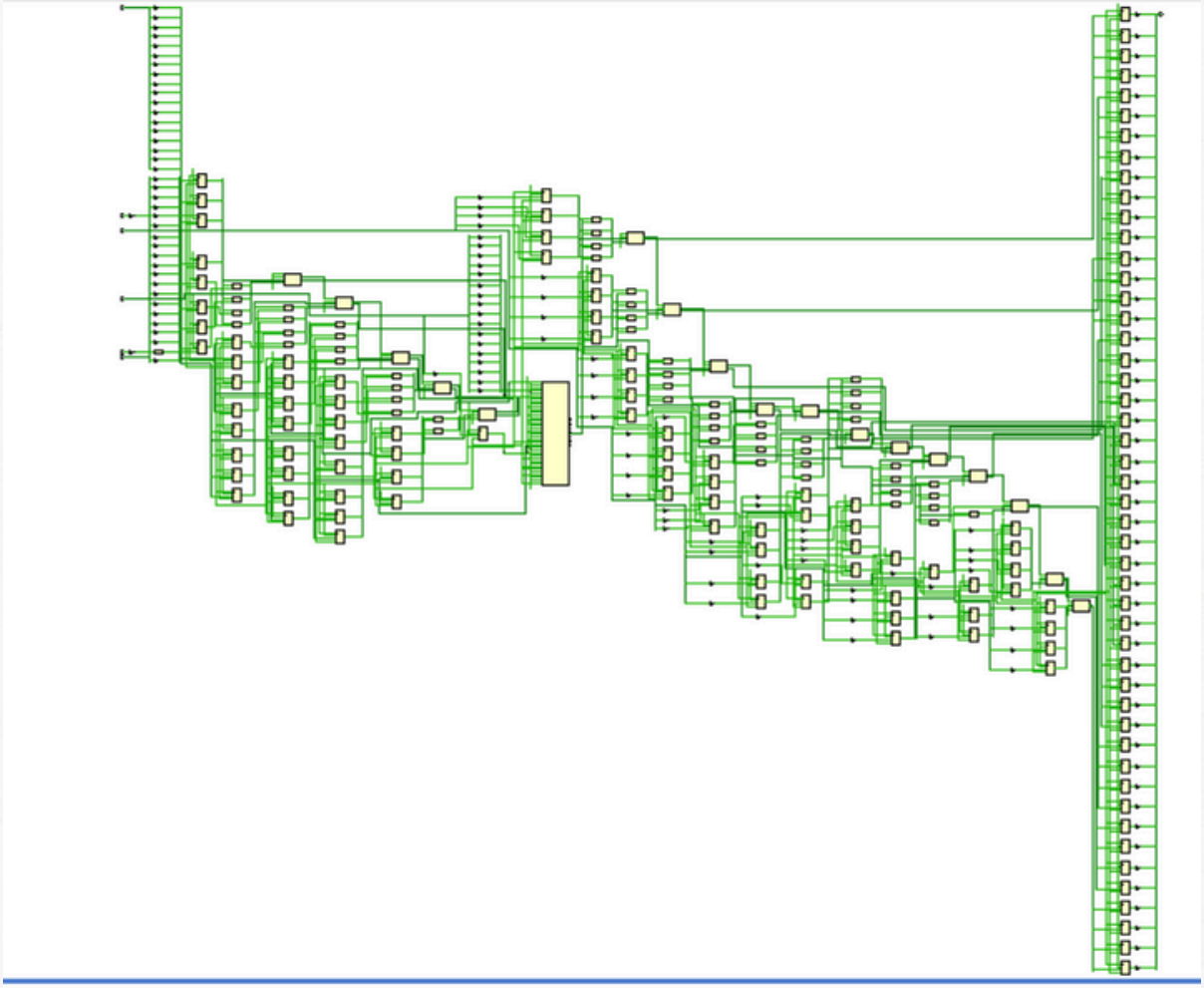


- **RTL schematic : (Elaporation)**



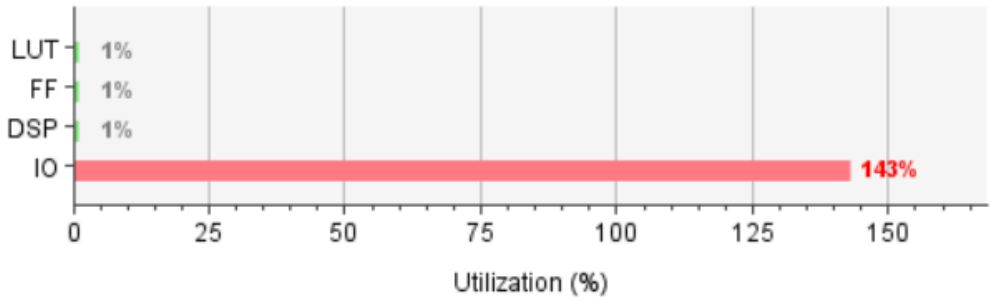
- **Messages : (synthesis)**

- **Technology schematic : (Synthesis)**



- **Utilization report: (Synthesis)**

Resource	Utilization	Available	Utilization %
LUT	56	20800	0.27
FF	133	41600	0.32
DSP	1	90	1.11
IO	152	106	143.40



- **Timing report : (Synthesis)**

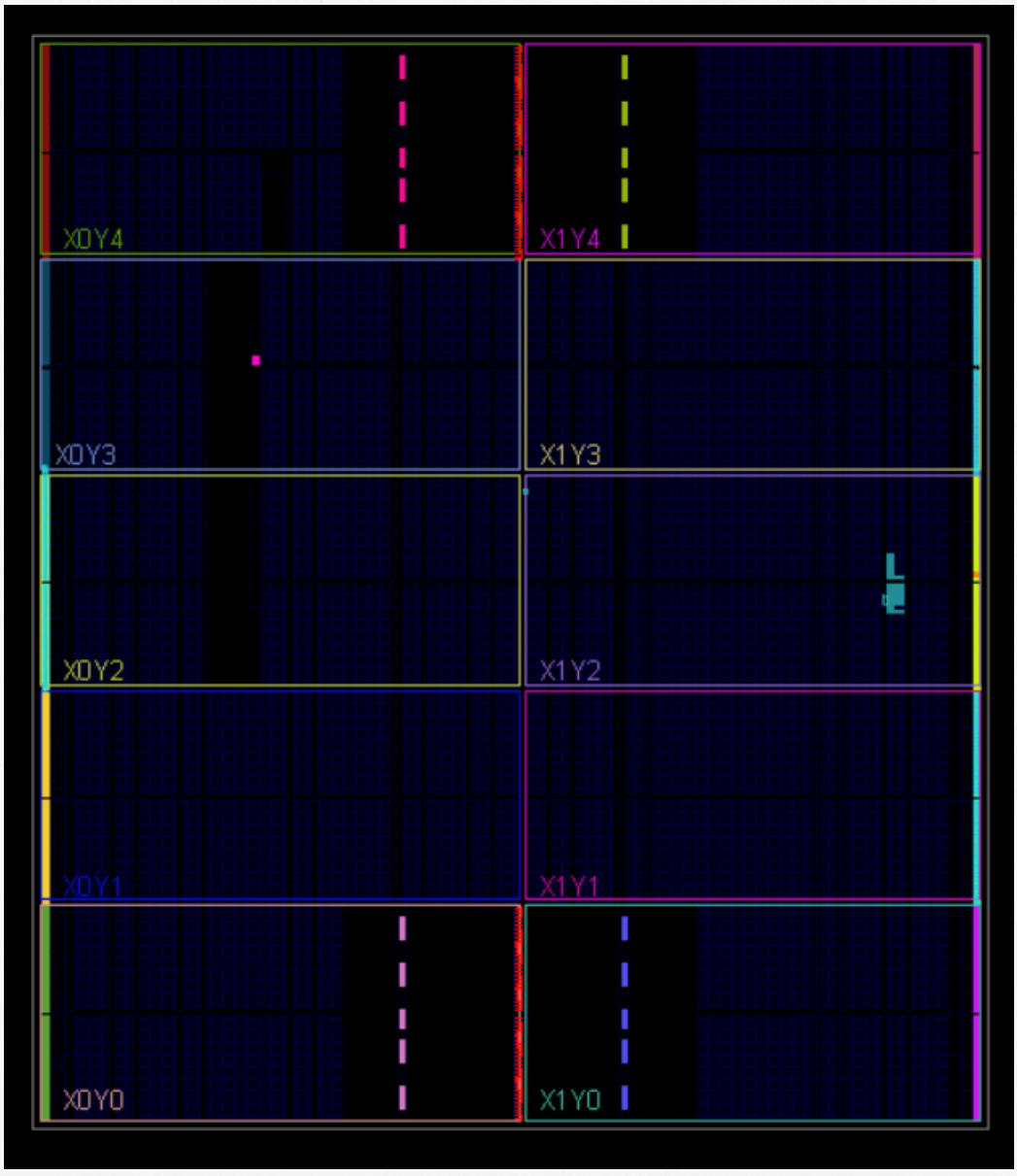
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.127 ns	Worst Hold Slack (WHS):	0.183 ns	Worst Pulse Width Slack (WPWS):	4.500 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:	69	Total Number of Endpoints:	69	Total Number of Endpoints:	135

All user specified timing constraints are met.

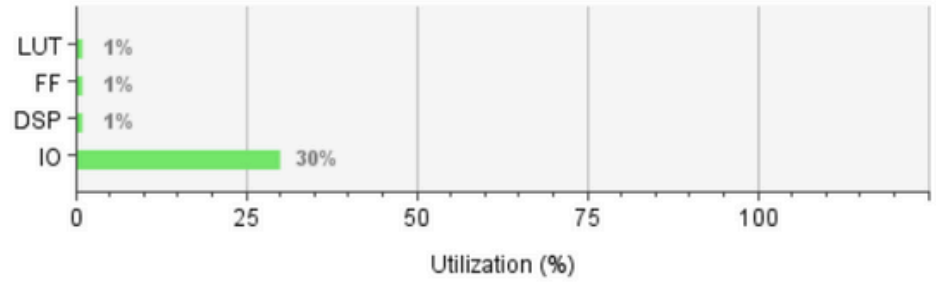
- **Messages : (Implementation)**

- **Device snippet : (Implementation)**



- **Utilization report: (Implementation)**

Resource	Utilization	Available	Utilization %
LUT	56	133800	0.04
FF	133	267600	0.05
DSP	1	740	0.14
IO	152	500	30.40

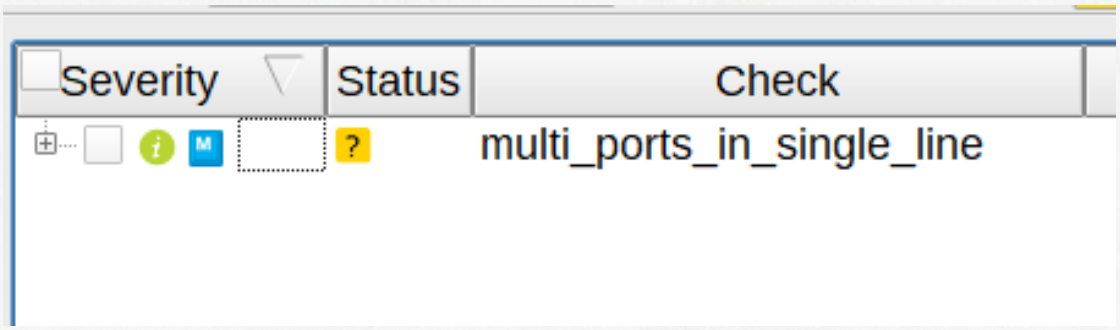


- **Timing report : (Implementation)**

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	7.238 ns	Worst Hold Slack (WHS):	0.165 ns	Worst Pulse Width Slack (WPWS):	4.500 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:	69	Total Number of Endpoints:	69	Total Number of Endpoints:	135

- **lint result :**



Module 3 :

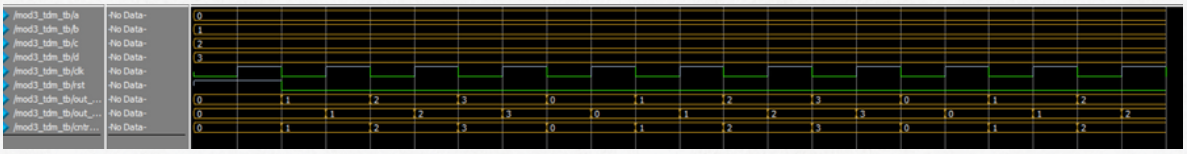
- **RTL code :**

```
1   module mod3_tdm(input [1:0] a, b, c, d,
2                   input clk, rst,
3                   output reg [1:0]out);
4       reg [1:0]counter;
5       always @(posedge clk, posedge rst) begin
6           if (rst)
7               counter <= 0;
8           else
9               counter = counter + 1;
10          case (counter)
11              0 : out <= a;
12              1 : out <= b;
13              2 : out <= c;
14              3 : out <= d;
15          endcase
16      end
17  endmodule
18
19
```

- **Testbench code :**

```
module mod3_tdm_tb();
    reg [1:0] a, b, c, d;
    reg clk, rst;
    reg [1:0] out_exp;
    wire [1:0] out_dut;
    reg [1:0]cntr_tb;
    mod3_tdm dut(a, b, c, d, clk, rst, out_dut);

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

    initial begin
        rst = 1;
        cntr_tb = 0;
        a = 0;
        b = 1;
        c = 2;
        d = 3;
        out_exp = 0;
        @(negedge clk);
        rst = 0;
        repeat(10) begin
            cntr_tb = cntr_tb +1;
            case (cntr_tb)
                0 : out_exp <= a;
                2 : out_exp <= c;
                1 : out_exp <= b;
                3 : out_exp <= d;
            endcase
            @(negedge clk);
            if(out_dut != out_exp) begin
                $display("ERROR");
                $stop;
            end
        end
        $stop;
    end
endmodule
```

- **Waveforms : (questasim)**



- **Do file :**

```
vlib work
vlog 3.v 3_tb.v
vsim -voptargs=+acc work.mod3_tdm_tb
add wave *

run -all

#quit -sim
```

- **Constraint file (vivado)**

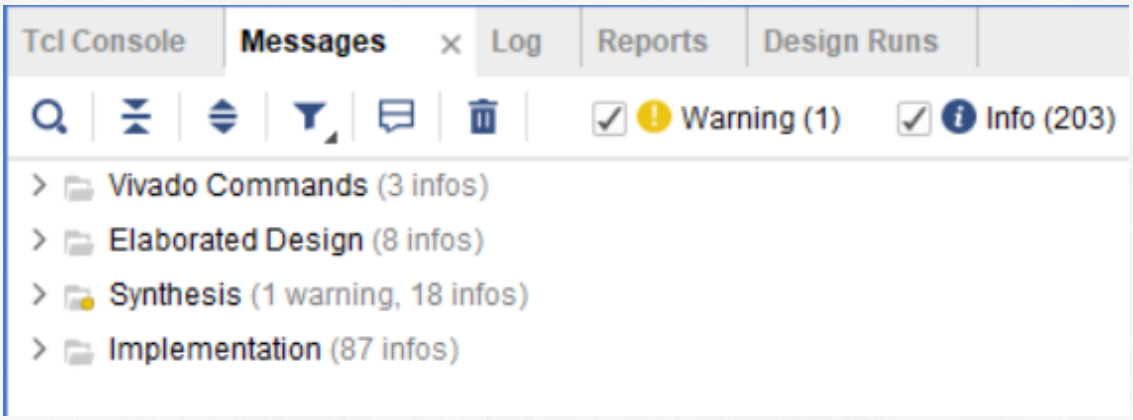
```
##Clock signal
set_property -dict {PACKAGE_PIN W5 IOSTANDARD LVCMOS33} [get_ports clk]
create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_ports clk]


## Switches
set_property -dict {PACKAGE_PIN V17 IOSTANDARD LVCMOS33} [get_ports {a[0]}]
set_property -dict {PACKAGE_PIN V16 IOSTANDARD LVCMOS33} [get_ports {a[1]}]
set_property -dict {PACKAGE_PIN W16 IOSTANDARD LVCMOS33} [get_ports {b[0]}]
set_property -dict {PACKAGE_PIN W17 IOSTANDARD LVCMOS33} [get_ports {b[1]}]
set_property -dict {PACKAGE_PIN W15 IOSTANDARD LVCMOS33} [get_ports {c[0]}]
set_property -dict {PACKAGE_PIN V15 IOSTANDARD LVCMOS33} [get_ports {c[1]}]
set_property -dict {PACKAGE_PIN W14 IOSTANDARD LVCMOS33} [get_ports {d[0]}]
set_property -dict {PACKAGE_PIN W13 IOSTANDARD LVCMOS33} [get_ports {d[1]}]
```

```
## LEDs
set_property -dict {PACKAGE_PIN U16 IOSTANDARD LVCMOS33} [get_ports {out[0]}]
set_property -dict {PACKAGE_PIN E19 IOSTANDARD LVCMOS33} [get_ports {out[1]}]
```

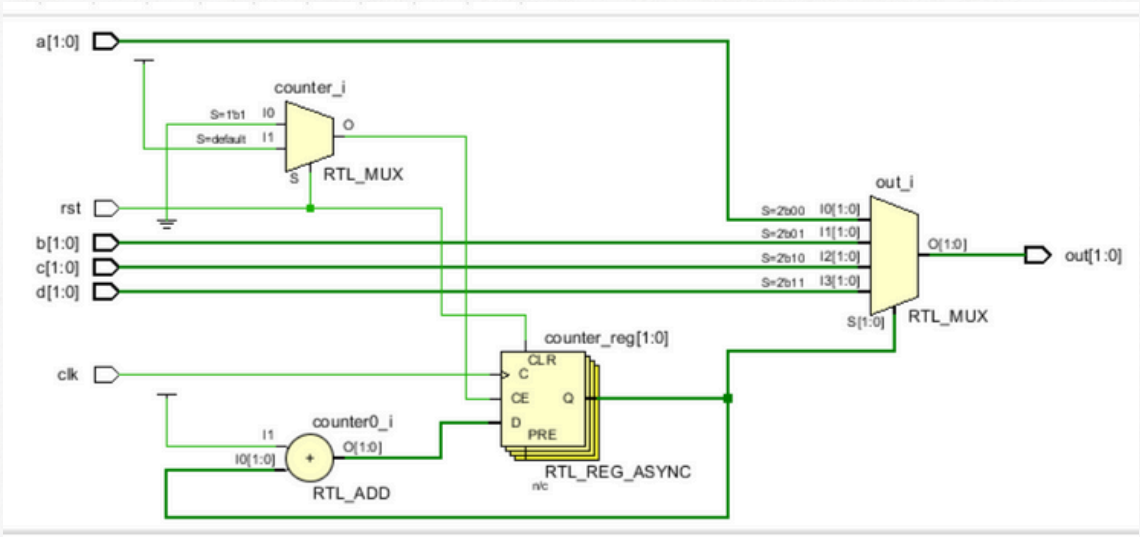
```
##Buttons
set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports rst]
```

- **Messages : (Elaporation)**

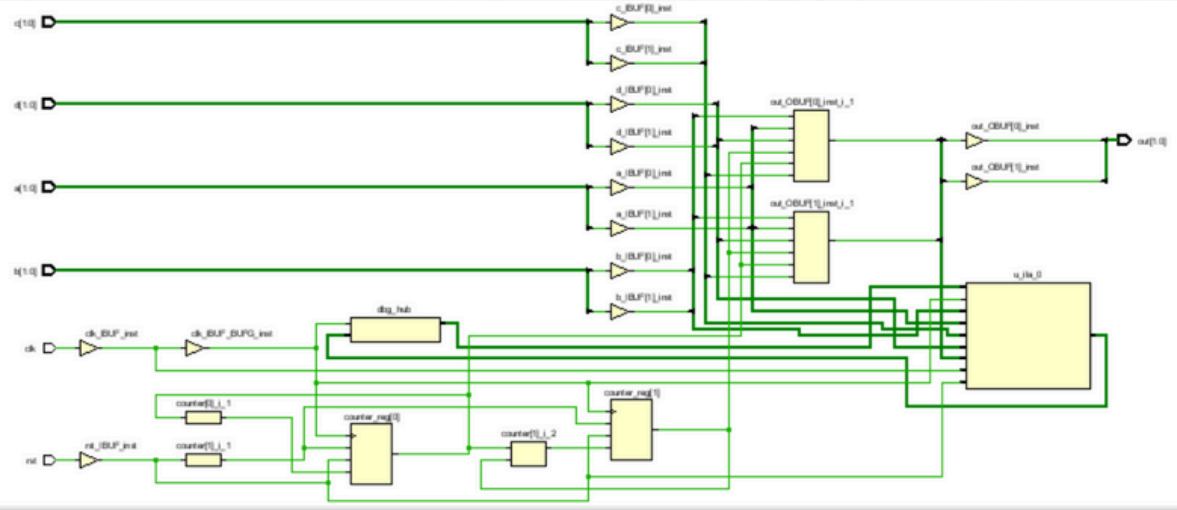
I already ran all othem



- **RTL schematic : (Elaporation)**



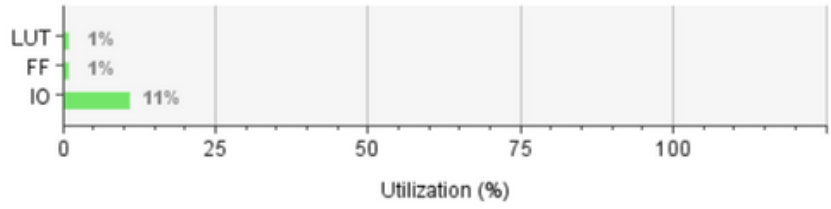
- **Technology schematic : (Synthesis)**



- **Utilization report: (Synthesis)**

Summary

Resource	Utilization	Available	Utilization %
LUT	4	20800	0.02
FF	2	41600	0.00
IO	12	106	11.32

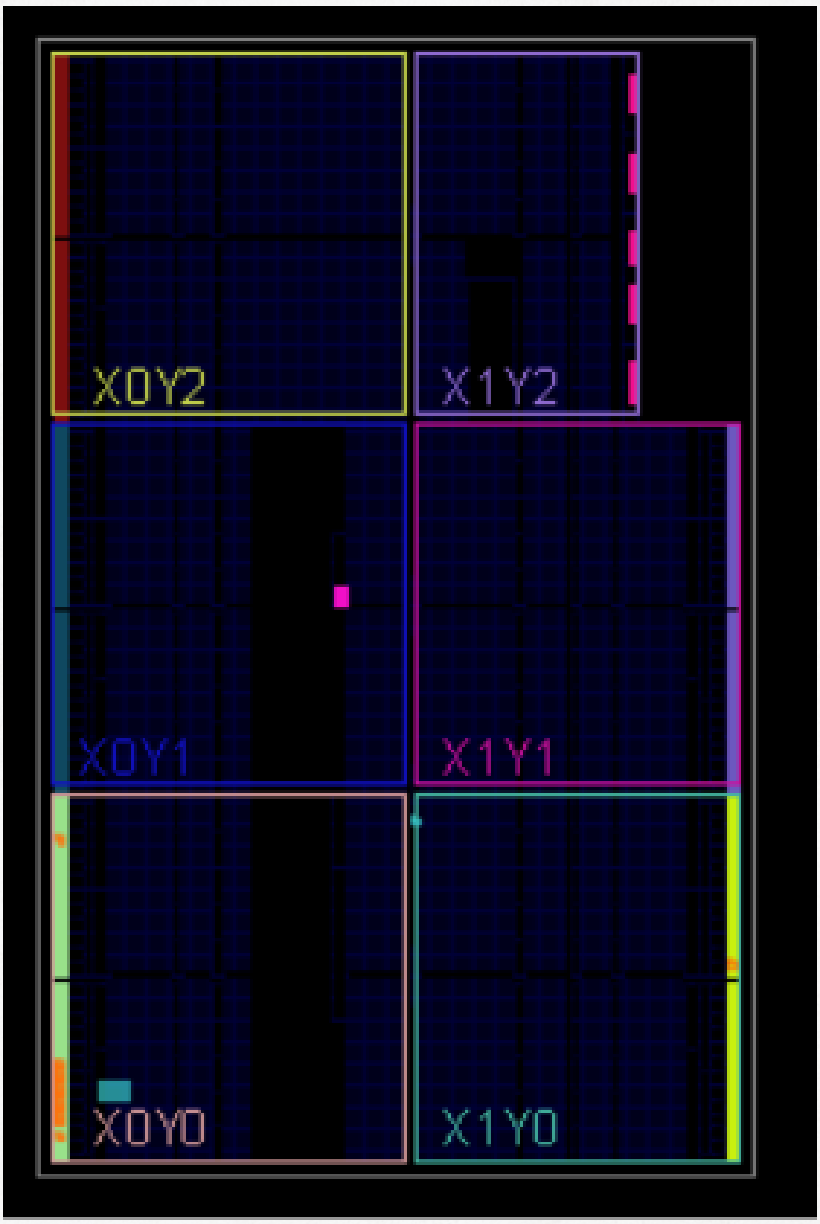


- **Timing report : (Synthesis)**

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	8.578 ns	Worst Hold Slack (WHS):	0.144 ns	Worst Pulse Width Slack (WPWS):	4.500 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:	2	Total Number of Endpoints:	2	Total Number of Endpoints:	3

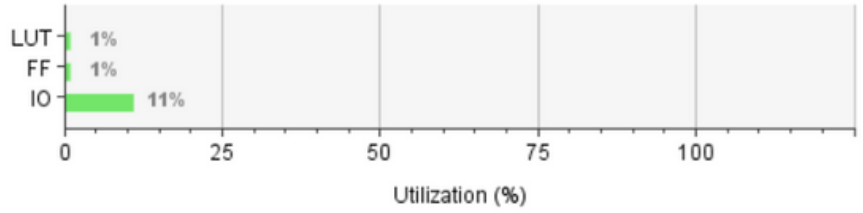
- **Device snippet : (Implementation)**



- **Utilization report: (Implementation)**

Summary

Resource	Utilization	Available	Utilization %
LUT	4	20800	0.02
FF	2	41600	0.00
IO	12	106	11.32

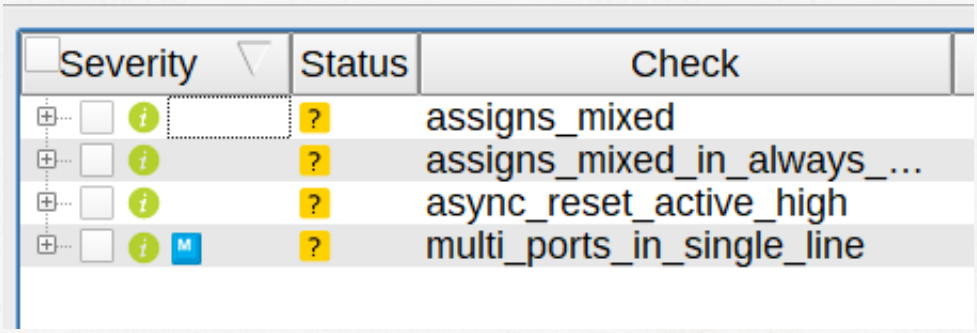


- **Timing report : (Implementation)**

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	8.263 ns	Worst Hold Slack (WHS):	0.411 ns	Worst Pulse Width Slack (WPWS):	4.500 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:	2	Total Number of Endpoints:	2	Total Number of Endpoints:	3

- **lint result :**



Severity	Status	Check
	?	assigns_mixed
	?	assigns_mixed_in_always_…
	?	async_reset_active_high
M	?	multi_ports_in_single_line