Digital Design Diploma

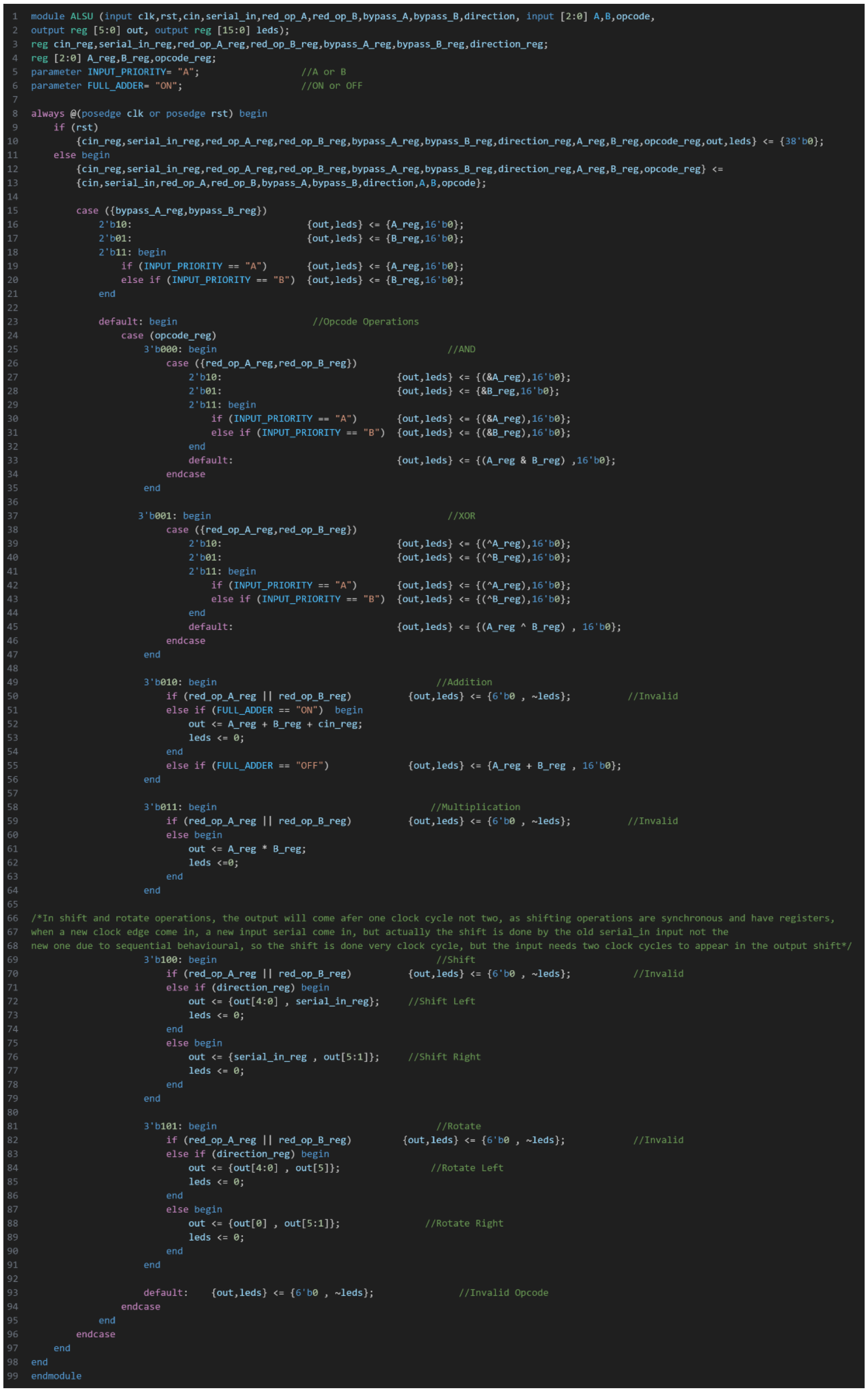
Assignment 4

Sequential Logic Design

Name	كريم حسن عاطف علي
Group	G2

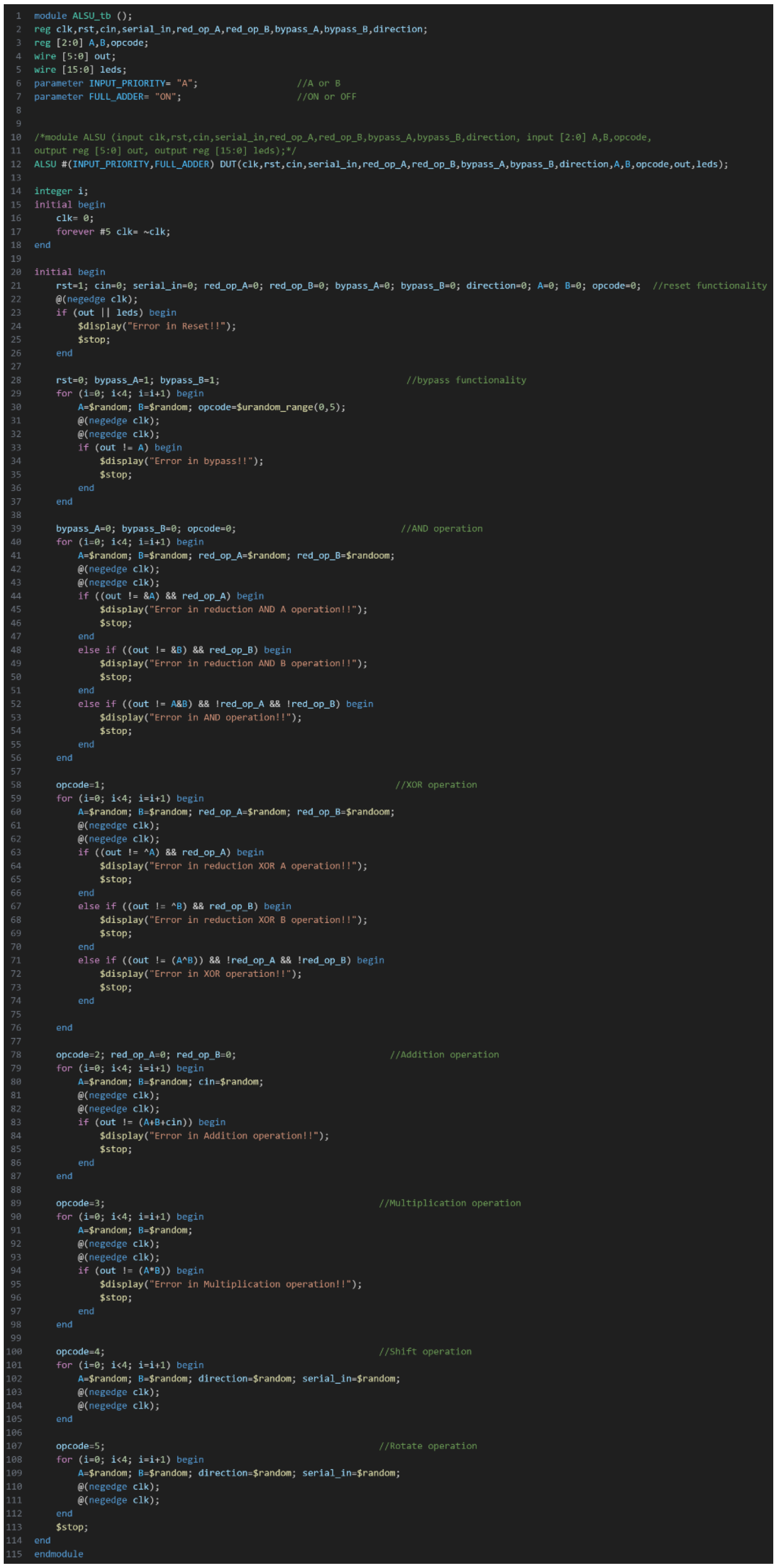
Submitted to: Eng. Kareem Waseem

1) ALSU:



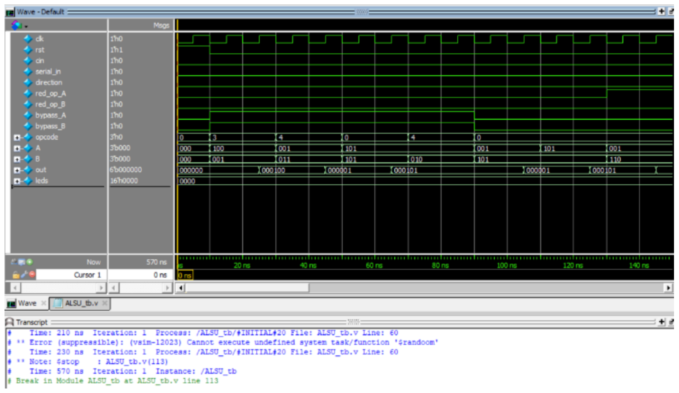
```verilog
module ALSU (input clk,rst,cin,serial_in,red_op_A,red_op_B,bypass_A,bypass_B,direction, input [2:0] A,B,opcode,
output reg [5:0] out, output reg [15:0] leds);
reg cin_reg,serial_in_reg,red_op_A_reg,red_op_B_reg,bypass_A_reg,bypass_B_reg,direction_reg;
reg [2:0] A_reg,B_reg,opcode_reg;
parameter INPUT_PRIORITY= "A";                  //A or B
parameter FULL_ADDER= "ON";                     //ON or OFF

always @(posedge clk or posedge rst) begin
    if (rst)
        {cin_reg,serial_in_reg,red_op_A_reg,red_op_B_reg,bypass_A_reg,bypass_B_reg,direction_reg,A_reg,B_reg,opcode_reg,out,leds} <= {38'b0};
    else begin
        {cin_reg,serial_in_reg,red_op_A_reg,red_op_B_reg,bypass_A_reg,bypass_B_reg,direction_reg,A_reg,B_reg,opcode_reg} <=
        {cin,serial_in,red_op_A,red_op_B,bypass_A,bypass_B,direction,A,B,opcode};

        case ({bypass_A_reg,bypass_B_reg})
            2'b10:                          {out,leds} <= {A_reg,16'b0};
            2'b01:                          {out,leds} <= {B_reg,16'b0};
            2'b11: begin
                if (INPUT_PRIORITY == "A")       {out,leds} <= {A_reg,16'b0};
                else if (INPUT_PRIORITY == "B")  {out,leds} <= {B_reg,16'b0};
            end

            default: begin                         //Opcode Operations
                case (opcode_reg)
                    3'b000: begin                                       //AND
                        case ({red_op_A_reg,red_op_B_reg})
                            2'b10:                              {out,leds} <= {(&A_reg),16'b0};
                            2'b01:                              {out,leds} <= {&B_reg,16'b0};
                            2'b11: begin
                                if (INPUT_PRIORITY == "A")       {out,leds} <= {(&A_reg),16'b0};
                                else if (INPUT_PRIORITY == "B")  {out,leds} <= {(&B_reg),16'b0};
                            end
                            default:                            {out,leds} <= {(A_reg & B_reg) ,16'b0};
                        endcase
                    end

                   3'b001: begin                                        //XOR
                        case ({red_op_A_reg,red_op_B_reg})
                            2'b10:                              {out,leds} <= {(^A_reg),16'b0};
                            2'b01:                              {out,leds} <= {(^B_reg),16'b0};
                            2'b11: begin
                                if (INPUT_PRIORITY == "A")       {out,leds} <= {(^A_reg),16'b0};
                                else if (INPUT_PRIORITY == "B")  {out,leds} <= {(^B_reg),16'b0};
                            end
                            default:                            {out,leds} <= {(A_reg ^ B_reg) , 16'b0};
                        endcase
                    end

                    3'b010: begin                                     //Addition
                        if (red_op_A_reg || red_op_B_reg)          {out,leds} <= {6'b0 , ~leds};          //Invalid
                        else if (FULL_ADDER == "ON")  begin
                            out <= A_reg + B_reg + cin_reg;
                            leds <= 0;
                        end
                        else if (FULL_ADDER == "OFF")              {out,leds} <= {A_reg + B_reg , 16'b0};
                    end

                    3'b011: begin                                    //Multiplication
                        if (red_op_A_reg || red_op_B_reg)          {out,leds} <= {6'b0 , ~leds};          //Invalid
                        else begin
                            out <= A_reg * B_reg;
                            leds <=0;
                        end
                    end

/*In shift and rotate operations, the output will come afer one clock cycle not two, as shifting operations are synchronous and have registers,
when a new clock edge come in, a new input serial come in, but actually the shift is done by the old serial_in input not the
new one due to sequential behavioural, so the shift is done very clock cycle, but the input needs two clock cycles to appear in the output shift*/
                    3'b100: begin                                     //Shift
                        if (red_op_A_reg || red_op_B_reg)          {out,leds} <= {6'b0 , ~leds};           //Invalid
                        else if (direction_reg) begin
                            out <= {out[4:0] , serial_in_reg};     //Shift Left
                            leds <= 0;
                        end
                        else begin
                            out <= {serial_in_reg , out[5:1]};     //Shift Right
                            leds <= 0;
                        end
                    end

                    3'b101: begin                                     //Rotate
                        if (red_op_A_reg || red_op_B_reg)         {out,leds} <= {6'b0 , ~leds};            //Invalid
                        else if (direction_reg) begin
                            out <= {out[4:0] , out[5]};                //Rotate Left
                            leds <= 0;
                        end
                        else begin
                            out <= {out[0] , out[5:1]};               //Rotate Right
                            leds <= 0;
                        end
                    end

                    default:    {out,leds} <= {6'b0 , ~leds};               //Invalid Opcode
                endcase
            end
        endcase
    end
end
endmodule
```

Figure1: Q1 Code



```verilog
module ALSU_tb ();
reg clk,rst,cin,serial_in,red_op_A,red_op_B,bypass_A,bypass_B,direction;
reg [2:0] A,B,opcode;
wire [5:0] out;
wire [15:0] leds;
parameter INPUT_PRIORITY= "A";                  //A or B
parameter FULL_ADDER= "ON";                     //ON or OFF


/*module ALSU (input clk,rst,cin,serial_in,red_op_A,red_op_B,bypass_A,bypass_B,direction, input [2:0] A,B,opcode,
output reg [5:0] out, output reg [15:0] leds);*/
ALSU #(INPUT_PRIORITY,FULL_ADDER) DUT(clk,rst,cin,serial_in,red_op_A,red_op_B,bypass_A,bypass_B,direction,A,B,opcode,out,leds);

integer i;
initial begin
    clk= 0;
    forever #5 clk= ~clk;
end

initial begin
    rst=1; cin=0; serial_in=0; red_op_A=0; red_op_B=0; bypass_A=0; bypass_B=0; direction=0; A=0; B=0; opcode=0;  //reset functionality
    @(negedge clk);
    if (out || leds) begin
        $display("Error in Reset!!");
        $stop;
    end

    rst=0; bypass_A=1; bypass_B=1;                                  //bypass functionality
    for (i=0; i<4; i=i+1) begin
        A=$random; B=$random; opcode=$urandom_range(0,5);
        @(negedge clk);
        @(negedge clk);
        if (out != A) begin
            $display("Error in bypass!!");
            $stop;
        end
    end

    bypass_A=0; bypass_B=0; opcode=0;                               //AND operation
    for (i=0; i<4; i=i+1) begin
        A=$random; B=$random; red_op_A=$random; red_op_B=$randoom;
        @(negedge clk);
        @(negedge clk);
        if ((out != &A) && red_op_A) begin
            $display("Error in reduction AND A operation!!");
            $stop;
        end
        else if ((out != &B) && red_op_B) begin
            $display("Error in reduction AND B operation!!");
            $stop;
        end
        else if ((out != A&B) && !red_op_A && !red_op_B) begin
            $display("Error in AND operation!!");
            $stop;
        end
    end

    opcode=1;                                                       //XOR operation
    for (i=0; i<4; i=i+1) begin
        A=$random; B=$random; red_op_A=$random; red_op_B=$randoom;
        @(negedge clk);
        @(negedge clk);
        if ((out != ^A) && red_op_A) begin
            $display("Error in reduction XOR A operation!!");
            $stop;
        end
        else if ((out != ^B) && red_op_B) begin
            $display("Error in reduction XOR B operation!!");
            $stop;
        end
        else if ((out != (A^B)) && !red_op_A && !red_op_B) begin
            $display("Error in XOR operation!!");
            $stop;
        end

    end

    opcode=2; red_op_A=0; red_op_B=0;                               //Addition operation
    for (i=0; i<4; i=i+1) begin
        A=$random; B=$random; cin=$random;
        @(negedge clk);
        @(negedge clk);
        if (out != (A+B+cin)) begin
            $display("Error in Addition operation!!");
            $stop;
        end
    end

    opcode=3;                                                       //Multiplication operation
    for (i=0; i<4; i=i+1) begin
        A=$random; B=$random;
        @(negedge clk);
        @(negedge clk);
        if (out != (A*B)) begin
            $display("Error in Multiplication operation!!");
            $stop;
        end
    end

    opcode=4;                                                       //Shift operation
    for (i=0; i<4; i=i+1) begin
        A=$random; B=$random; direction=$random; serial_in=$random;
        @(negedge clk);
        @(negedge clk);
    end

    opcode=5;                                                       //Rotate operation
    for (i=0; i<4; i=i+1) begin
        A=$random; B=$random; direction=$random; serial_in=$random;
        @(negedge clk);
        @(negedge clk);
    end
    $stop;
end
endmodule
```

Figure2: Q1 Testbench



```
#    Time: 210 ns  Iteration: 1  Process: /ALSU_tb/#INITIAL#20 File: ALSU_tb.v Line: 60
# ** Error (suppressible): (vsim-12023) Cannot execute undefined system task/function '$randoom'
#    Time: 230 ns  Iteration: 1  Process: /ALSU_tb/#INITIAL#20 File: ALSU_tb.v Line: 60
# ** Note: $stop    : ALSU_tb.v(113)
#    Time: 570 ns  Iteration: 1  Instance: /ALSU_tb
# Break in Module ALSU_tb at ALSU_tb.v line 113
```

Figure3: Q1 Wave1

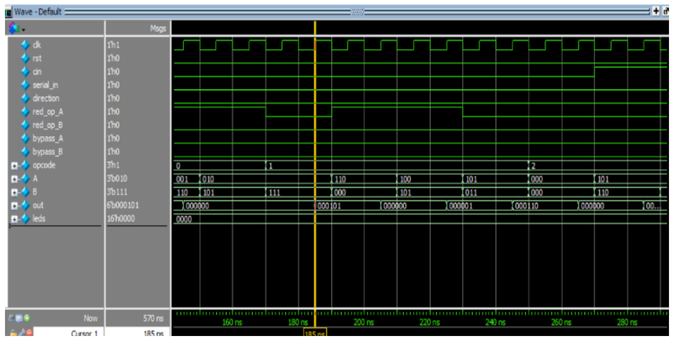


Figure4: Q1 Wave2

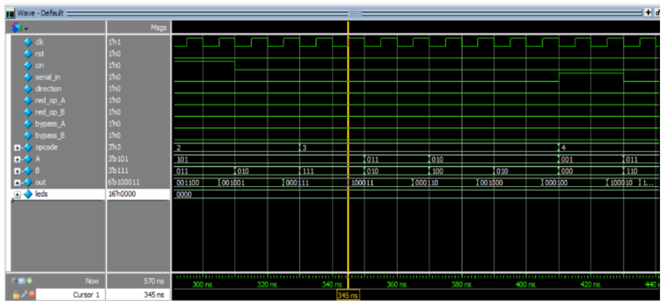


Figure5: Q1 Wave3

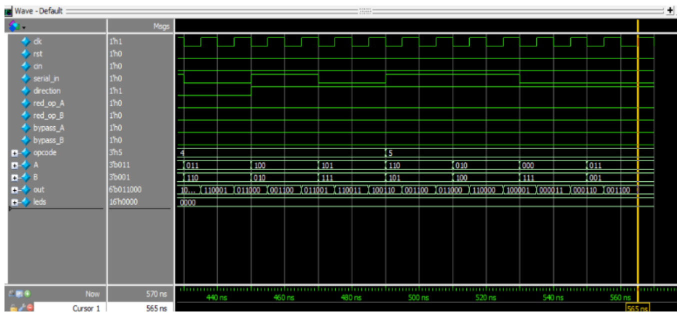
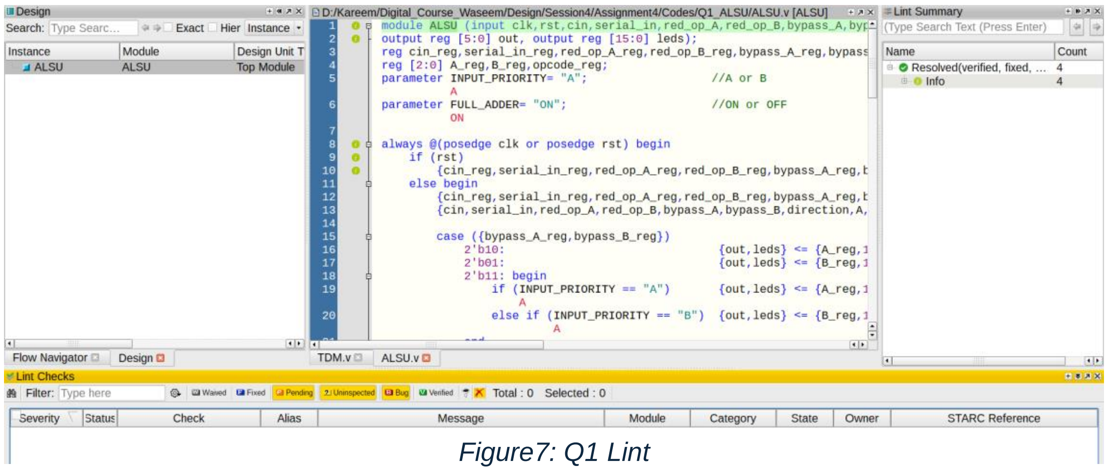


Figure6: Q1 Wave4



D:/Kareem/Digital_Course_Waseem/Design/Session4/Assignment4/Codes/Q1_ALSU/ALSU.v [ALSU]

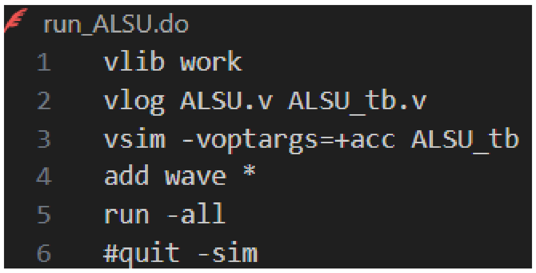
```
module ALSU (input clk,rst,cin,serial_in,red_op_A,red_op_B,bypass_A,byp
output reg [5:0] out, output reg [15:0] leds);
reg cin_reg,serial_in_reg,red_op_A_reg,red_op_B_reg,bypass_A_reg,bypass
reg [2:0] A_reg,B_reg,opcode_reg;
parameter INPUT_PRIORITY= "A";                    //A or B
          A
parameter FULL_ADDER= "ON";                       //ON or OFF
          ON

always @(posedge clk or posedge rst) begin
    if (rst)
        {cin_reg,serial_in_reg,red_op_A_reg,red_op_B_reg,bypass_A_reg,b
    else begin
        {cin_reg,serial_in_reg,red_op_A_reg,red_op_B_reg,bypass_A_reg,b
        {cin,serial_in,red_op_A,red_op_B,bypass_A,bypass_B,direction,A,

        case ({bypass_A_reg,bypass_B_reg})
            2'b10:                                {out,leds} <= {A_reg,1
            2'b01:                                {out,leds} <= {B_reg,1
            2'b11: begin
                if (INPUT_PRIORITY == "A")        {out,leds} <= {A_reg,1
                    A
                else if (INPUT_PRIORITY == "B")  {out,leds} <= {B_reg,1
                         A
```

Name	Count
Resolved(verified, fixed, ...	4
Info	4

Lint Checks — Total : 0 Selected : 0

Severity	Status	Check	Alias	Message	Module	Category	State	Owner	STARC Reference

Figure7: Q1 Lint



run_ALSU.do

```
vlib work
vlog ALSU.v ALSU_tb.v
vsim -voptargs=+acc ALSU_tb
add wave *
run -all
#quit -sim
```

Figure8: Q1 DO File

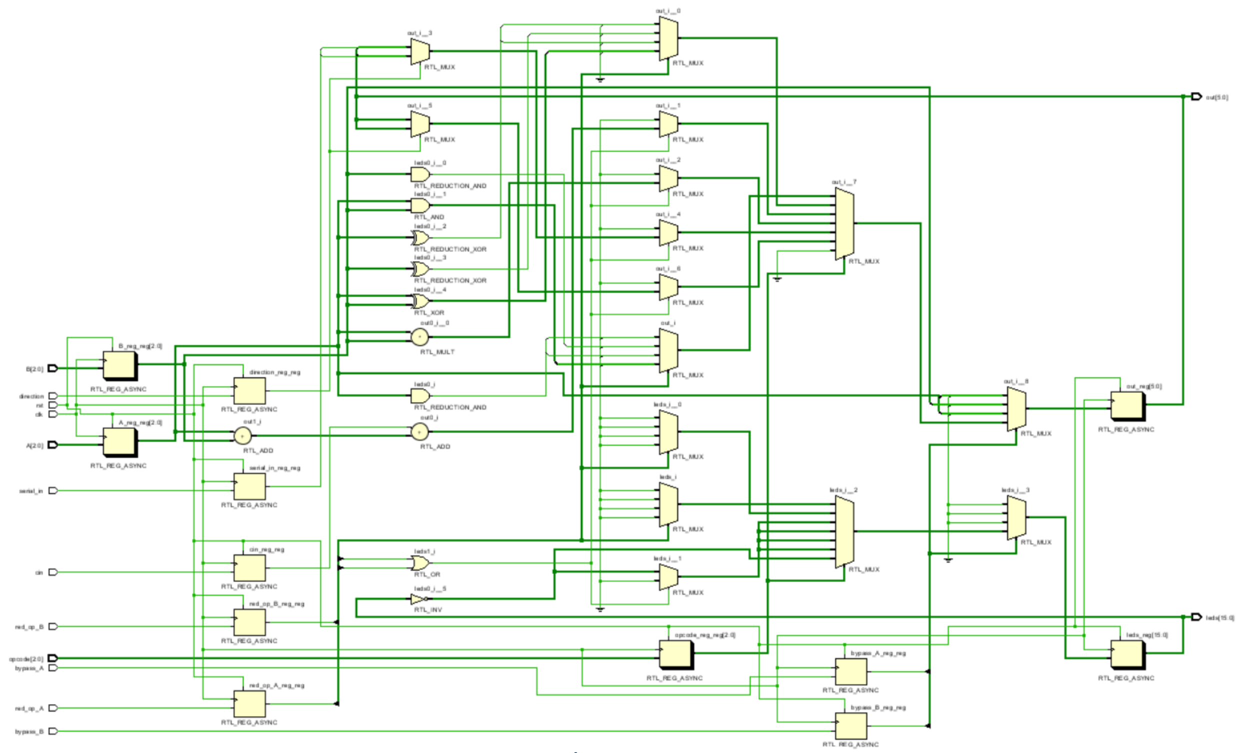


Figure9: Q1 RTL

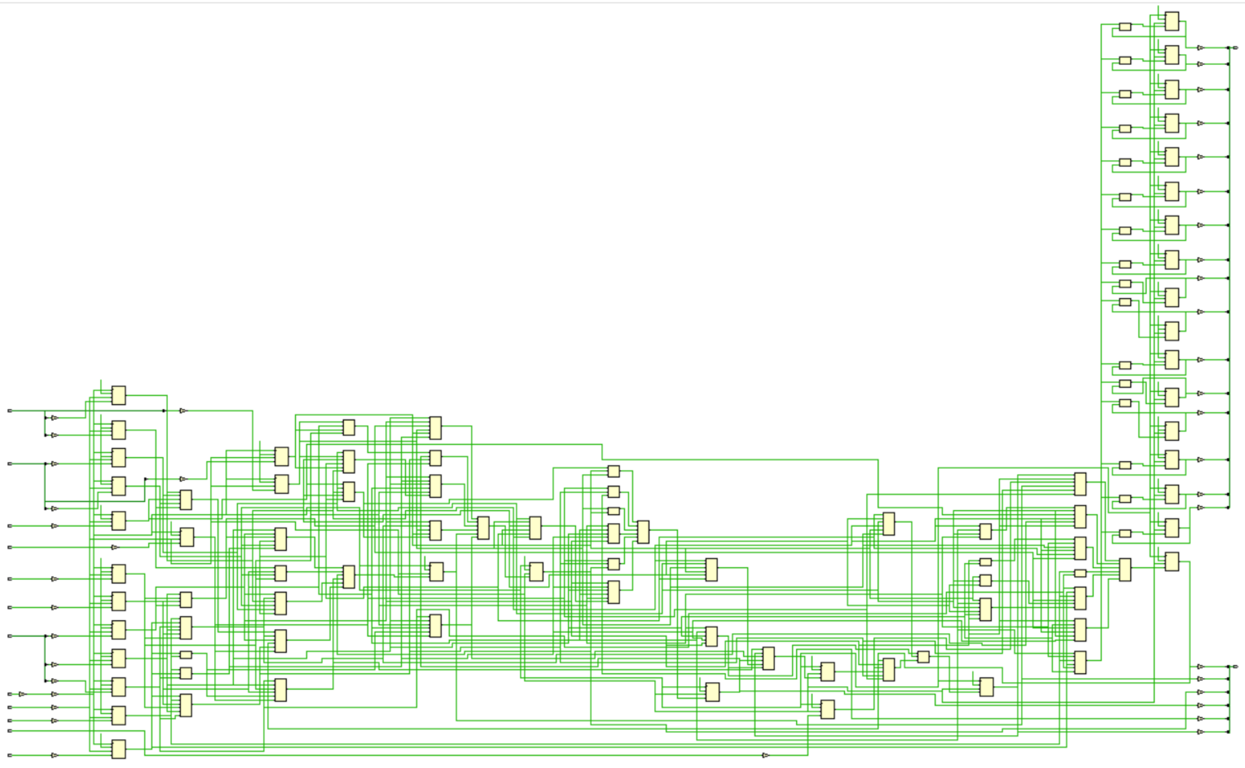


Figure10: Q1 Synthesis

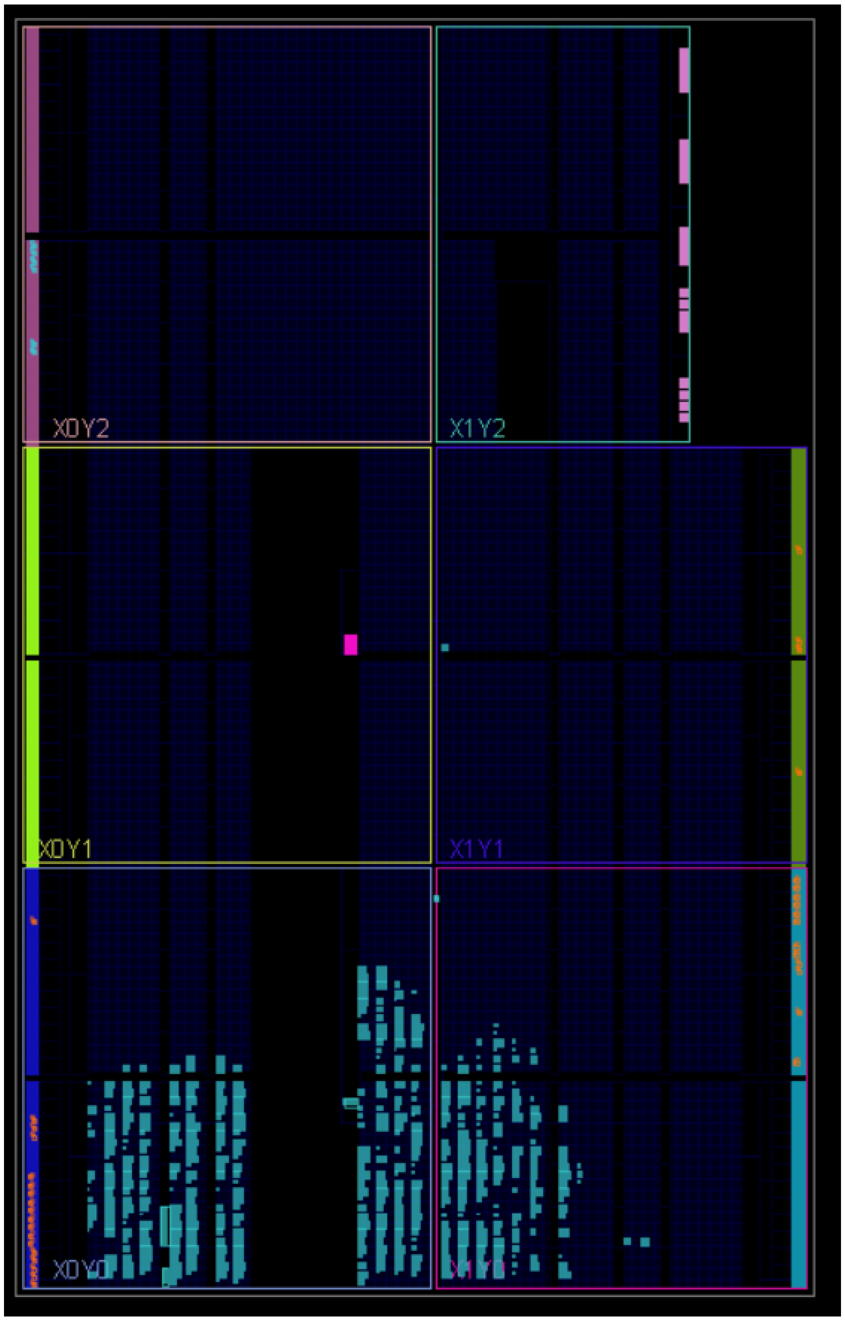
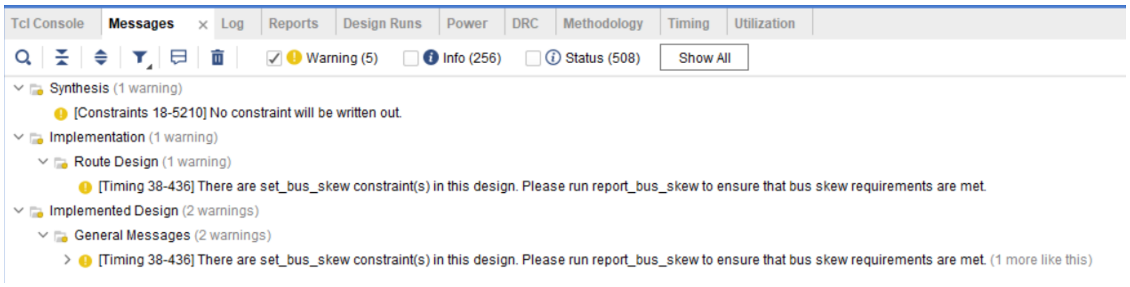
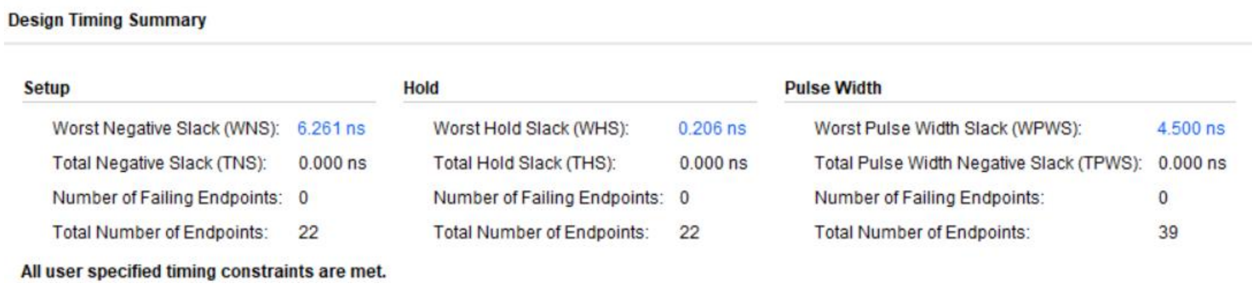


Figure11: Q1 Device



```
Synthesis (1 warning)
    [Constraints 18-5210] No constraint will be written out.
Implementation (1 warning)
    Route Design (1 warning)
        [Timing 38-436] There are set_bus_skew constraint(s) in this design. Please run report_bus_skew to ensure that bus skew requirements are met.
Implemented Design (2 warnings)
    General Messages (2 warnings)
        [Timing 38-436] There are set_bus_skew constraint(s) in this design. Please run report_bus_skew to ensure that bus skew requirements are met. (1 more like this)
```

Figure12: Q1 Messages

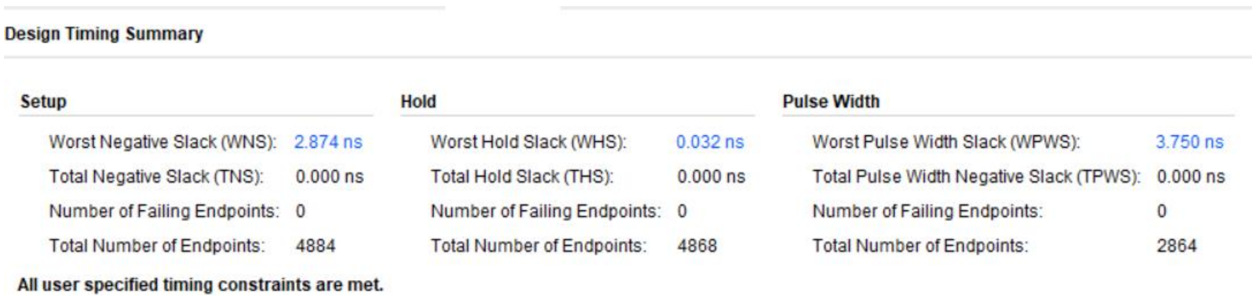


Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.261 ns	Worst Hold Slack (WHS):	0.206 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:	22	Total Number of Endpoints:	22	Total Number of Endpoints:	39

All user specified timing constraints are met.

Figure13: Q1 Synthesis Timing Report

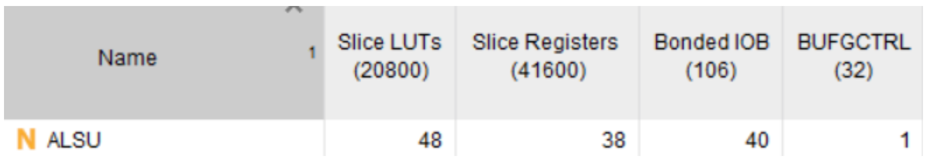


Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.874 ns	Worst Hold Slack (WHS):	0.032 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:	4884	Total Number of Endpoints:	4868	Total Number of Endpoints:	2864

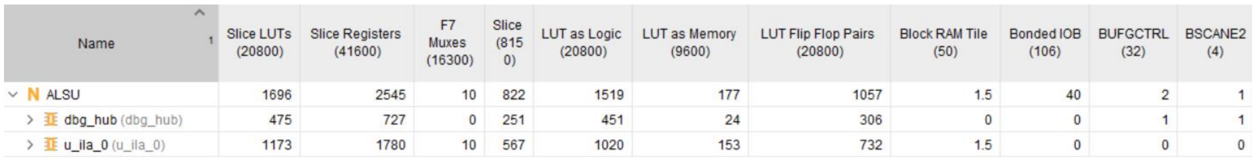
All user specified timing constraints are met.

Figure14: Q1 Implementation Timing Report



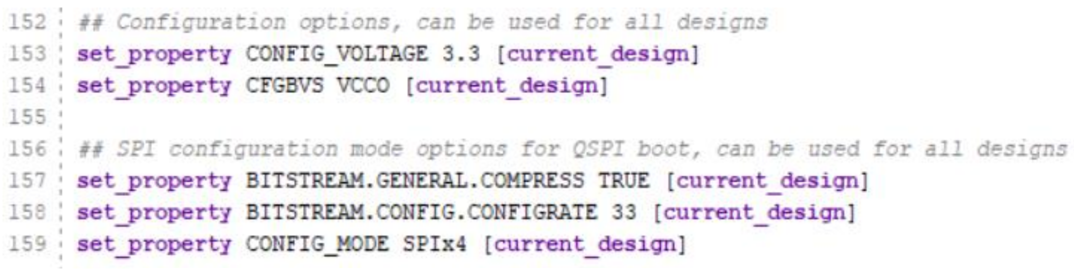
Name	Slice LUTs (20800)	Slice Registers (41600)	Bonded IOB (106)	BUFGCTRL (32)
ALSU	48	38	40	1

Figure15: Q1 Synthesis Utilization Report



Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Slice (8150)	LUT as Logic (20800)	LUT as Memory (9600)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)	BSCANE2 (4)
ALSU	1696	2545	10	822	1519	177	1057	1.5	40	2	1
dbg_hub (dbg_hub)	475	727	0	251	451	24	306	0	0	1	1
u_ila_0 (u_ila_0)	1173	1780	10	567	1020	153	732	1.5	0	0	0

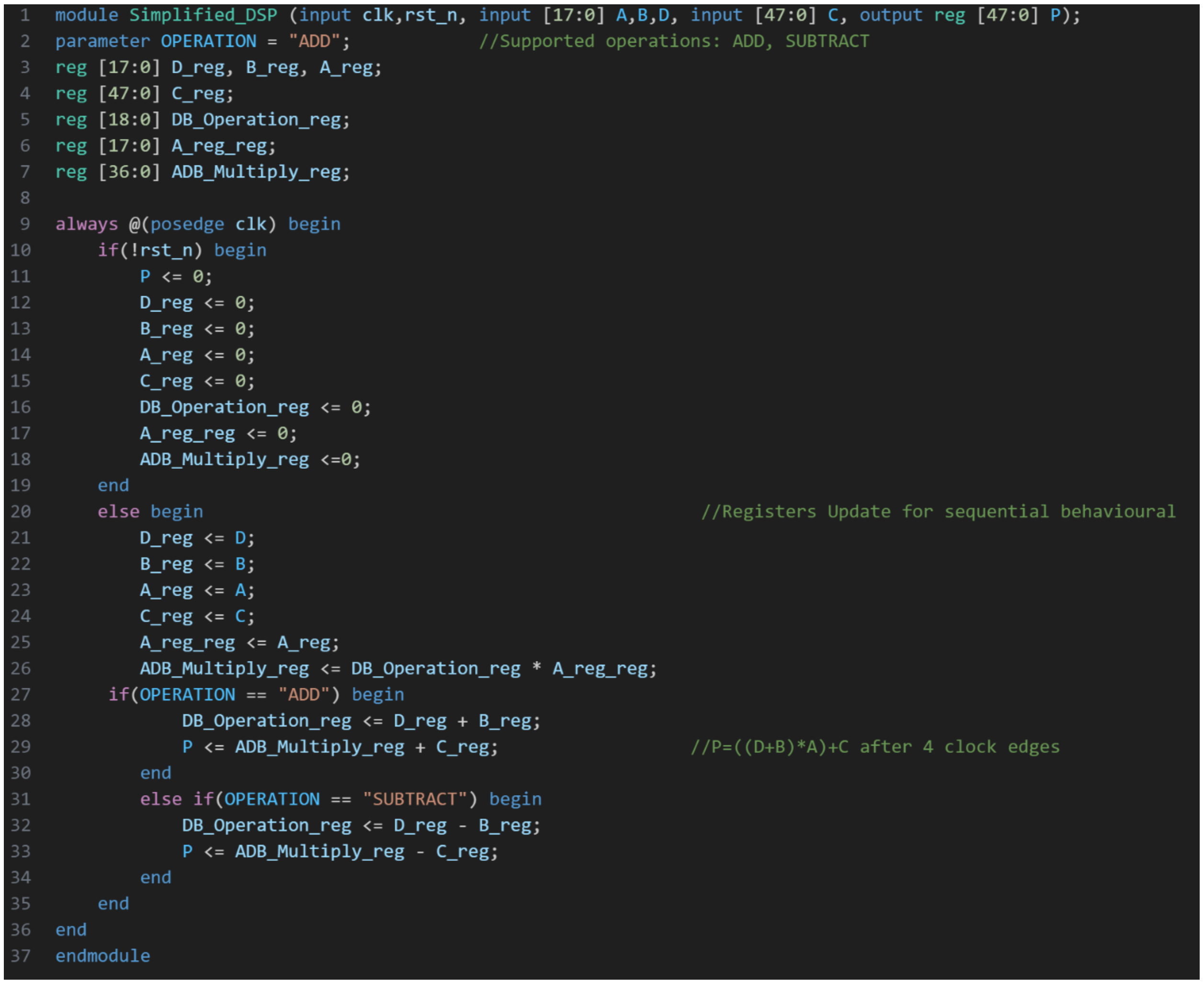
Figure16: Q1 Implementation Utilization Report



```
## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]
```

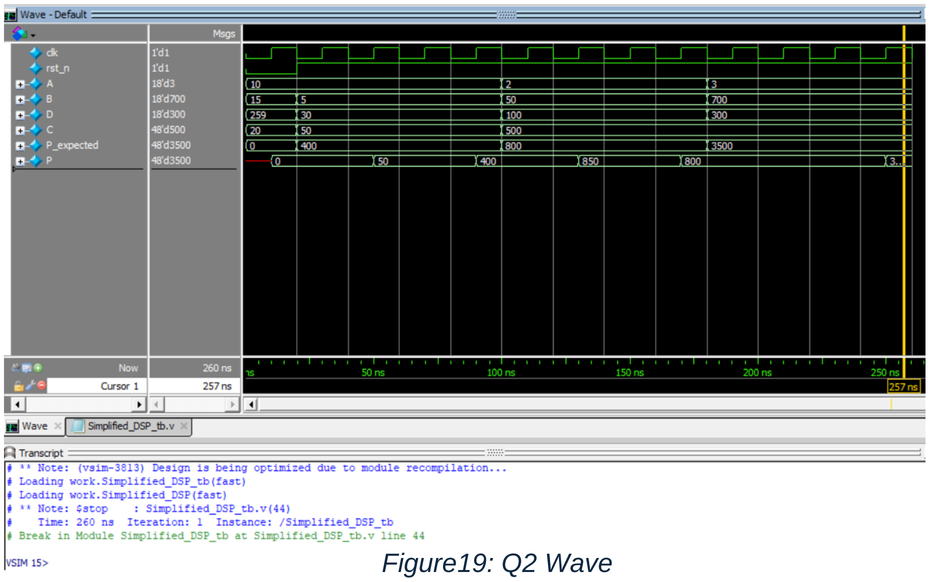
Figure17: Q1 Constraints File

2) Simplified DSP:



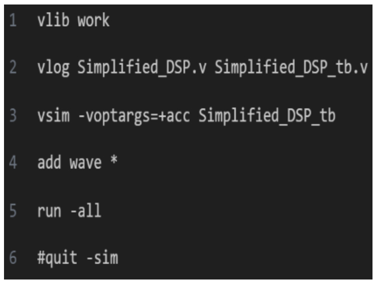
```verilog
module Simplified_DSP (input clk,rst_n, input [17:0] A,B,D, input [47:0] C, output reg [47:0] P);
parameter OPERATION = "ADD";            //Supported operations: ADD, SUBTRACT
reg [17:0] D_reg, B_reg, A_reg;
reg [47:0] C_reg;
reg [18:0] DB_Operation_reg;
reg [17:0] A_reg_reg;
reg [36:0] ADB_Multiply_reg;

always @(posedge clk) begin
    if(!rst_n) begin
        P <= 0;
        D_reg <= 0;
        B_reg <= 0;
        A_reg <= 0;
        C_reg <= 0;
        DB_Operation_reg <= 0;
        A_reg_reg <= 0;
        ADB_Multiply_reg <=0;
    end
    else begin                                   //Registers Update for sequential behavioural
        D_reg <= D;
        B_reg <= B;
        A_reg <= A;
        C_reg <= C;
        A_reg_reg <= A_reg;
        ADB_Multiply_reg <= DB_Operation_reg * A_reg_reg;
     if(OPERATION == "ADD") begin
            DB_Operation_reg <= D_reg + B_reg;
            P <= ADB_Multiply_reg + C_reg;                //P=((D+B)*A)+C after 4 clock edges
        end
        else if(OPERATION == "SUBTRACT") begin
            DB_Operation_reg <= D_reg - B_reg;
            P <= ADB_Multiply_reg - C_reg;
        end
    end
end
endmodule
```

Figure18: Q2 Code



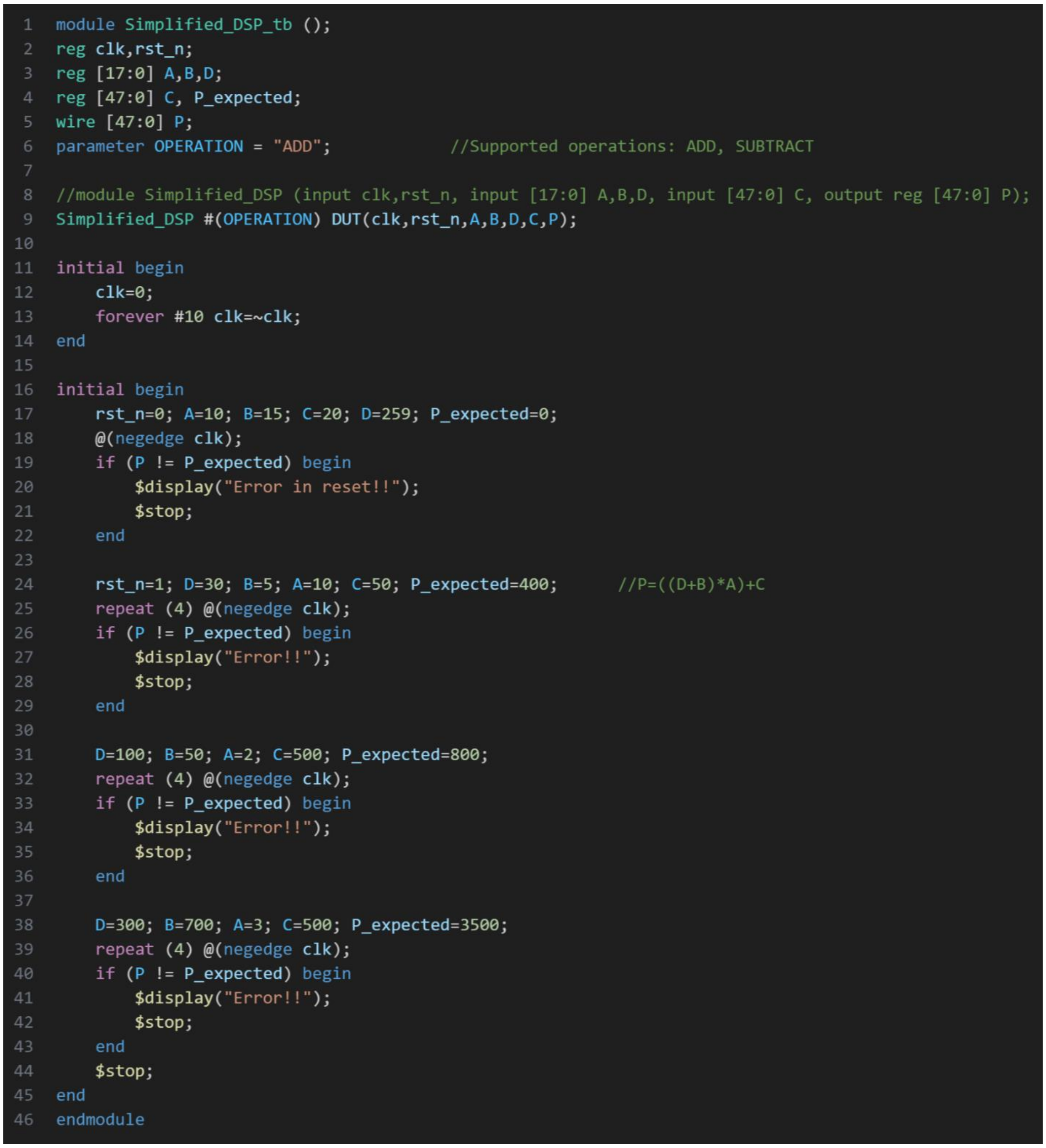
```
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading work.Simplified_DSP_tb(fast)
# Loading work.Simplified_DSP(fast)
# ** Note: $stop    : Simplified_DSP_tb.v(44)
#    Time: 260 ns  Iteration: 1  Instance: /Simplified_DSP_tb
# Break in Module Simplified_DSP_tb at Simplified_DSP_tb.v line 44

VSIM 15>
```

Figure19: Q2 Wave



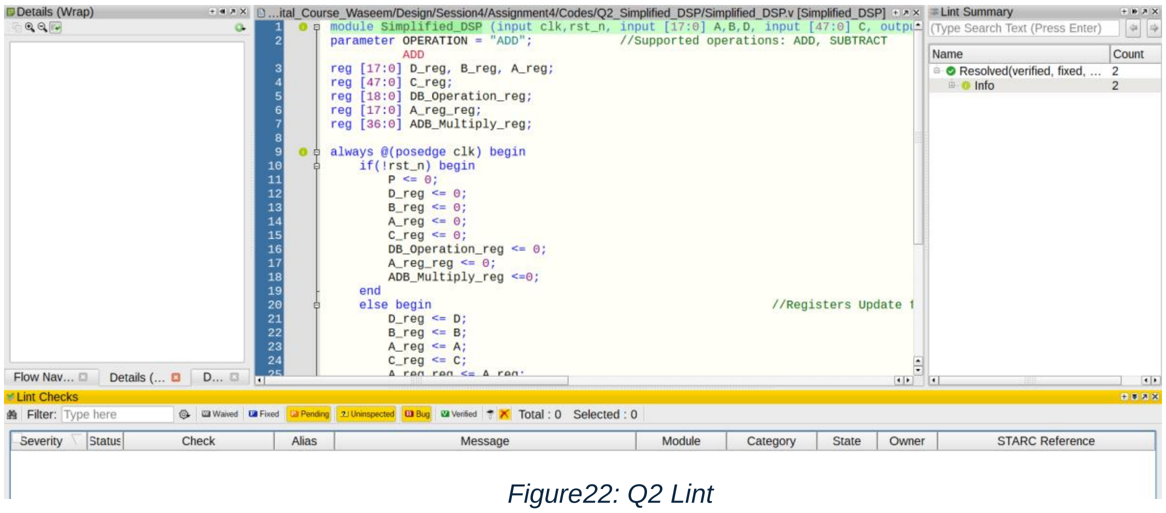
```
vlib work
vlog Simplified_DSP.v Simplified_DSP_tb.v
vsim -voptargs=+acc Simplified_DSP_tb
add wave *
run -all
#quit -sim
```

Figure20: Q2 Do File



```verilog
module Simplified_DSP_tb ();
reg clk,rst_n;
reg [17:0] A,B,D;
reg [47:0] C, P_expected;
wire [47:0] P;
parameter OPERATION = "ADD";            //Supported operations: ADD, SUBTRACT

//module Simplified_DSP (input clk,rst_n, input [17:0] A,B,D, input [47:0] C, output reg [47:0] P);
Simplified_DSP #(OPERATION) DUT(clk,rst_n,A,B,D,C,P);

initial begin
    clk=0;
    forever #10 clk=~clk;
end

initial begin
    rst_n=0; A=10; B=15; C=20; D=259; P_expected=0;
    @(negedge clk);
    if (P != P_expected) begin
        $display("Error in reset!!");
        $stop;
    end

    rst_n=1; D=30; B=5; A=10; C=50; P_expected=400;      //P=((D+B)*A)+C
    repeat (4) @(negedge clk);
    if (P != P_expected) begin
        $display("Error!!");
        $stop;
    end

    D=100; B=50; A=2; C=500; P_expected=800;
    repeat (4) @(negedge clk);
    if (P != P_expected) begin
        $display("Error!!");
        $stop;
    end

    D=300; B=700; A=3; C=500; P_expected=3500;
    repeat (4) @(negedge clk);
    if (P != P_expected) begin
        $display("Error!!");
        $stop;
    end
    $stop;
end
endmodule
```

Figure21: Q2 Testbench



Details (Wrap)

…ital_Course_Waseem/Design/Session4/Assignment4/Codes/Q2_Simplified_DSP/Simplified_DSP.v [Simplified_DSP]

```
module Simplified_DSP (input clk,rst_n, input [17:0] A,B,D, input [47:0] C, outpu
parameter OPERATION = "ADD";          //Supported operations: ADD, SUBTRACT
          ADD
reg [17:0] D_reg, B_reg, A_reg;
reg [47:0] C_reg;
reg [18:0] DB_Operation_reg;
reg [17:0] A_reg_reg;
reg [36:0] ADB_Multiply_reg;

always @(posedge clk) begin
    if(!rst_n) begin
        P <= 0;
        D_reg <= 0;
        B_reg <= 0;
        A_reg <= 0;
        C_reg <= 0;
        DB_Operation_reg <= 0;
        A_reg_reg <= 0;
        ADB_Multiply_reg <=0;
    end
    else begin                                          //Registers Update f
        D_reg <= D;
        B_reg <= B;
        A_reg <= A;
        C_reg <= C;
        A_reg_reg <= A_reg;
```

Lint Summary

(Type Search Text (Press Enter)

Name	Count
Resolved(verified, fixed, …	2
Info	2

Flow Nav… Details (… D…

Lint Checks

Filter: Type here Waived Fixed Pending Uninspected Bug Verified Total : 0 Selected : 0

Severity	Status	Check	Alias	Message	Module	Category	State	Owner	STARC Reference

Figure22: Q2 Lint

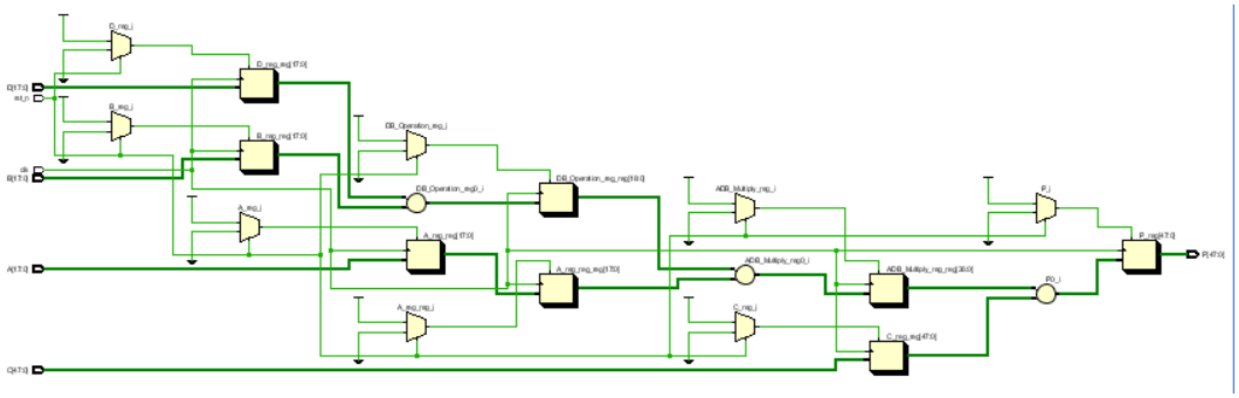


Figure23: Q2 RTL

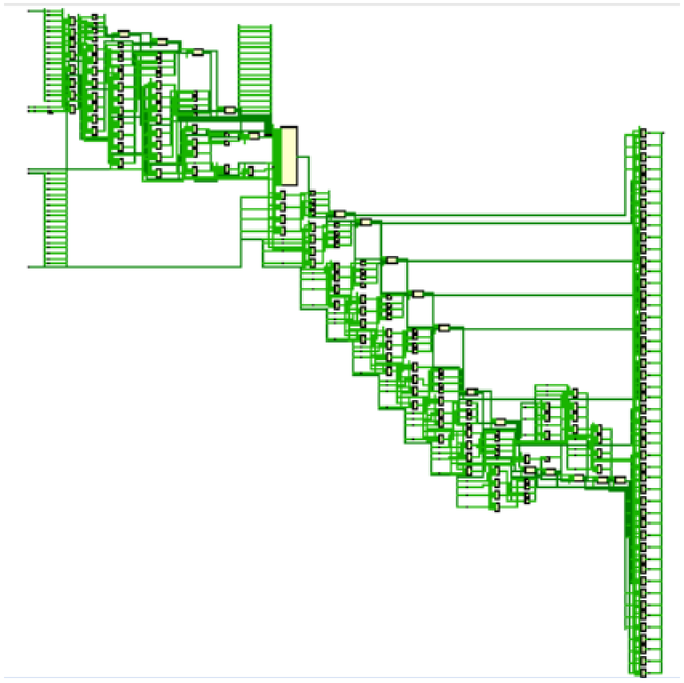


Figure24: Q2 Synthesis

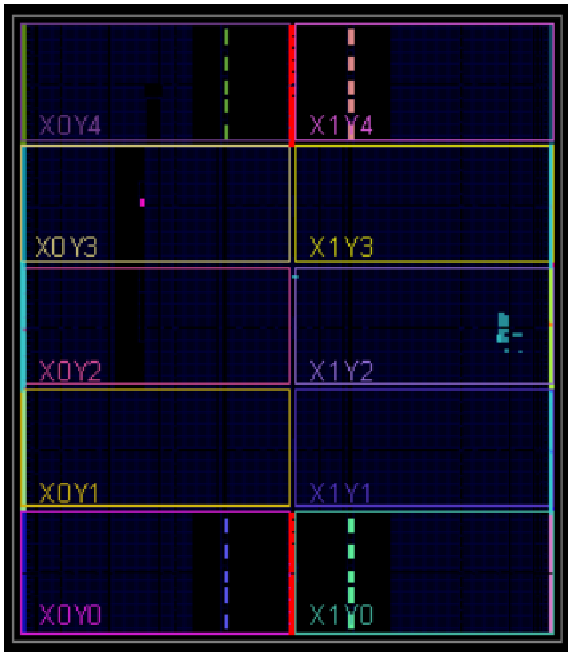


Figure25: Q2 Device1

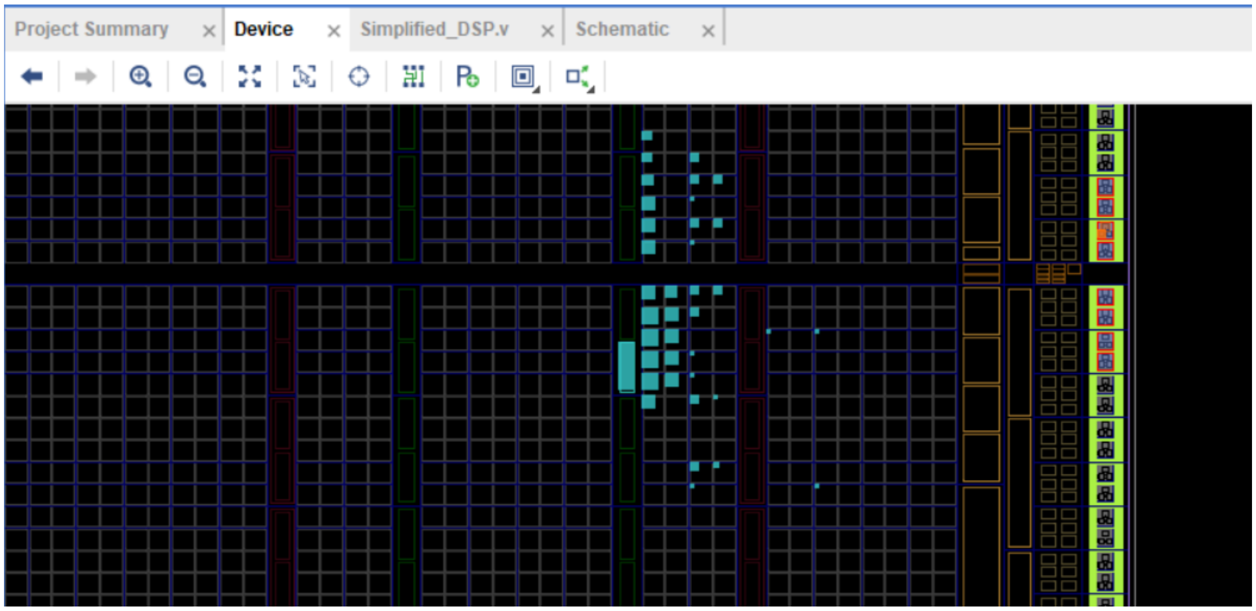
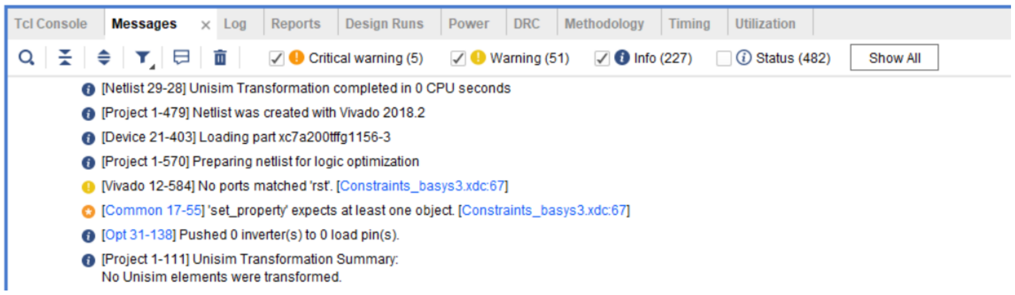
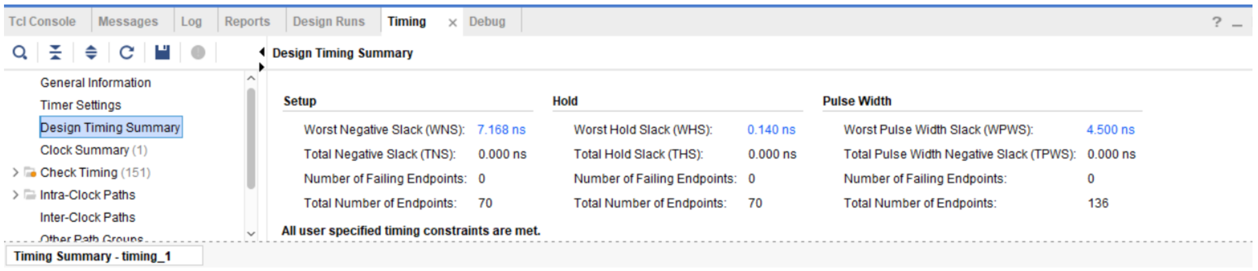


Figure26: Q2 Device2



```
[Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
[Project 1-479] Netlist was created with Vivado 2018.2
[Device 21-403] Loading part xc7a200tffg1156-3
[Project 1-570] Preparing netlist for logic optimization
[Vivado 12-584] No ports matched 'rst'. [Constraints_basys3.xdc:67]
[Common 17-55] 'set_property' expects at least one object. [Constraints_basys3.xdc:67]
[Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
[Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.
```

Figure27: Q2 Messages

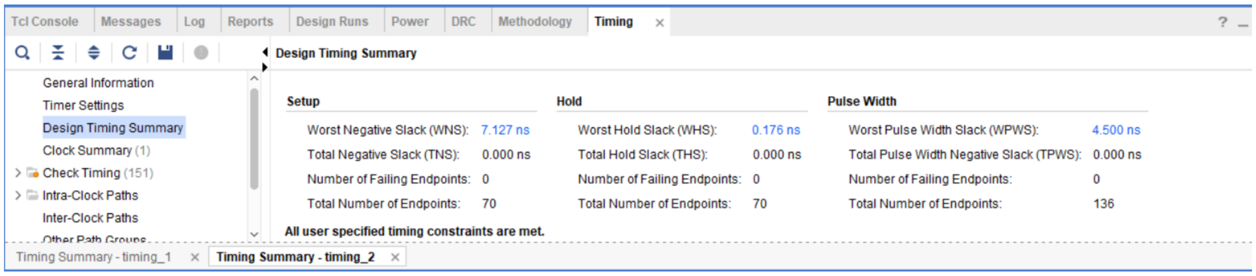


Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	7.168 ns	Worst Hold Slack (WHS):	0.140 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:	70	Total Number of Endpoints:	70	Total Number of Endpoints:	136

All user specified timing constraints are met.

Figure28: Q2 Synthesis Timing Report

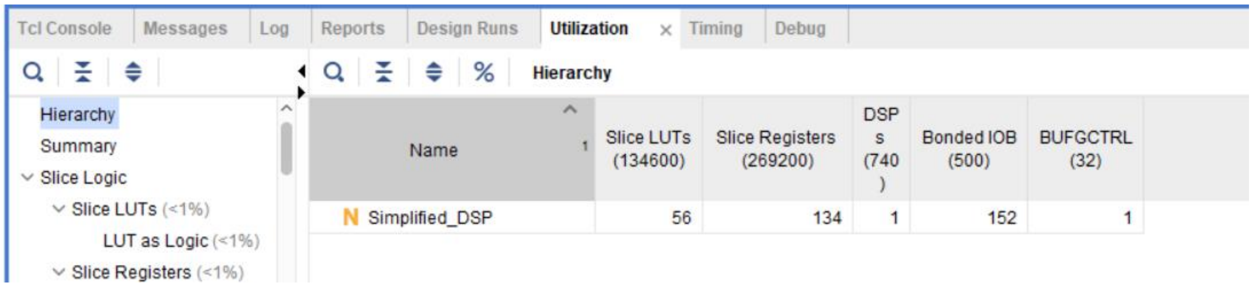


Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	7.127 ns	Worst Hold Slack (WHS):	0.176 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:	70	Total Number of Endpoints:	70	Total Number of Endpoints:	136

All user specified timing constraints are met.

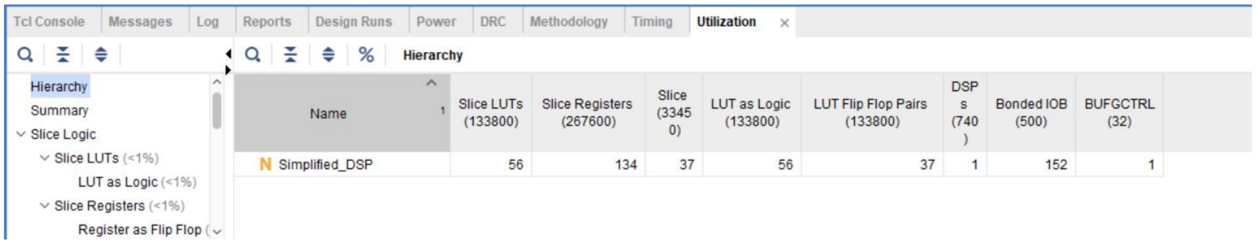
Figure29: Q2 Implementation Timing Report



Hierarchy

Name	Slice LUTs (134600)	Slice Registers (269200)	DSPs (740)	Bonded IOB (500)	BUFGCTRL (32)
N Simplified_DSP	56	134	1	152	1

Figure30: Q2 Synthesis Utilization Report

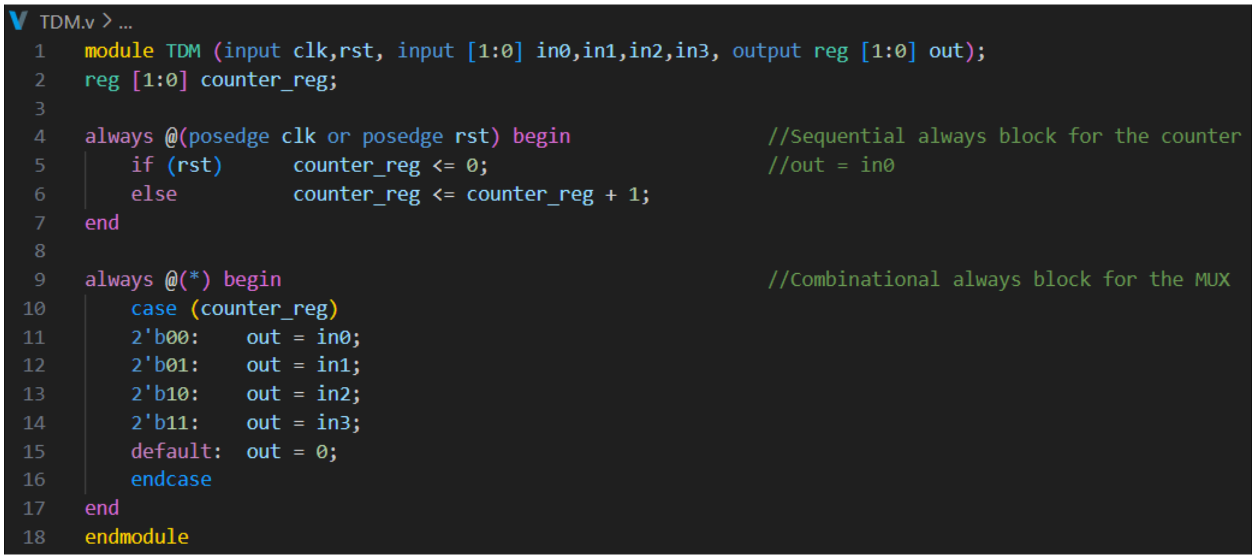


Hierarchy

Name	Slice LUTs (133800)	Slice Registers (267600)	Slice (33450)	LUT as Logic (133800)	LUT Flip Flop Pairs (133800)	DSPs (740)	Bonded IOB (500)	BUFGCTRL (32)
N Simplified_DSP	56	134	37	56	37	1	152	1

Figure31: Q2 Implementation Utilization Report

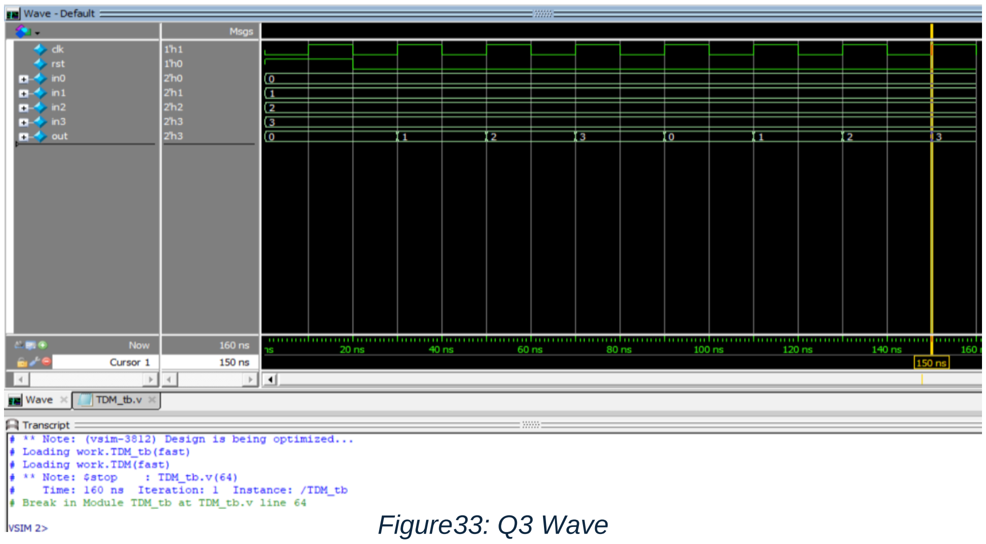
3) TDM:



TDM.v > ...

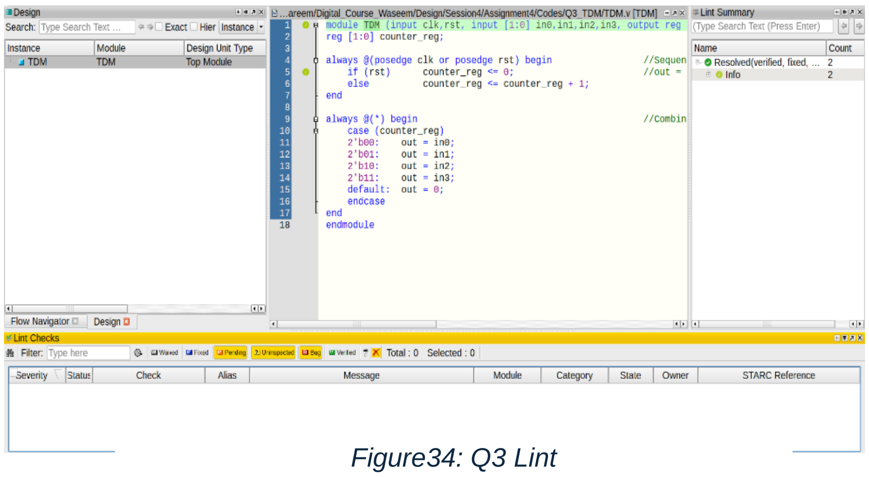
```verilog
module TDM (input clk,rst, input [1:0] in0,in1,in2,in3, output reg [1:0] out);
reg [1:0] counter_reg;

always @(posedge clk or posedge rst) begin             //Sequential always block for the counter
    if (rst)       counter_reg <= 0;                   //out = in0
    else           counter_reg <= counter_reg + 1;
end

always @(*) begin                                      //Combinational always block for the MUX
    case (counter_reg)
    2'b00:     out = in0;
    2'b01:     out = in1;
    2'b10:     out = in2;
    2'b11:     out = in3;
    default:   out = 0;
    endcase
end
endmodule
```

Figure32: Q3 Code



```
# ** Note: (vsim-3812) Design is being optimized...
# Loading work.TDM_tb(fast)
# Loading work.TDM(fast)
# ** Note: $stop    : TDM_tb.v(64)
#    Time: 160 ns  Iteration: 1  Instance: /TDM_tb
# Break in Module TDM_tb at TDM_tb.v line 64

VSIM 2>
```

Figure33: Q3 Wave



Design
Search: Type Search Text ... Exact Hier Instance
Flow Navigator
Design
...areem/Digital_Course_Waseem/Design/Session4/Assignment4/Codes/Q3_TDM/TDM.v [TDM]
Lint Summary
(Type Search Text (Press Enter)
Lint Checks
Filter: Type here
Waived Fixed Pending Uninspected Bug Verified
Total : 0 Selected : 0

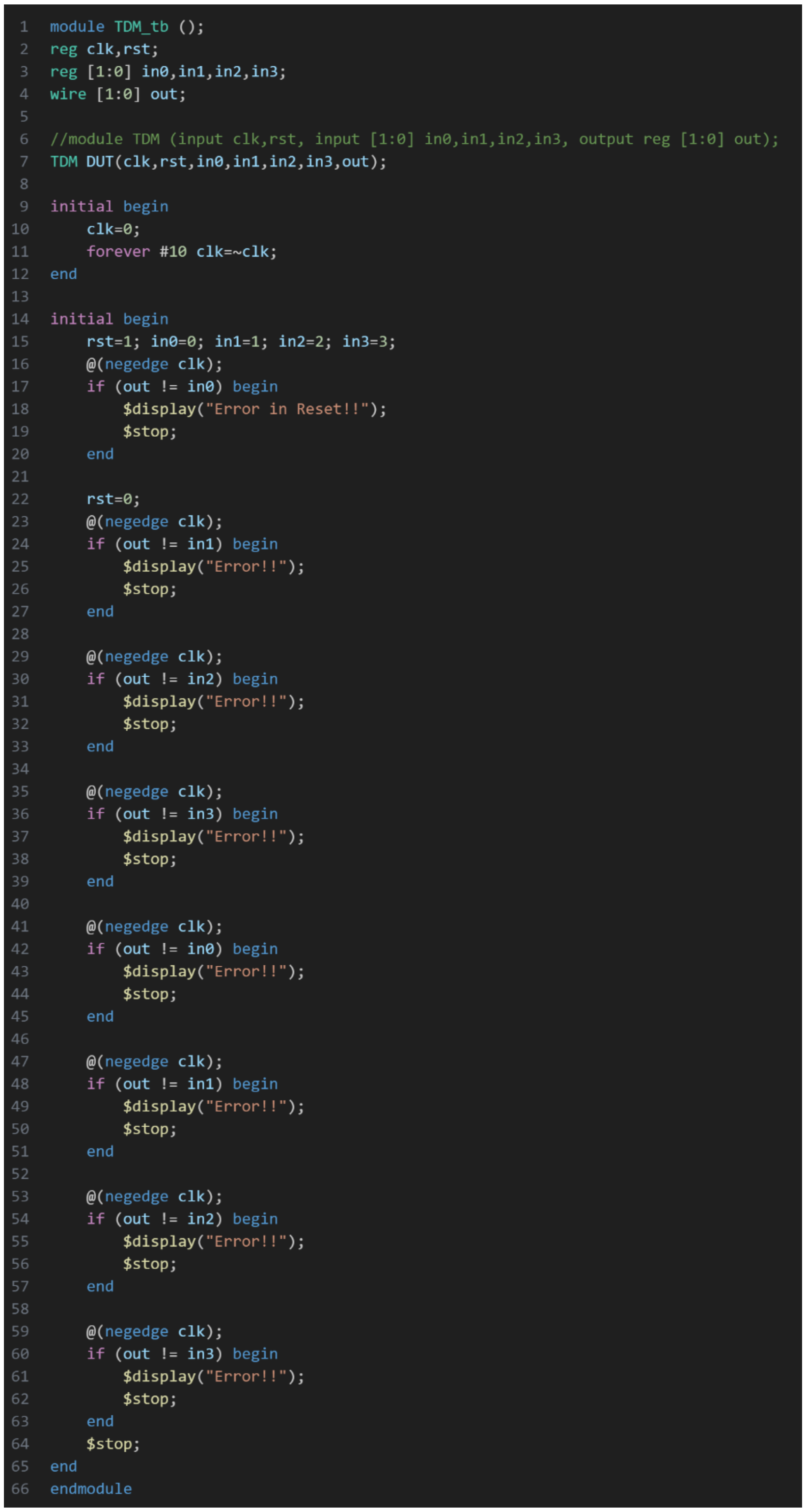
Instance	Module	Design Unit Type
TDM	TDM	Top Module

```verilog
module TDM (input clk,rst, input [1:0] in0,in1,in2,in3, output reg
reg [1:0] counter_reg;

always @(posedge clk or posedge rst) begin             //Sequen
    if (rst)       counter_reg <= 0;                   //out =
    else           counter_reg <= counter_reg + 1;
end

always @(*) begin                                      //Combin
    case (counter_reg)
    2'b00:     out = in0;
    2'b01:     out = in1;
    2'b10:     out = in2;
    2'b11:     out = in3;
    default:   out = 0;
    endcase
end
endmodule
```

Name	Count
Resolved(verified, fixed, ...	2
Info	2

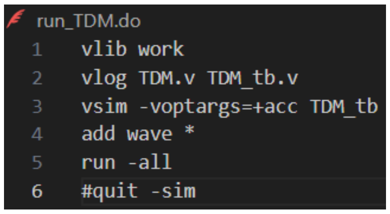
Severity	Status	Check	Alias	Message	Module	Category	State	Owner	STARC Reference

Figure34: Q3 Lint



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

//module TDM (input clk,rst, input [1:0] in0,in1,in2,in3, output reg [1:0] out);
TDM DUT(clk,rst,in0,in1,in2,in3,out);

initial begin
    clk=0;
    forever #10 clk=~clk;
end

initial begin
    rst=1; in0=0; in1=1; in2=2; in3=3;
    @(negedge clk);
    if (out != in0) begin
        $display("Error in Reset!!");
        $stop;
    end

    rst=0;
    @(negedge clk);
    if (out != in1) begin
        $display("Error!!");
        $stop;
    end

    @(negedge clk);
    if (out != in2) begin
        $display("Error!!");
        $stop;
    end

    @(negedge clk);
    if (out != in3) begin
        $display("Error!!");
        $stop;
    end

    @(negedge clk);
    if (out != in0) begin
        $display("Error!!");
        $stop;
    end

    @(negedge clk);
    if (out != in1) begin
        $display("Error!!");
        $stop;
    end

    @(negedge clk);
    if (out != in2) begin
        $display("Error!!");
        $stop;
    end

    @(negedge clk);
    if (out != in3) begin
        $display("Error!!");
        $stop;
    end
    $stop;
end
endmodule
```

Figure35: Q3 Testbench



run_TDM.do

```
vlib work
vlog TDM.v TDM_tb.v
vsim -voptargs=+acc TDM_tb
add wave *
run -all
#quit -sim
```

Figure36: Q3 DO File

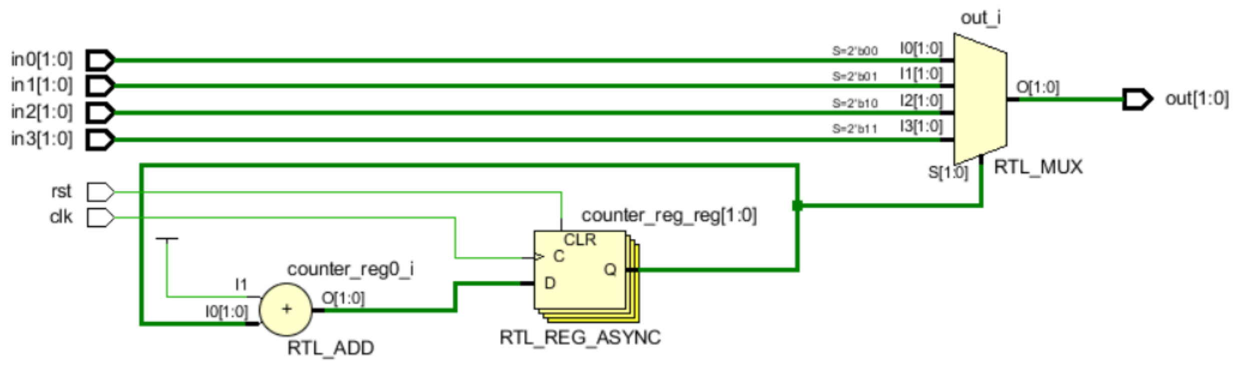


Figure37: Q3 RTL

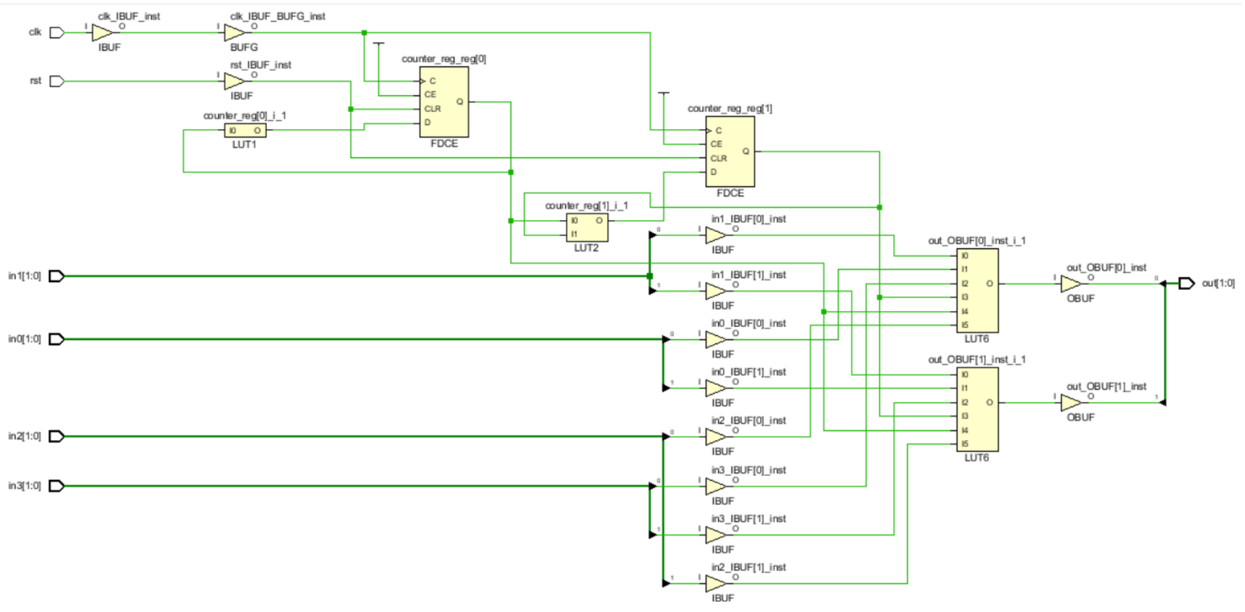
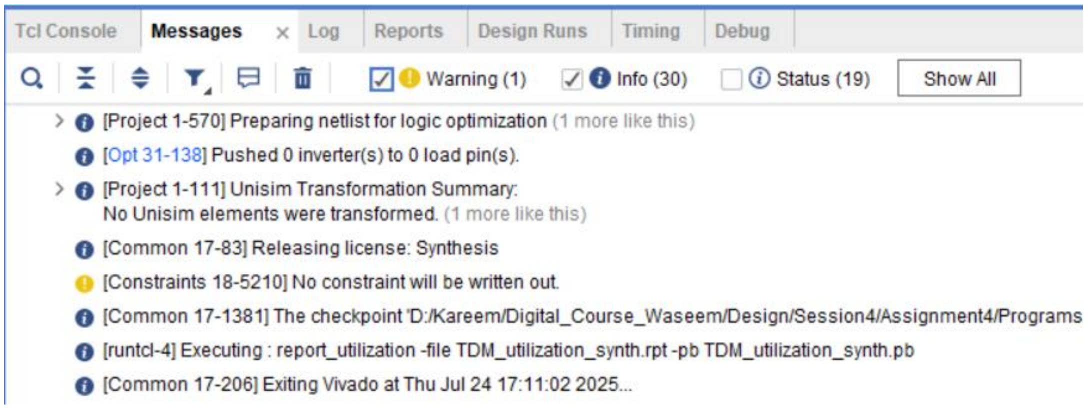


Figure38: Q3 Synthesis



```
[Project 1-570] Preparing netlist for logic optimization (1 more like this)
[Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
[Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed. (1 more like this)
[Common 17-83] Releasing license: Synthesis
[Constraints 18-5210] No constraint will be written out.
[Common 17-1381] The checkpoint 'D:/Kareem/Digital_Course_Waseem/Design/Session4/Assignment4/Programs
[runtcl-4] Executing : report_utilization -file TDM_utilization_synth.rpt -pb TDM_utilization_synth.pb
[Common 17-206] Exiting Vivado at Thu Jul 24 17:11:02 2025...
```

Figure39: Q3 Messages

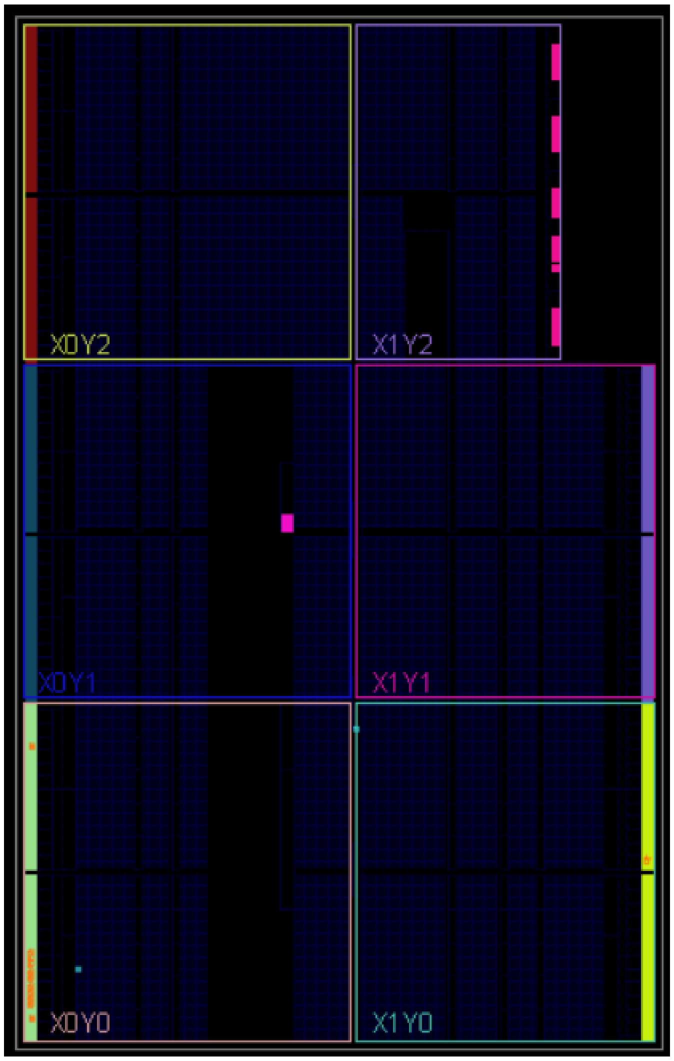


Figure40: Q3 Device1

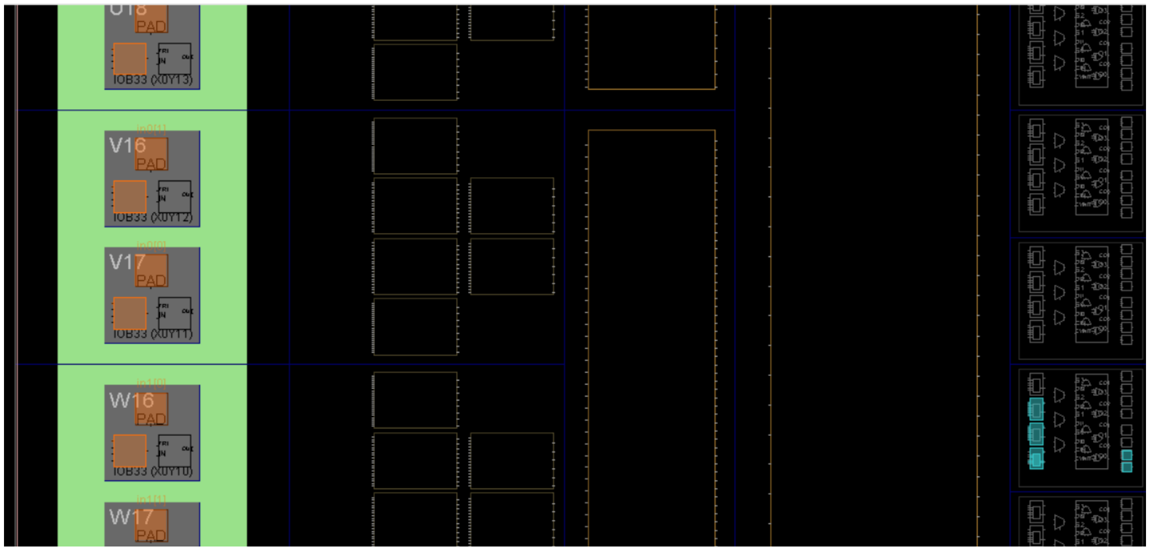
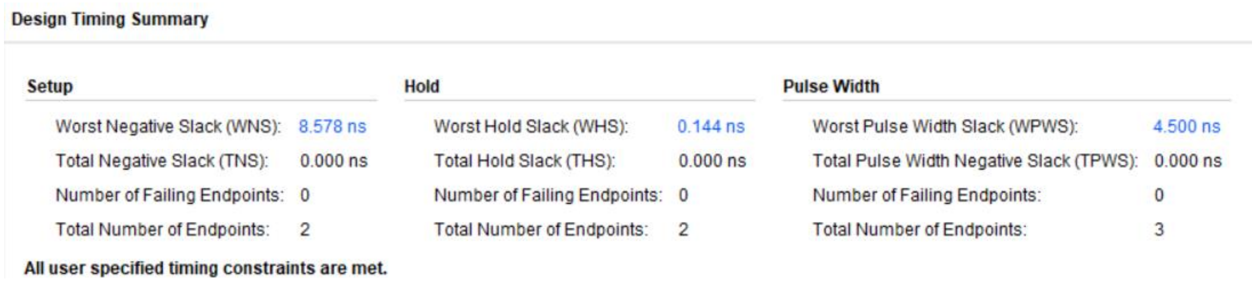


Figure41: Q3 Device2

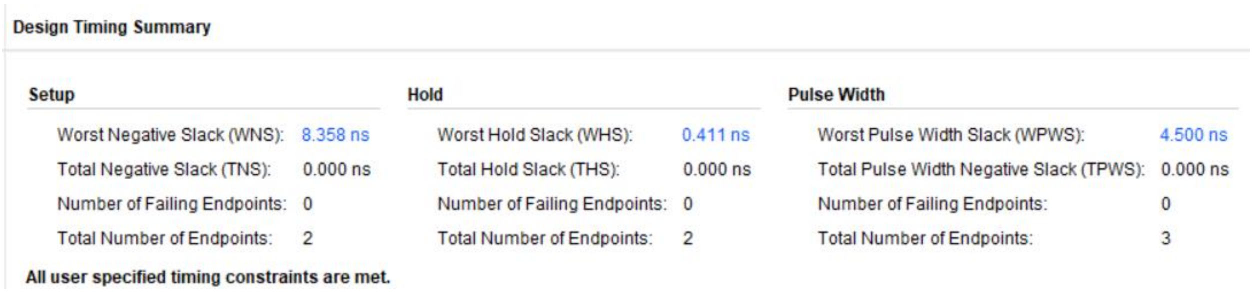


Design Timing Summary

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

All user specified timing constraints are met.

Figure42: Q3 Synthesis Timing Report

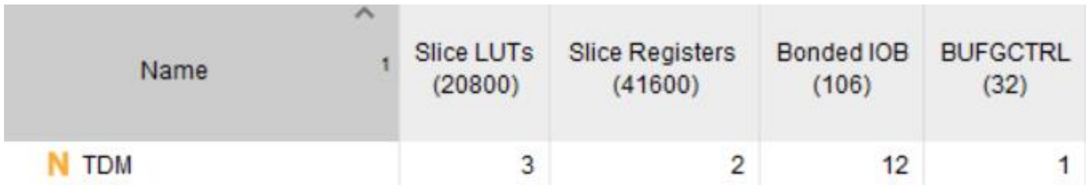


Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	8.358 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

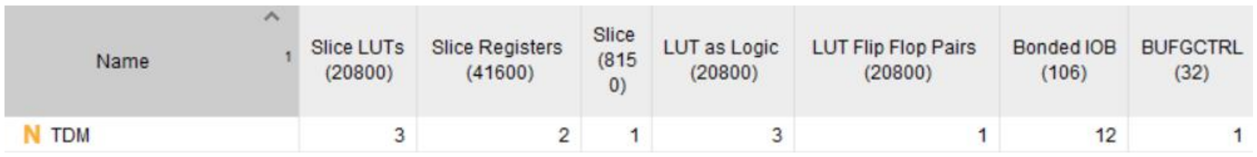
All user specified timing constraints are met.

Figure43: Q3 Implementation Timing Report



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

Figure44: Q3 Synthesis Utilization Report



Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Bonded IOB (106)	BUFGCTRL (32)
N TDM	3	2	1	3	1	12	1

Figure45: Q3 Implementation Utilization Report