Assignment 4

Ameer Ashraf Louly – G3

Q1: ALSU

RTL Code:

```
module alsu_seq(A, B, opcode, cin, serial_in, direction, red_op_A, red_op_B,
bypass_A, bypass_B, clk, rst,  out, leds);
    output reg [5:0] out;
    output reg [15:0] leds;
    parameter INPUT_PRIORITY = "A";
    parameter FULL_ADDER = "ON";
    input [2:0] A;
    input [2:0] B;
    input [2:0] opcode;
    input   cin,
            serial_in,
            direction,
            red_op_A,
            red_op_B,
            bypass_A,
            bypass_B,
            clk,
            rst;

    reg Invalid_Case;

    reg cin_reg;
    reg serial_in_reg;
    reg direction_reg;
    reg red_op_A_reg;
    reg red_op_B_reg;
    reg bypass_A_reg;
    reg bypass_B_reg;

    reg [2:0] A_reg;
    reg [2:0] B_reg;
    reg [2:0] opcode_reg;


    // Inputs Always Block
    always @(posedge clk or posedge rst) begin
        if(rst) begin
            cin_reg <= 0;
            serial_in_reg <= 0;
            direction_reg <= 0;
            red_op_A_reg <= 0;
            red_op_B_reg <= 0;
            bypass_A_reg <= 0;
            bypass_B_reg <= 0;
```

```verilog
            A_reg <= 0;
            B_reg <= 0;
            opcode_reg <= 0;
        end
        else begin
            cin_reg <= cin;
            serial_in_reg <= serial_in;
            direction_reg <= direction;
            red_op_A_reg <= red_op_A;
            red_op_B_reg <= red_op_B;
            bypass_A_reg <= bypass_A;
            bypass_B_reg <= bypass_B;
            A_reg <= A;
            B_reg <= B;
            opcode_reg <= opcode;
        end
    end

    // LEDs Always Block
    always @(posedge clk or posedge rst) begin
        if(rst)
            leds <= 0;
        else if(Invalid_Case == 1)
            leds <= ~leds;
        else
            leds <= 0;
    end

    always @(posedge clk or posedge rst) begin
        if(rst) begin
            out <= 0;
        end
        else if(bypass_A_reg || bypass_B_reg) begin
            if(bypass_A_reg && bypass_B_reg) begin
                if(INPUT_PRIORITY == "A")
                    out <= A_reg;
                if(INPUT_PRIORITY == "B")
                    out <= B_reg;
            end
            else if(bypass_A_reg)
                out <= A_reg;
            else if(bypass_B_reg)
                out <= B_reg;
        end
        else begin
            case(opcode_reg)
                3'h0: begin
                    if(red_op_A_reg == 1 && red_op_B_reg == 1) begin
```

```
                        if(INPUT_PRIORITY == "A")
                            out <= &A_reg;
                        else if(INPUT_PRIORITY == "B")
                            out <= &B_reg;
                    end else if (red_op_A_reg == 1)
                        out <= &A_reg;
                    else if(red_op_B_reg == 1)
                        out <= &B_reg;
                    else begin
                        out <= A_reg & B_reg;
                    end

                    Invalid_Case <= 0;
                end
                3'h1: begin
                    if(red_op_A_reg && red_op_B_reg) begin
                        if(INPUT_PRIORITY == "A")
                            out <= ^A_reg;
                        else if(INPUT_PRIORITY == "B")
                            out <= ^B_reg;
                    end
                    else if (red_op_A_reg)
                        out <= ^A_reg;
                    else if(red_op_B_reg)
                        out <= ^B_reg;
                    else begin
                        out <= A_reg ^ B_reg;
                    end

                    Invalid_Case <= 0;
                end
                3'h2: begin
                    if (red_op_A_reg || red_op_B_reg)
                        Invalid_Case <= 1;
                    else if(FULL_ADDER == "ON") begin
                        out <= A_reg + B_reg + cin_reg;
                        Invalid_Case <= 0;
                    end
                    else if (FULL_ADDER == "OFF") begin
                        out <= A_reg + B_reg;
                        Invalid_Case <= 0;
                    end
                end
                3'h3: begin
                    if (red_op_A_reg || red_op_B_reg)
                        Invalid_Case <= 1;
                    else begin
                        out <= A_reg * B_reg;
                        Invalid_Case <= 0;
                    end
                end
                3'h4: begin
                    if (red_op_A_reg || red_op_B_reg)
                        Invalid_Case <= 1;
                    else if(direction_reg) begin
                        out <= {out[4:0], serial_in_reg};
                        Invalid_Case <= 0;
                    end
                    else begin
                        out <= {serial_in_reg, out[5:1]};
                        Invalid_Case <= 0;
                    end
                end
                3'h5: begin
                    if (red_op_A_reg || red_op_B_reg)
                        Invalid_Case <= 1;
                    else if(direction_reg) begin
                        out <= {out[4:0], out[5]};
                        Invalid_Case <= 0;
                    end
                    else begin
                        out <= {out[0], out[5:1]};
                        Invalid_Case <= 0;
                    end
                end
                3'h6: Invalid_Case <= 1;
                3'h7: Invalid_Case <= 1;
            endcase
        end
    end

endmodule
```

Testbench Code:

```
module alsu_tb();

    reg [2:0] A, B, opcode;
    reg cin,
        serial_in,
        direction,
        red_op_A,
        red_op_B,
        bypass_A,
        bypass_B,
        clk,
        rst;
    reg [5:0] out_expected;
    wire [5:0] out;
    wire [15:0] leds;

    alsu_seq DUT(A,
             B,
             opcode,
             cin,
             serial_in,
             direction,
             red_op_A,
             red_op_B,
             bypass_A,
             bypass_B,
             clk,
             rst,
             out,
             leds);

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

    initial begin
        // Reset Functionality
        A = 0;
        B = 0;
        opcode = 0;
        cin = 0;
        serial_in = 0;
        direction = 0;
```

```verilog
        red_op_A = 0;
        red_op_B = 0;
        bypass_A = 0;
        bypass_B = 0;
        rst = 1;
        repeat(2) @(negedge clk)
        if(out != 0 && leds != 0) begin
            $display("Error - Reset");
            $exit;
        end

        // bypass functionality
        rst = 0;
        bypass_A = 1;
        bypass_B = 1;
        repeat(50) begin
            A = $random;
            B = $random;
            opcode =  $urandom_range(0, 5);
            out_expected = A;
            repeat(2) @(negedge clk);
            if(out != out_expected) begin
                $display("Error - Bypass");
                $exit;
            end
        end

        rst = 1;
        repeat(2) @(negedge clk)

        // Opcode = 0 functionality
        rst = 0;
        A = 0;
        B = 0;
        bypass_A = 0;
        bypass_B = 0;
        opcode = 0;
        repeat(2) @(negedge clk)
        repeat(50) begin
            A = $random;
            B = $random;
            red_op_A = $random;
            red_op_B = $random;
            if(red_op_A && red_op_B)
                out_expected = &A;
            else if(red_op_A)
                out_expected = &A;
            else if(red_op_B)
```

```
                out_expected = &B;
            else
                out_expected = A & B;
            repeat(2) @(negedge clk);
            if(out != out_expected) begin
                $display("Error - Opcode 0 AND");
                $exit;
            end
        end

        // Opcode = 1 functionality
        rst = 0;
        A = 0;
        B = 0;
        opcode = 1;
        repeat(2) @(negedge clk);
        repeat(50) begin
            A = $random;
            B = $random;
            red_op_A = $random;
            red_op_B = $random;
            if(red_op_A && red_op_B)
                out_expected = ^A;
            else if(red_op_A)
                out_expected = ^A;
            else if(red_op_B)
                out_expected = ^B;
            else
                out_expected = A ^ B;
            repeat(2) @(negedge clk);
            if(out != out_expected) begin
                $display("Error - Opcode 1 XOR");
                $exit;
            end
        end

        // Opcode = 2 functionality
        rst = 0;
        A = 0;
        B = 0;
        opcode = 2;
        red_op_A = 0;
        red_op_B = 0;
        repeat(2) @(negedge clk)
        repeat(50) begin
            A = $random;
            B = $random;
            cin = $random;
            out_expected = A + B + cin;
            repeat(2) @(negedge clk);
            if(out != out_expected) begin
                $display("Error - Opcode 2 ADD");
                $exit;
            end
        end

        // Opcode = 3 functionality
        rst = 0;
        A = 0;
        B = 0;
        opcode = 3;
        red_op_A = 0;
        red_op_B = 0;
        repeat(2) @(negedge clk)
        repeat(50) begin
            A = $random;
            B = $random;
            out_expected = A * B;
            repeat(2) @(negedge clk);
            if(out != out_expected) begin
                $display("Error - Opcode 3 Multiply");
                $exit;
            end
        end
        $display("Simulation Successful");
        $display("Testing Invalid Case");

        rst = 0;
        A = 0;
        B = 0;
        opcode = 3;
        red_op_A = 1;
        red_op_B = 0;
        repeat(100) @(negedge clk);

        $exit;
    end // End of Initial Block

endmodule
```

Simulation:

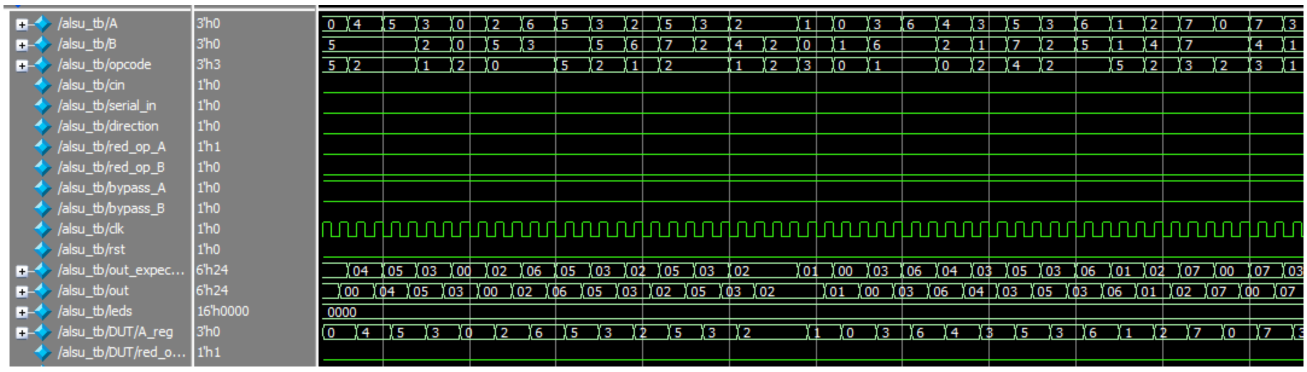


Figure 1 Testing Reset

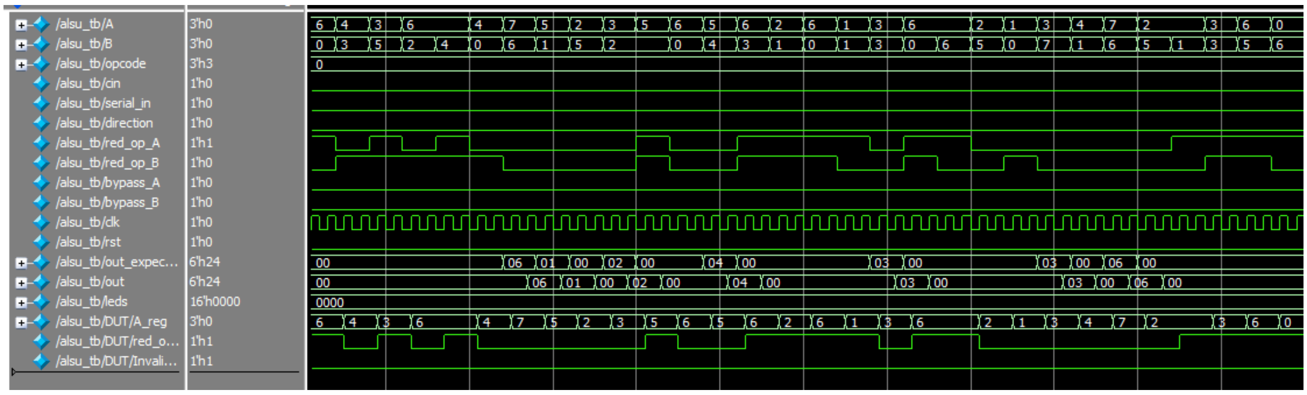


Figure 2 Opcode 0

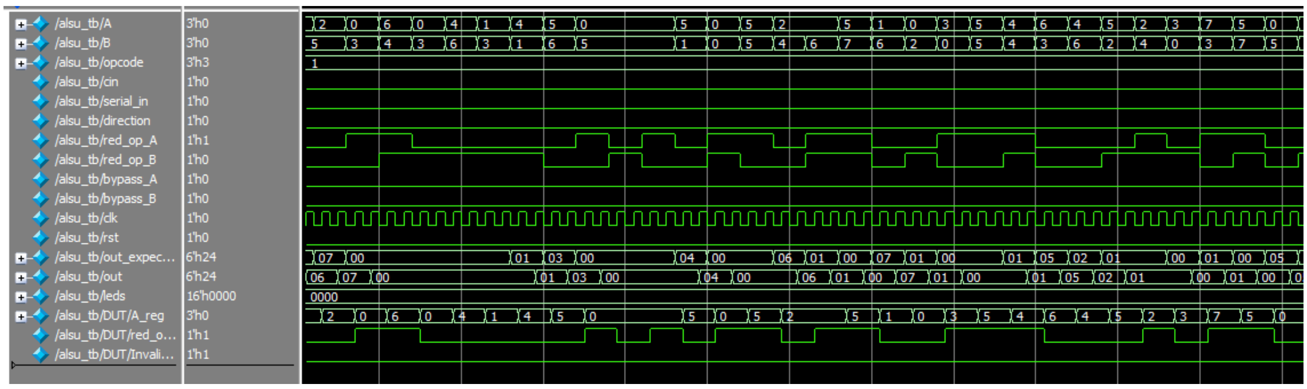


Figure 3 Opcode 1

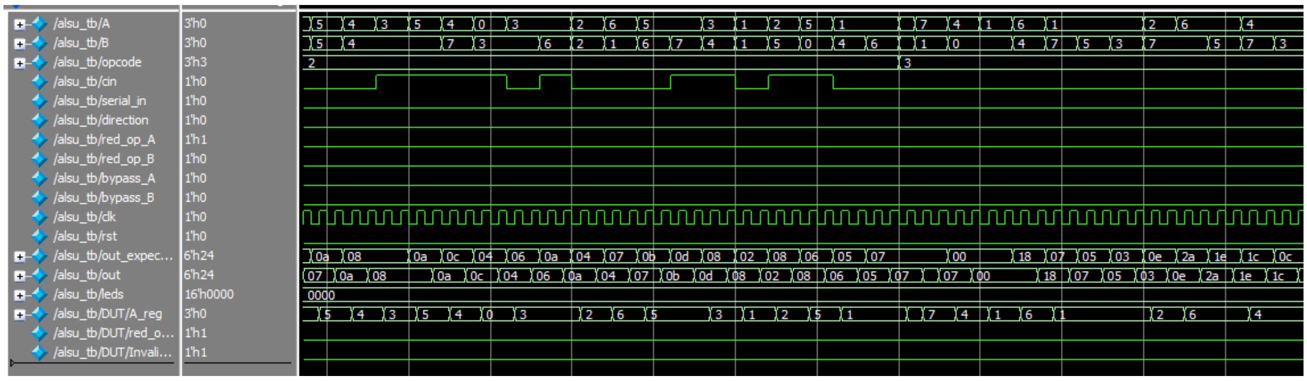


Figure 4 Opcode 2

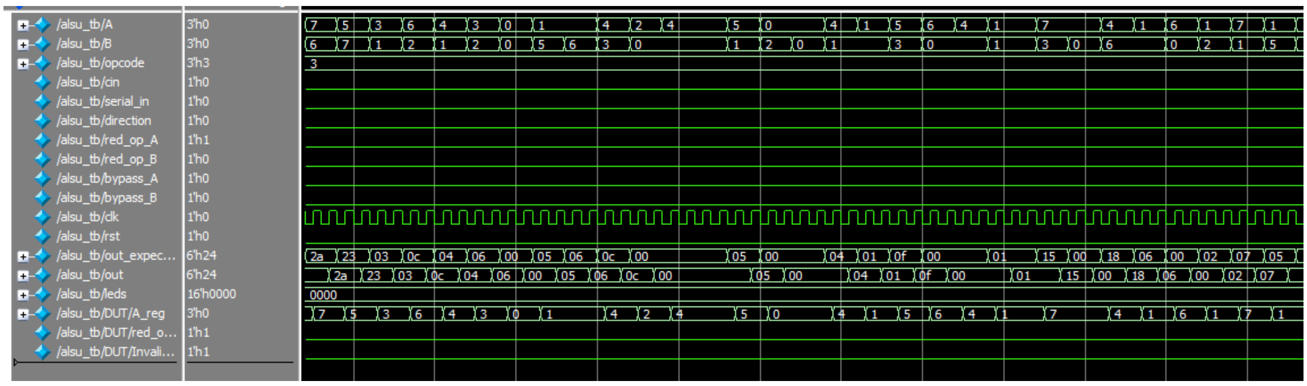


Figure 5 Opcode 3

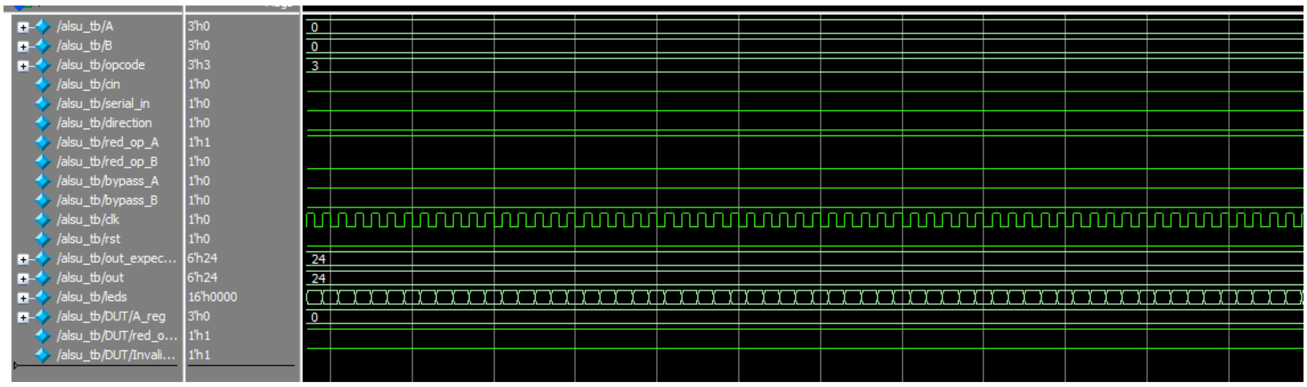
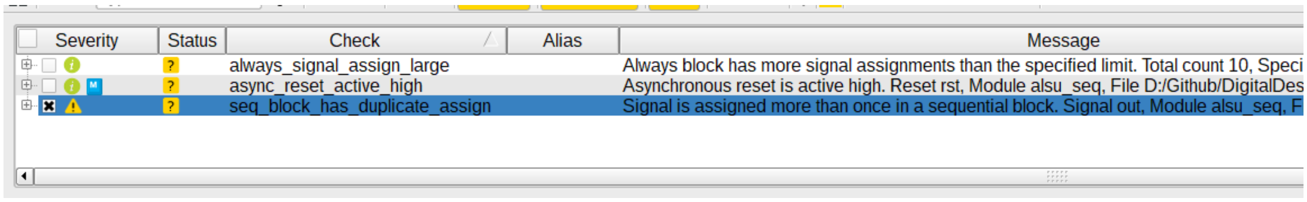


Figure 6 Invalid Case

Linting:



Severity	Status	Check	Alias	Message
	?	always_signal_assign_large		Always block has more signal assignments than the specified limit. Total count 10, Speci
	?	async_reset_active_high		Asynchronous reset is active high. Reset rst, Module alsu_seq, File D:/Github/DigitalDes
⚠	?	seq_block_has_duplicate_assign		Signal is assigned more than once in a sequential block. Signal out, Module alsu_seq, F

Figure 7 Linting Messages

Do File:

```
vlib work

vlog ALSU.v ALSU_tb.v

vsim -voptargs=+acc work.alsu_tb

add wave *
add wave alsu_tb/DUT.A_reg
add wave alsu_tb/DUT.red_op_A_reg
add wave alsu_tb/DUT.Invalid_Case

run -all

quit -sim
```

Vivado:

Constraint File:

```
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top
level signal names in the project

## 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 {serial_in}]
set_property -dict {PACKAGE_PIN V16 IOSTANDARD LVCMOS33} [get_ports {direction}]
set_property -dict {PACKAGE_PIN W16 IOSTANDARD LVCMOS33} [get_ports {bypass_B}]
set_property -dict {PACKAGE_PIN W17 IOSTANDARD LVCMOS33} [get_ports {bypass_A}]
set_property -dict {PACKAGE_PIN W15 IOSTANDARD LVCMOS33} [get_ports {red_op_B}]
set_property -dict {PACKAGE_PIN V15 IOSTANDARD LVCMOS33} [get_ports {red_op_A}]
set_property -dict {PACKAGE_PIN W14 IOSTANDARD LVCMOS33} [get_ports {cin}]
set_property -dict {PACKAGE_PIN W13 IOSTANDARD LVCMOS33} [get_ports {B[2]}]
set_property -dict {PACKAGE_PIN V2 IOSTANDARD LVCMOS33} [get_ports {B[1]}]
set_property -dict {PACKAGE_PIN T3 IOSTANDARD LVCMOS33} [get_ports {B[0]}]
set_property -dict { PACKAGE_PIN T2    IOSTANDARD LVCMOS33 } [get_ports {A[2]}]
set_property -dict { PACKAGE_PIN R3    IOSTANDARD LVCMOS33 } [get_ports {A[1]}]
set_property -dict { PACKAGE_PIN W2    IOSTANDARD LVCMOS33 } [get_ports {A[0]}]
set_property -dict { PACKAGE_PIN U1    IOSTANDARD LVCMOS33 } [get_ports {opcode[2]}]
set_property -dict { PACKAGE_PIN T1    IOSTANDARD LVCMOS33 } [get_ports {opcode[1]}]
set_property -dict { PACKAGE_PIN R2    IOSTANDARD LVCMOS33 } [get_ports {opcode[0]}]


## 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]}]


##7 Segment Display
#set_property -dict { PACKAGE_PIN W7   IOSTANDARD LVCMOS33 } [get_ports {seg[0]}]
#set_property -dict { PACKAGE_PIN W6   IOSTANDARD LVCMOS33 } [get_ports {seg[1]}]
#set_property -dict { PACKAGE_PIN U8   IOSTANDARD LVCMOS33 } [get_ports {seg[2]}]
#set_property -dict { PACKAGE_PIN V8   IOSTANDARD LVCMOS33 } [get_ports {seg[3]}]
#set_property -dict { PACKAGE_PIN U5   IOSTANDARD LVCMOS33 } [get_ports {seg[4]}]
#set_property -dict { PACKAGE_PIN V5   IOSTANDARD LVCMOS33 } [get_ports {seg[5]}]
#set_property -dict { PACKAGE_PIN U7   IOSTANDARD LVCMOS33 } [get_ports {seg[6]}]

#set_property -dict { PACKAGE_PIN V7   IOSTANDARD LVCMOS33 } [get_ports dp]

#set_property -dict { PACKAGE_PIN U2   IOSTANDARD LVCMOS33 } [get_ports {an[0]}]
#set_property -dict { PACKAGE_PIN U4   IOSTANDARD LVCMOS33 } [get_ports {an[1]}]
#set_property -dict { PACKAGE_PIN V4   IOSTANDARD LVCMOS33 } [get_ports {an[2]}]
#set_property -dict { PACKAGE_PIN W4   IOSTANDARD LVCMOS33 } [get_ports {an[3]}]


##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]


##Pmod Header JA
#set_property -dict { PACKAGE_PIN J1   IOSTANDARD LVCMOS33 } [get_ports
{JA[0]}];#Sch name = JA1
#set_property -dict { PACKAGE_PIN L2   IOSTANDARD LVCMOS33 } [get_ports
{JA[1]}];#Sch name = JA2
#set_property -dict { PACKAGE_PIN J2   IOSTANDARD LVCMOS33 } [get_ports
{JA[2]}];#Sch name = JA3
#set_property -dict { PACKAGE_PIN G2   IOSTANDARD LVCMOS33 } [get_ports
{JA[3]}];#Sch name = JA4
#set_property -dict { PACKAGE_PIN H1   IOSTANDARD LVCMOS33 } [get_ports
{JA[4]}];#Sch name = JA7
#set_property -dict { PACKAGE_PIN K2   IOSTANDARD LVCMOS33 } [get_ports
{JA[5]}];#Sch name = JA8
#set_property -dict { PACKAGE_PIN H2   IOSTANDARD LVCMOS33 } [get_ports
{JA[6]}];#Sch name = JA9
#set_property -dict { PACKAGE_PIN G3   IOSTANDARD LVCMOS33 } [get_ports
{JA[7]}];#Sch name = JA10

##Pmod Header JB
#set_property -dict { PACKAGE_PIN A14   IOSTANDARD LVCMOS33 } [get_ports
{JB[0]}];#Sch name = JB1
#set_property -dict { PACKAGE_PIN A16   IOSTANDARD LVCMOS33 } [get_ports
{JB[1]}];#Sch name = JB2
#set_property -dict { PACKAGE_PIN B15   IOSTANDARD LVCMOS33 } [get_ports
{JB[2]}];#Sch name = JB3
#set_property -dict { PACKAGE_PIN B16   IOSTANDARD LVCMOS33 } [get_ports
{JB[3]}];#Sch name = JB4
#set_property -dict { PACKAGE_PIN A15   IOSTANDARD LVCMOS33 } [get_ports
{JB[4]}];#Sch name = JB7
#set_property -dict { PACKAGE_PIN A17   IOSTANDARD LVCMOS33 } [get_ports
{JB[5]}];#Sch name = JB8
#set_property -dict { PACKAGE_PIN C15   IOSTANDARD LVCMOS33 } [get_ports
{JB[6]}];#Sch name = JB9
#set_property -dict { PACKAGE_PIN C16   IOSTANDARD LVCMOS33 } [get_ports
{JB[7]}];#Sch name = JB10

##Pmod Header JC
#set_property -dict { PACKAGE_PIN K17   IOSTANDARD LVCMOS33 } [get_ports
{JC[0]}];#Sch name = JC1
#set_property -dict { PACKAGE_PIN M18   IOSTANDARD LVCMOS33 } [get_ports
{JC[1]}];#Sch name = JC2
#set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports
{JC[2]}];#Sch name = JC3
#set_property -dict { PACKAGE_PIN P18   IOSTANDARD LVCMOS33 } [get_ports
{JC[3]}];#Sch name = JC4
```

```
#set_property -dict { PACKAGE_PIN L17   IOSTANDARD LVCMOS33 } [get_ports
{JC[4]}];#Sch name = JC7
#set_property -dict { PACKAGE_PIN M19   IOSTANDARD LVCMOS33 } [get_ports
{JC[5]}];#Sch name = JC8
#set_property -dict { PACKAGE_PIN P17   IOSTANDARD LVCMOS33 } [get_ports
{JC[6]}];#Sch name = JC9
#set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports
{JC[7]}];#Sch name = JC10

##Pmod Header JXADC
#set_property -dict { PACKAGE_PIN J3   IOSTANDARD LVCMOS33 } [get_ports
{JXADC[0]}];#Sch name = XA1_P
#set_property -dict { PACKAGE_PIN L3   IOSTANDARD LVCMOS33 } [get_ports
{JXADC[1]}];#Sch name = XA2_P
#set_property -dict { PACKAGE_PIN M2   IOSTANDARD LVCMOS33 } [get_ports
{JXADC[2]}];#Sch name = XA3_P
#set_property -dict { PACKAGE_PIN N2   IOSTANDARD LVCMOS33 } [get_ports
{JXADC[3]}];#Sch name = XA4_P
#set_property -dict { PACKAGE_PIN K3   IOSTANDARD LVCMOS33 } [get_ports
{JXADC[4]}];#Sch name = XA1_N
#set_property -dict { PACKAGE_PIN M3   IOSTANDARD LVCMOS33 } [get_ports
{JXADC[5]}];#Sch name = XA2_N
#set_property -dict { PACKAGE_PIN M1   IOSTANDARD LVCMOS33 } [get_ports
{JXADC[6]}];#Sch name = XA3_N
#set_property -dict { PACKAGE_PIN N1   IOSTANDARD LVCMOS33 } [get_ports
{JXADC[7]}];#Sch name = XA4_N


##VGA Connector
#set_property -dict { PACKAGE_PIN G19   IOSTANDARD LVCMOS33 } [get_ports
{vgaRed[0]}]
#set_property -dict { PACKAGE_PIN H19   IOSTANDARD LVCMOS33 } [get_ports
{vgaRed[1]}]
#set_property -dict { PACKAGE_PIN J19   IOSTANDARD LVCMOS33 } [get_ports
{vgaRed[2]}]
#set_property -dict { PACKAGE_PIN N19   IOSTANDARD LVCMOS33 } [get_ports
{vgaRed[3]}]
#set_property -dict { PACKAGE_PIN N18   IOSTANDARD LVCMOS33 } [get_ports
{vgaBlue[0]}]
#set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports
{vgaBlue[1]}]
#set_property -dict { PACKAGE_PIN K18   IOSTANDARD LVCMOS33 } [get_ports
{vgaBlue[2]}]
#set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports
{vgaBlue[3]}]
#set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports
{vgaGreen[0]}]
#set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports
{vgaGreen[1]}]
```

```
#set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports
{vgaGreen[2]}]
#set_property -dict { PACKAGE_PIN D17   IOSTANDARD LVCMOS33 } [get_ports
{vgaGreen[3]}]
#set_property -dict { PACKAGE_PIN P19   IOSTANDARD LVCMOS33 } [get_ports Hsync]
#set_property -dict { PACKAGE_PIN R19   IOSTANDARD LVCMOS33 } [get_ports Vsync]


##USB-RS232 Interface
#set_property -dict { PACKAGE_PIN B18   IOSTANDARD LVCMOS33 } [get_ports RsRx]
#set_property -dict { PACKAGE_PIN A18   IOSTANDARD LVCMOS33 } [get_ports RsTx]


##USB HID (PS/2)
#set_property -dict { PACKAGE_PIN C17   IOSTANDARD LVCMOS33   PULLUP true }
[get_ports PS2Clk]
#set_property -dict { PACKAGE_PIN B17   IOSTANDARD LVCMOS33   PULLUP true }
[get_ports PS2Data]


##Quad SPI Flash
##Note that CCLK_0 cannot be placed in 7 series devices. You can access it using the
##STARTUPE2 primitive.
#set_property -dict { PACKAGE_PIN D18   IOSTANDARD LVCMOS33 } [get_ports
{QspiDB[0]}]
#set_property -dict { PACKAGE_PIN D19   IOSTANDARD LVCMOS33 } [get_ports
{QspiDB[1]}]
#set_property -dict { PACKAGE_PIN G18   IOSTANDARD LVCMOS33 } [get_ports
{QspiDB[2]}]
#set_property -dict { PACKAGE_PIN F18   IOSTANDARD LVCMOS33 } [get_ports
{QspiDB[3]}]
#set_property -dict { PACKAGE_PIN K19   IOSTANDARD LVCMOS33 } [get_ports QspiCSn]


## 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]

set_property MARK_DEBUG true [get_nets {A_IBUF[2]}]
set_property MARK_DEBUG true [get_nets {leds_OBUF[3]}]
set_property MARK_DEBUG true [get_nets {leds_OBUF[6]}]
set_property MARK_DEBUG true [get_nets {leds_OBUF[12]}]
set_property MARK_DEBUG true [get_nets {opcode_IBUF[0]}]
set_property MARK_DEBUG true [get_nets {opcode_IBUF[2]}]
set_property MARK_DEBUG true [get_nets bypass_A_IBUF]
set_property MARK_DEBUG true [get_nets {leds_OBUF[4]}]
set_property MARK_DEBUG true [get_nets {A_IBUF[0]}]
set_property MARK_DEBUG true [get_nets {A_IBUF[1]}]
set_property MARK_DEBUG true [get_nets red_op_B_IBUF]
set_property MARK_DEBUG true [get_nets red_op_A_IBUF]
set_property MARK_DEBUG true [get_nets bypass_B_IBUF]
set_property MARK_DEBUG true [get_nets direction_IBUF]
set_property MARK_DEBUG true [get_nets {B_IBUF[1]}]
set_property MARK_DEBUG true [get_nets {leds_OBUF[1]}]
set_property MARK_DEBUG true [get_nets {out_OBUF[0]}]
set_property MARK_DEBUG true [get_nets {out_OBUF[3]}]
set_property MARK_DEBUG true [get_nets {leds_OBUF[7]}]
set_property MARK_DEBUG true [get_nets {leds_OBUF[9]}]
set_property MARK_DEBUG true [get_nets {leds_OBUF[11]}]
set_property MARK_DEBUG true [get_nets {leds_OBUF[14]}]
set_property MARK_DEBUG true [get_nets {leds_OBUF[13]}]
set_property MARK_DEBUG true [get_nets {out_OBUF[1]}]
set_property MARK_DEBUG true [get_nets {B_IBUF[0]}]
set_property MARK_DEBUG true [get_nets {B_IBUF[2]}]
set_property MARK_DEBUG true [get_nets cin_IBUF]
set_property MARK_DEBUG true [get_nets {leds_OBUF[0]}]
set_property MARK_DEBUG true [get_nets {leds_OBUF[2]}]
set_property MARK_DEBUG true [get_nets {leds_OBUF[5]}]
set_property MARK_DEBUG true [get_nets {leds_OBUF[15]}]
set_property MARK_DEBUG true [get_nets {out_OBUF[2]}]
set_property MARK_DEBUG true [get_nets {out_OBUF[4]}]
set_property MARK_DEBUG true [get_nets {leds_OBUF[8]}]
set_property MARK_DEBUG true [get_nets {leds_OBUF[10]}]
set_property MARK_DEBUG true [get_nets {opcode_IBUF[1]}]
set_property MARK_DEBUG true [get_nets {out_OBUF[5]}]
set_property MARK_DEBUG true [get_nets clk_IBUF]
set_property MARK_DEBUG true [get_nets rst_IBUF]
set_property MARK_DEBUG true [get_nets serial_in_IBUF]
create_debug_core u_ila_0 ila
set_property ALL_PROBE_SAME_MU true [get_debug_cores u_ila_0]
set_property ALL_PROBE_SAME_MU_CNT 1 [get_debug_cores u_ila_0]
set_property C_ADV_TRIGGER false [get_debug_cores u_ila_0]
set_property C_DATA_DEPTH 1024 [get_debug_cores u_ila_0]
set_property C_EN_STRG_QUAL false [get_debug_cores u_ila_0]
set_property C_INPUT_PIPE_STAGES 0 [get_debug_cores u_ila_0]
set_property C_TRIGIN_EN false [get_debug_cores u_ila_0]
set_property C_TRIGOUT_EN false [get_debug_cores u_ila_0]
set_property port_width 1 [get_debug_ports u_ila_0/clk]
connect_debug_port u_ila_0/clk [get_nets [list clk_IBUF_BUFG]]
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe0]
set_property port_width 3 [get_debug_ports u_ila_0/probe0]
```

```
connect_debug_port u_ila_0/probe0 [get_nets [list {A_IBUF[0]} {A_IBUF[1]}
{A_IBUF[2]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe1]
set_property port_width 16 [get_debug_ports u_ila_0/probe1]
connect_debug_port u_ila_0/probe1 [get_nets [list {leds_OBUF[0]} {leds_OBUF[1]}
{leds_OBUF[2]} {leds_OBUF[3]} {leds_OBUF[4]} {leds_OBUF[5]} {leds_OBUF[6]}
{leds_OBUF[7]} {leds_OBUF[8]} {leds_OBUF[9]} {leds_OBUF[10]} {leds_OBUF[11]}
{leds_OBUF[12]} {leds_OBUF[13]} {leds_OBUF[14]} {leds_OBUF[15]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe2]
set_property port_width 3 [get_debug_ports u_ila_0/probe2]
connect_debug_port u_ila_0/probe2 [get_nets [list {opcode_IBUF[0]} {opcode_IBUF[1]}
{opcode_IBUF[2]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe3]
set_property port_width 6 [get_debug_ports u_ila_0/probe3]
connect_debug_port u_ila_0/probe3 [get_nets [list {out_OBUF[0]} {out_OBUF[1]}
{out_OBUF[2]} {out_OBUF[3]} {out_OBUF[4]} {out_OBUF[5]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe4]
set_property port_width 3 [get_debug_ports u_ila_0/probe4]
connect_debug_port u_ila_0/probe4 [get_nets [list {B_IBUF[0]} {B_IBUF[1]}
{B_IBUF[2]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe5]
set_property port_width 1 [get_debug_ports u_ila_0/probe5]
connect_debug_port u_ila_0/probe5 [get_nets [list bypass_A_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe6]
set_property port_width 1 [get_debug_ports u_ila_0/probe6]
connect_debug_port u_ila_0/probe6 [get_nets [list bypass_B_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe7]
set_property port_width 1 [get_debug_ports u_ila_0/probe7]
connect_debug_port u_ila_0/probe7 [get_nets [list cin_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe8]
set_property port_width 1 [get_debug_ports u_ila_0/probe8]
connect_debug_port u_ila_0/probe8 [get_nets [list clk_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe9]
set_property port_width 1 [get_debug_ports u_ila_0/probe9]
connect_debug_port u_ila_0/probe9 [get_nets [list direction_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe10]
set_property port_width 1 [get_debug_ports u_ila_0/probe10]
connect_debug_port u_ila_0/probe10 [get_nets [list red_op_A_IBUF]]
```

```
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe11]
set_property port_width 1 [get_debug_ports u_ila_0/probe11]
connect_debug_port u_ila_0/probe11 [get_nets [list red_op_B_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe12]
set_property port_width 1 [get_debug_ports u_ila_0/probe12]
connect_debug_port u_ila_0/probe12 [get_nets [list rst_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe13]
set_property port_width 1 [get_debug_ports u_ila_0/probe13]
connect_debug_port u_ila_0/probe13 [get_nets [list serial_in_IBUF]]
set_property C_CLK_INPUT_FREQ_HZ 300000000 [get_debug_cores dbg_hub]
set_property C_ENABLE_CLK_DIVIDER false [get_debug_cores dbg_hub]
set_property C_USER_SCAN_CHAIN 1 [get_debug_cores dbg_hub]
connect_debug_port dbg_hub/clk [get_nets clk_IBUF_BUFG]
```

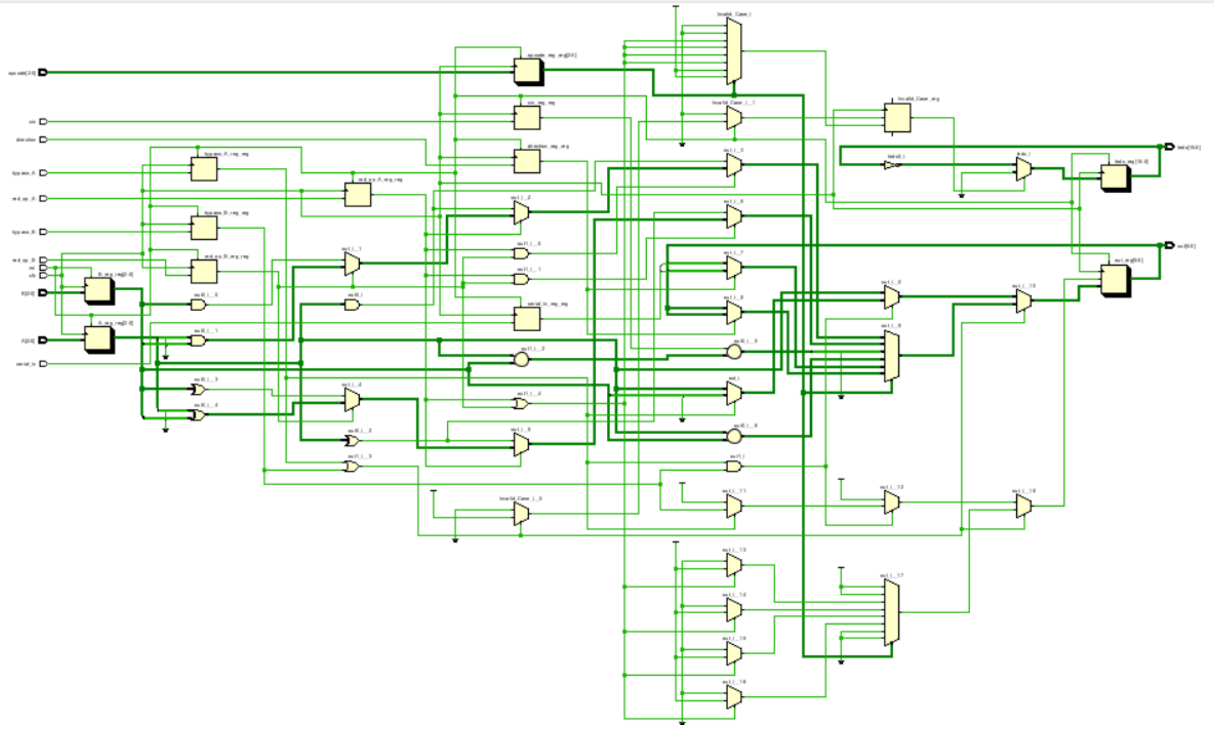
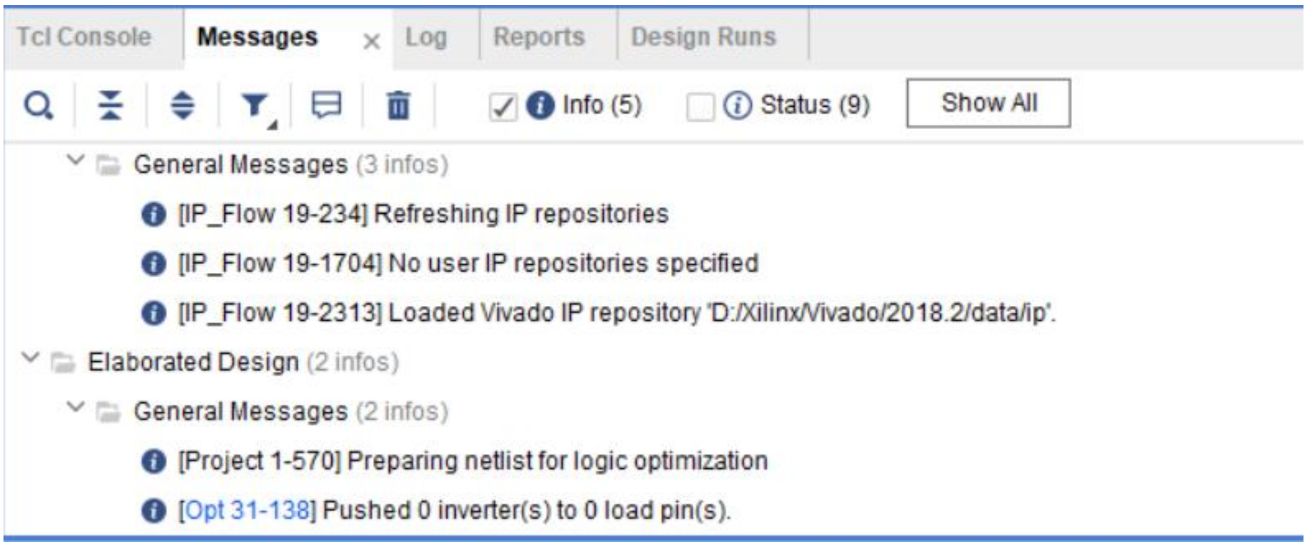


Figure 8 Elaboration Schematic



General Messages (3 infos)

[IP_Flow 19-234] Refreshing IP repositories

[IP_Flow 19-1704] No user IP repositories specified

[IP_Flow 19-2313] Loaded Vivado IP repository 'D:/Xilinx/Vivado/2018.2/data/ip'.

Elaborated Design (2 infos)

General Messages (2 infos)

[Project 1-570] Preparing netlist for logic optimization

[Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).

Figure 9 No Critical Warnings

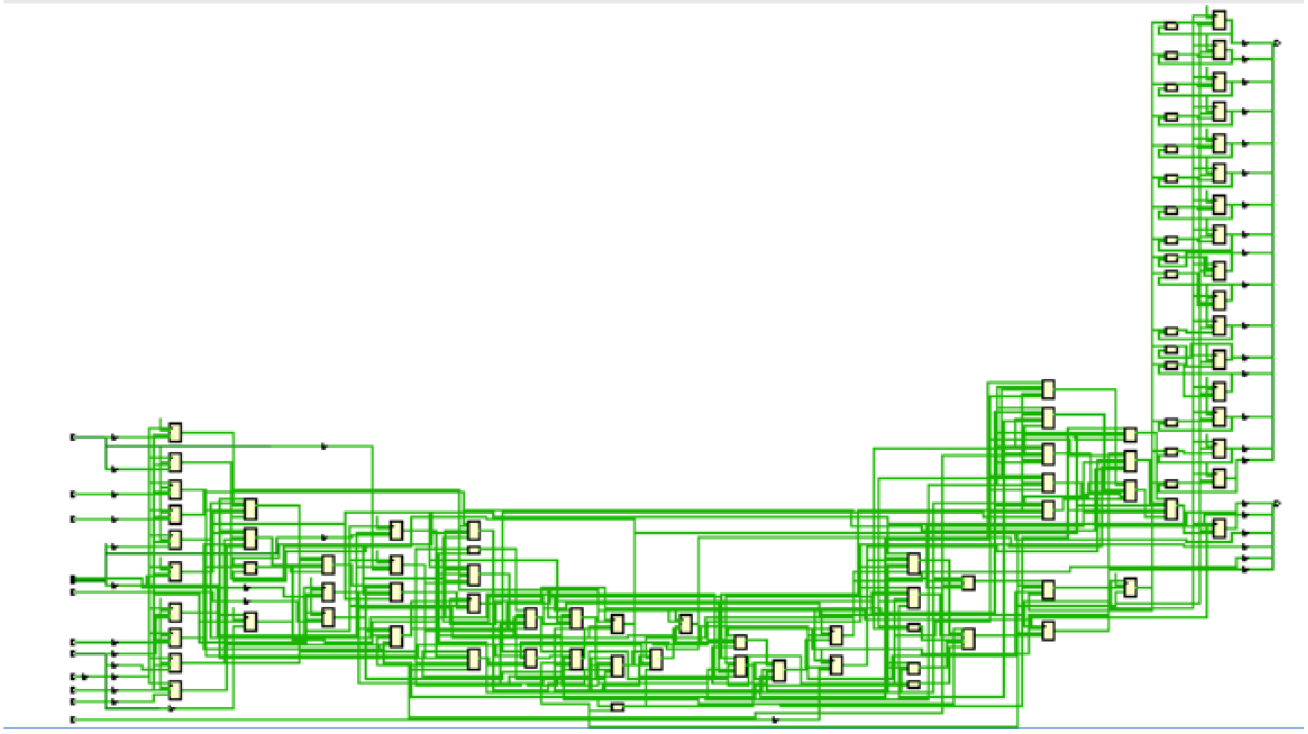
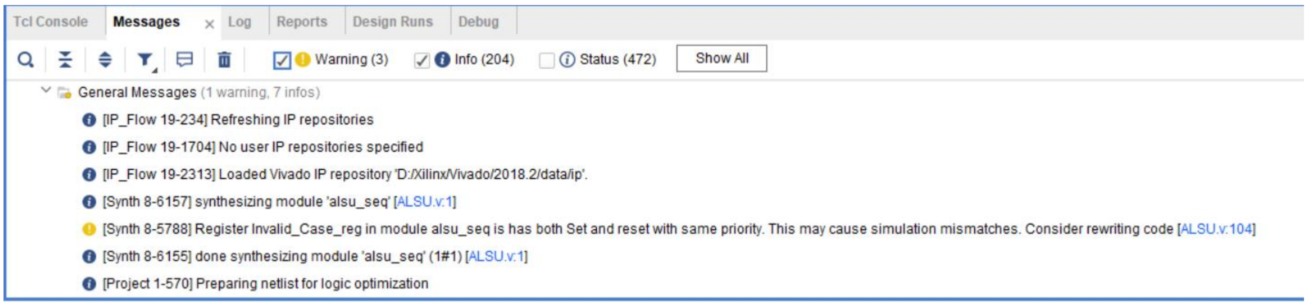
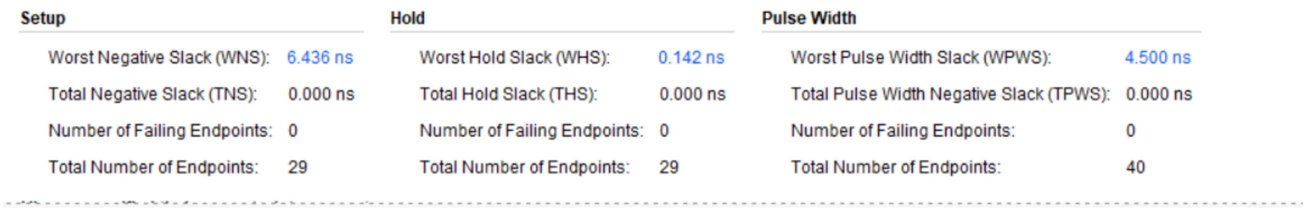


Figure 10 Syntesis Schematic



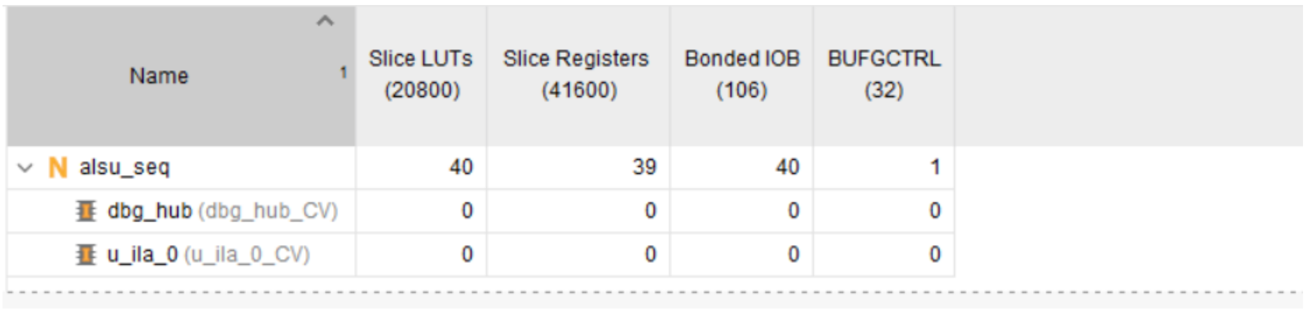
```
General Messages (1 warning, 7 infos)
    [IP_Flow 19-234] Refreshing IP repositories
    [IP_Flow 19-1704] No user IP repositories specified
    [IP_Flow 19-2313] Loaded Vivado IP repository 'D:/Xilinx/Vivado/2018.2/data/ip'.
    [Synth 8-6157] synthesizing module 'alsu_seq' [ALSU.v:1]
    [Synth 8-5788] Register Invalid_Case_reg in module alsu_seq is has both Set and reset with same priority. This may cause simulation mismatches. Consider rewriting code [ALSU.v:104]
    [Synth 8-6155] done synthesizing module 'alsu_seq' (1#1) [ALSU.v:1]
    [Project 1-570] Preparing netlist for logic optimization
```

Figure 11 No Critical Warnings or Errors



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.436 ns	Worst Hold Slack (WHS):	0.142 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:	29	Total Number of Endpoints:	29	Total Number of Endpoints:	40

Figure 12 Syntehsis Timing Summary



Name	Slice LUTs (20800)	Slice Registers (41600)	Bonded IOB (106)	BUFGCTRL (32)
alsu_seq	40	39	40	1
dbg_hub (dbg_hub_CV)	0	0	0	0
u_ila_0 (u_ila_0_CV)	0	0	0	0

Figure 13 Syntehsis Utilization Summary

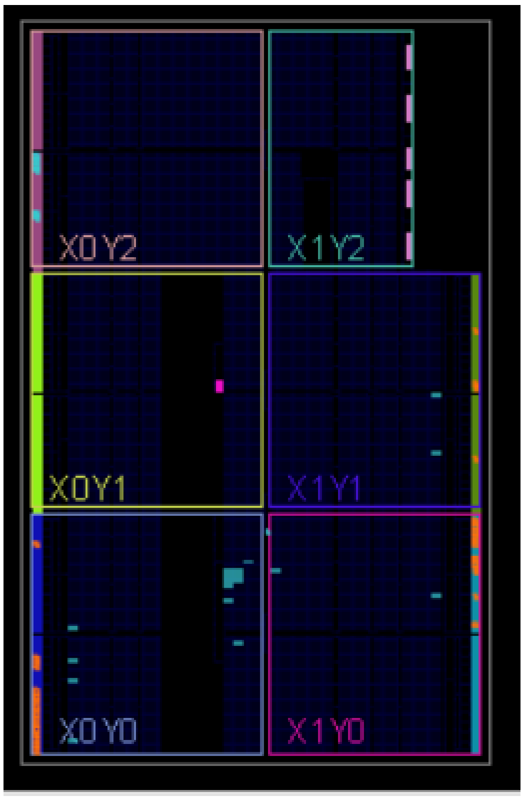
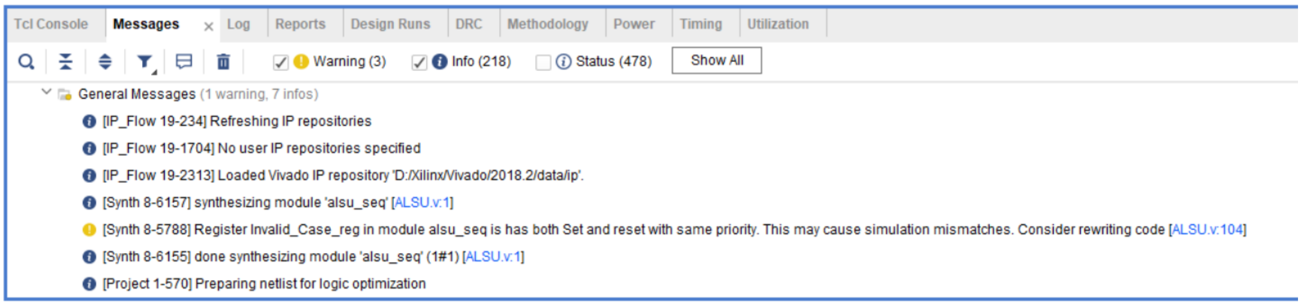
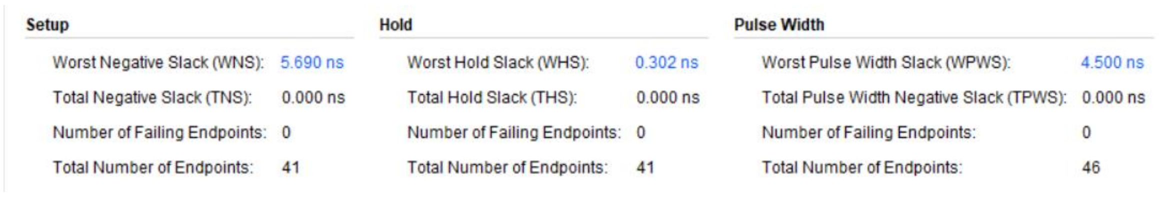


Figure 14 Device Snippet



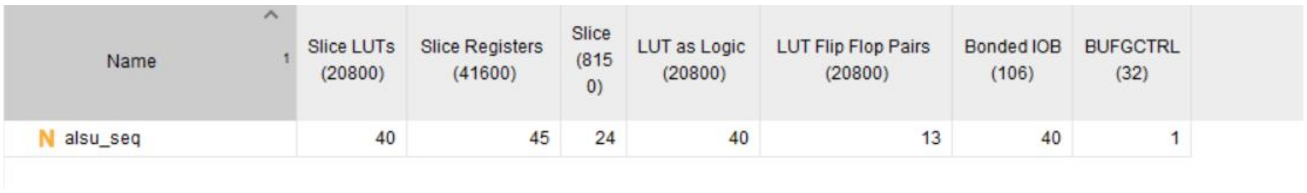
```
General Messages (1 warning, 7 infos)
    [IP_Flow 19-234] Refreshing IP repositories
    [IP_Flow 19-1704] No user IP repositories specified
    [IP_Flow 19-2313] Loaded Vivado IP repository 'D:/Xilinx/Vivado/2018.2/data/ip'.
    [Synth 8-6157] synthesizing module 'alsu_seq' [ALSU.v:1]
    [Synth 8-5788] Register Invalid_Case_reg in module alsu_seq is has both Set and reset with same priority. This may cause simulation mismatches. Consider rewriting code [ALSU.v:104]
    [Synth 8-6155] done synthesizing module 'alsu_seq' (1#1) [ALSU.v:1]
    [Project 1-570] Preparing netlist for logic optimization
```

Figure 15 Messages



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.690 ns	Worst Hold Slack (WHS):	0.302 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:	41	Total Number of Endpoints:	41	Total Number of Endpoints:	46

Figure 16 Timing Summary



Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Bonded IOB (106)	BUFGCTRL (32)
N alsu_seq	40	45	24	40	13	40	1

Figure 17 Utilization Summary

Q2: Simple DSP:

RTL Code:

```
module simple_dsp(
                    A,
                    B,
                    C,
                    D,
                    clk,
                    rst_n,
                    P);

    parameter OPERATION = "ADD";

    input [17 : 0]  A,
                    B,
                    D;
    input [47 : 0]  C;
    input   clk,
            rst_n;
    output [47 : 0] P;

    reg [17 : 0]    A_reg,
                    A1_reg,
                    B_reg,
                    D_reg,
                    pre_adder_out;
    reg [35 : 0]    multiplier_out;
    reg [47 : 0]    C_reg,
                    P_reg;

    // First Stage
    always @(posedge clk or negedge rst_n) 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
```

```verilog
    // Second Stage
    generate
        case(OPERATION)
            "ADD": begin
                always @(posedge clk or negedge rst_n) begin
                    if(~rst_n) begin
                        pre_adder_out <= 0;
                        A1_reg <= 0;
                    end
                    else begin
                        A1_reg <= A_reg;
                        pre_adder_out <= D_reg + B_reg;
                    end
                end
            end
            "SUBTRACT": begin
                always @(posedge clk or negedge rst_n) begin
                    if(~rst_n) begin
                        pre_adder_out <= 0;
                        A1_reg <= 0;
                    end
                    else begin
                        A1_reg <= A_reg;
                        pre_adder_out <= D_reg - B_reg;
                    end
                end
            end
        endcase
    endgenerate

    // Third Stage
    always @(posedge clk or negedge rst_n) begin
        if(~rst_n) begin
            multiplier_out <= 0;
        end
        else begin
            multiplier_out <= A1_reg * pre_adder_out;
        end
    end

    // Last Stage
    generate
        case(OPERATION)
            "ADD": begin
                always @(posedge clk or negedge rst_n) begin
                    if(~rst_n) begin
                        P_reg <= 0;
                    end
                    else begin
                        P_reg <= multiplier_out + C_reg;
                    end
                end
            end
            "SUBTRACT": begin
                always @(posedge clk or negedge rst_n) begin
                    if(~rst_n) begin
                        P_reg <= 0;
                    end
                    else begin
                        P_reg <= multiplier_out - C_reg;
                    end
                end
            end
        endcase
    endgenerate

    assign P = P_reg;



endmodule
```

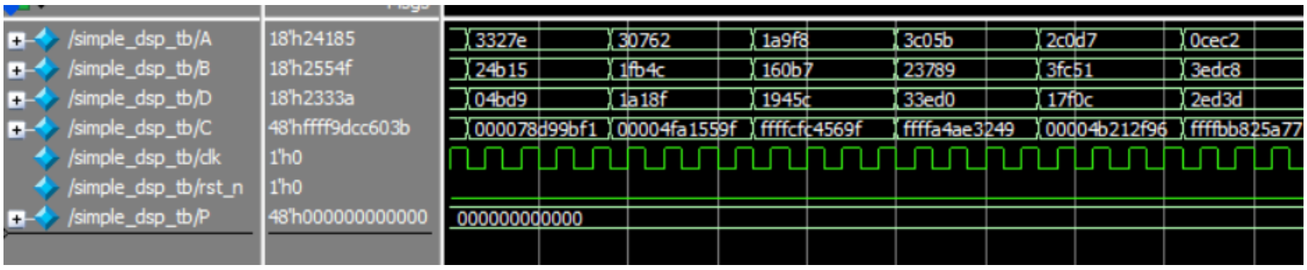
Testbench:

```
module simple_dsp_tb ();

    parameter OPERATION = "ADD";

    reg [17 : 0]    A,
                    B,
                    D;
    reg [47 : 0]  C;
    reg clk,
        rst_n;
    wire [47 : 0] P;

    simple_dsp  DUT(
                    A,
                    B,
                    C,
                    D,
                    clk,
                    rst_n,
                    P);

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

    initial begin
        rst_n = 0;
        repeat(50) begin
            A = $random;
            B = $random;
            C = $random;
            D = $random;
            repeat(4) @(negedge clk);
            if(P != 0) begin
                $display("ERROR - Reset");
                $stop;
            end
        end

        rst_n = 1;
        D = 20;
        B = 25;
        A = 53;
        C = 60;
        repeat(4) @(negedge clk);
        if(P != 2445) begin
            $display("Error - Path 1");
            $stop;
        end

        $display("Simulation Successfull");
        $stop;
    end

endmodule
```

Do File:

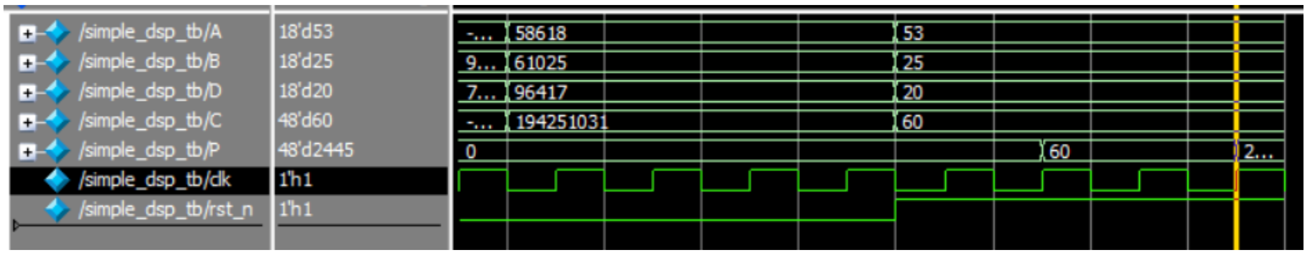
```
vlib work

vlog simple_dsp.v simple_dsp_tb.v

vsim -voptargs=+acc work.simple_dsp_tb

add wave *

run -all

quit -sim
```

Simulation:



```
/simple_dsp_tb/A      18'h24185         3327e          30762          1a9f8          3c05b          2c0d7          0cec2
/simple_dsp_tb/B      18'h2554f         24b15          1fb4c          160b7          23789          3fc51          3edc8
/simple_dsp_tb/D      18'h2333a         04bd9          1a18f          1945c          33ed0          17f0c          2ed3d
/simple_dsp_tb/C      48'hffff9dcc603b  000078d99bf1   00004fa1559f   ffffcfc4569f   ffffa4ae3249   00004b212f96   ffffbb825a77
/simple_dsp_tb/clk    1'h0
/simple_dsp_tb/rst_n  1'h0
/simple_dsp_tb/P      48'h000000000000  000000000000
```

Figure 18 Reset Test



```
/simple_dsp_tb/A      18'd53    -...  58618        53
/simple_dsp_tb/B      18'd25    9...  61025        25
/simple_dsp_tb/D      18'd20    7...  96417        20
/simple_dsp_tb/C      48'd60    -...  194251031    60
/simple_dsp_tb/P      48'd2445  0                        60    2...
/simple_dsp_tb/clk    1'h1
/simple_dsp_tb/rst_n  1'h1
```

Figure 19 Testing Path with fixed Value

Vivado:

Constraint File:

```
## 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]
```

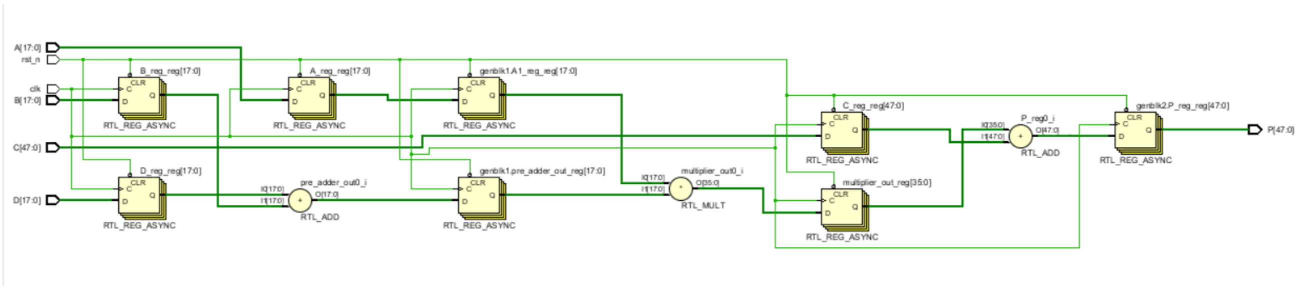
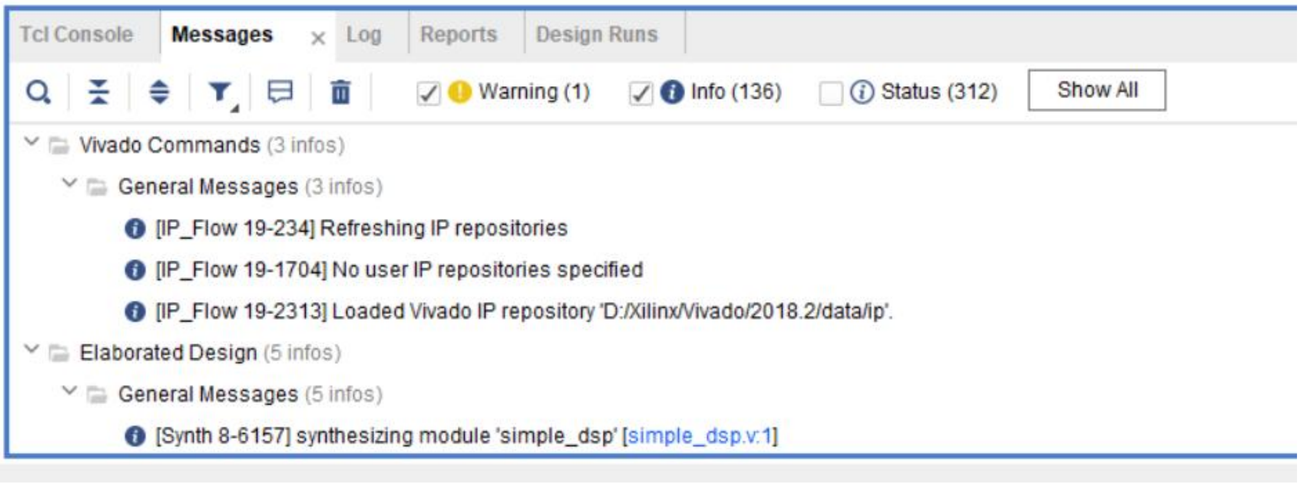


Figure 20 Elaboration



Vivado Commands (3 infos)
General Messages (3 infos)
[IP_Flow 19-234] Refreshing IP repositories
[IP_Flow 19-1704] No user IP repositories specified
[IP_Flow 19-2313] Loaded Vivado IP repository 'D:/Xilinx/Vivado/2018.2/data/ip'.
Elaborated Design (5 infos)
General Messages (5 infos)
[Synth 8-6157] synthesizing module 'simple_dsp' [simple_dsp.v:1]

Figure 21 Nor Critical Warnings or Errors

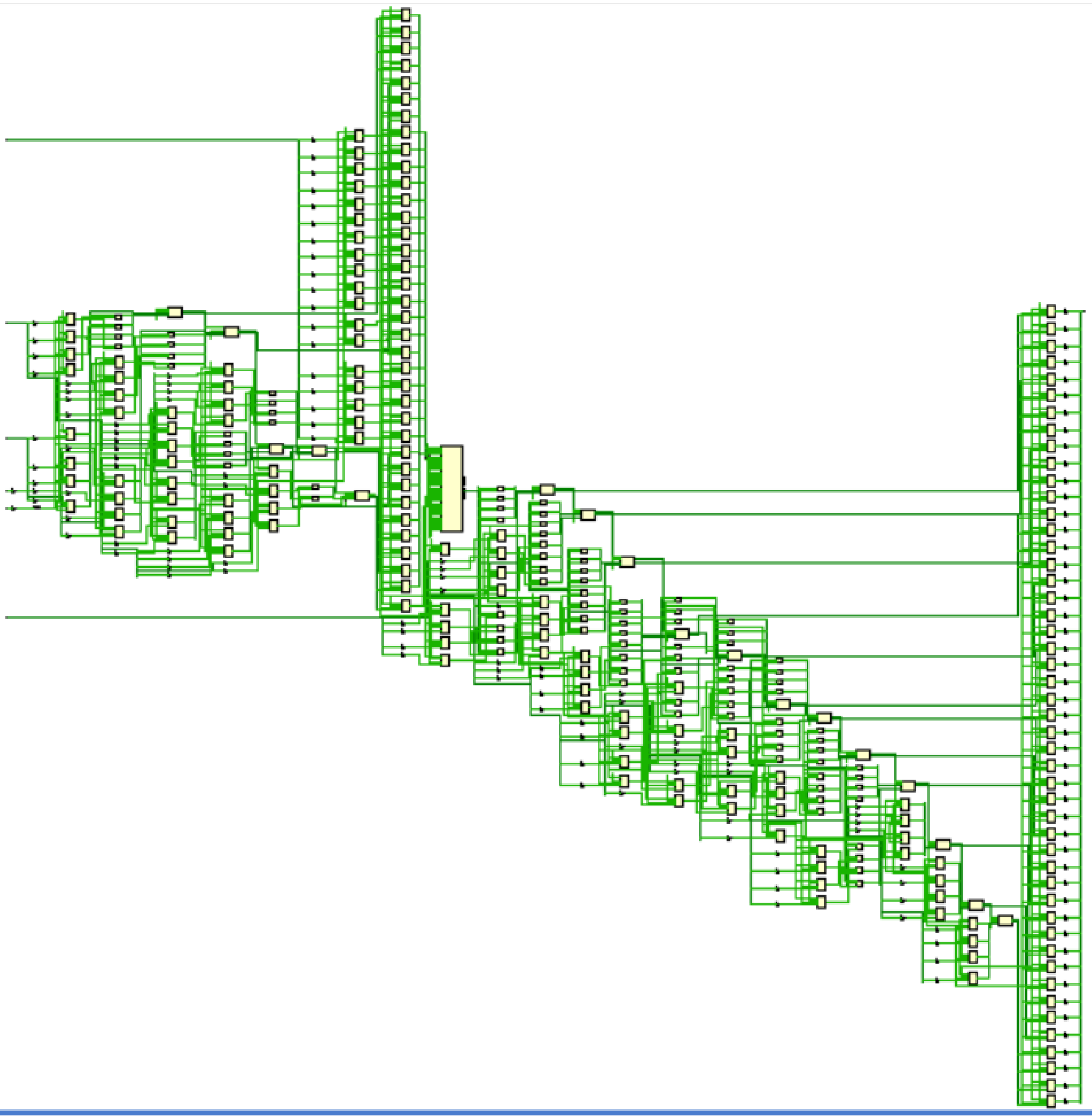
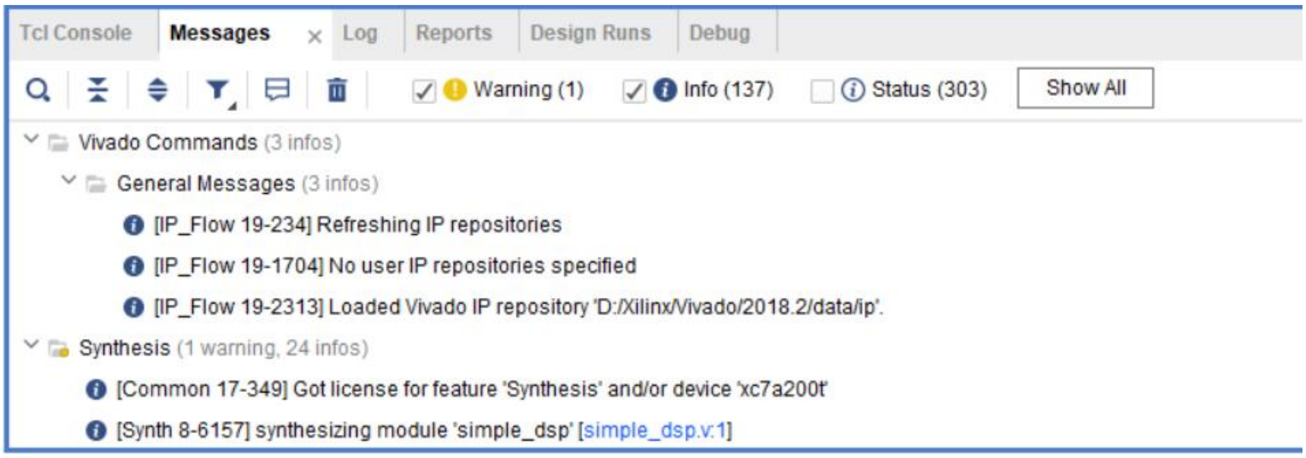
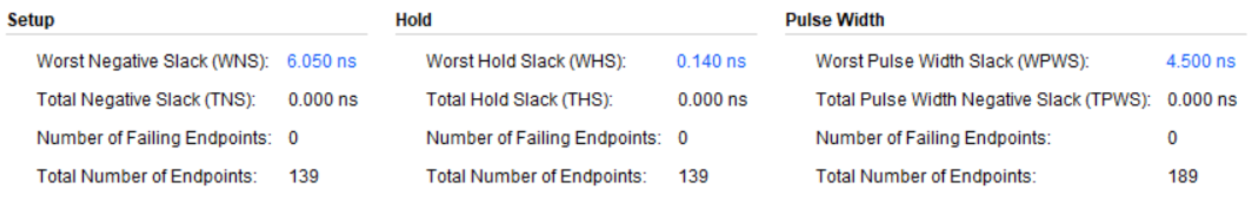


Figure 22 Synthesis



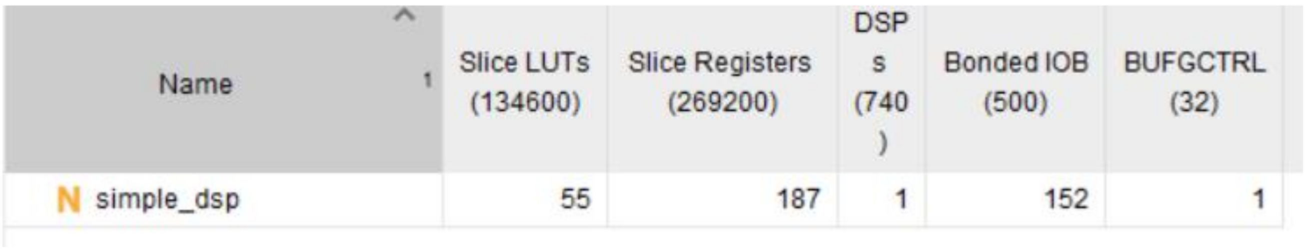
```
Vivado Commands (3 infos)
  General Messages (3 infos)
    [IP_Flow 19-234] Refreshing IP repositories
    [IP_Flow 19-1704] No user IP repositories specified
    [IP_Flow 19-2313] Loaded Vivado IP repository 'D:/Xilinx/Vivado/2018.2/data/ip'.
Synthesis (1 warning, 24 infos)
  [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a200t'
  [Synth 8-6157] synthesizing module 'simple_dsp' [simple_dsp.v:1]
```

Figure 23 No Critical Warnings or Errors



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.050 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:	139	Total Number of Endpoints:	139	Total Number of Endpoints:	189

Figure 24 Timing Summary



Name	Slice LUTs (134600)	Slice Registers (269200)	DSPs (740)	Bonded IOB (500)	BUFGCTRL (32)
simple_dsp	55	187	1	152	1

Figure 25 Utilization Summary

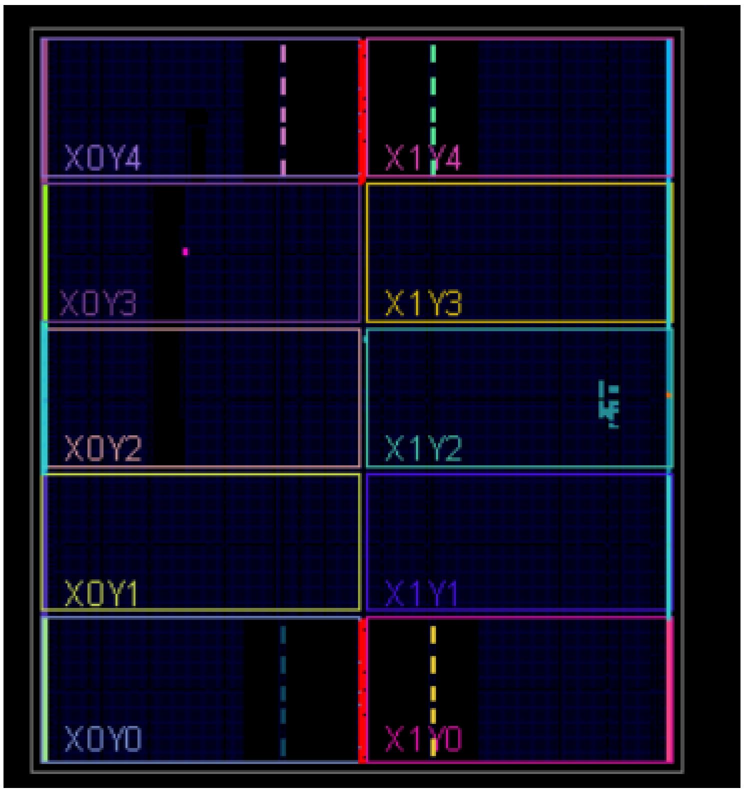
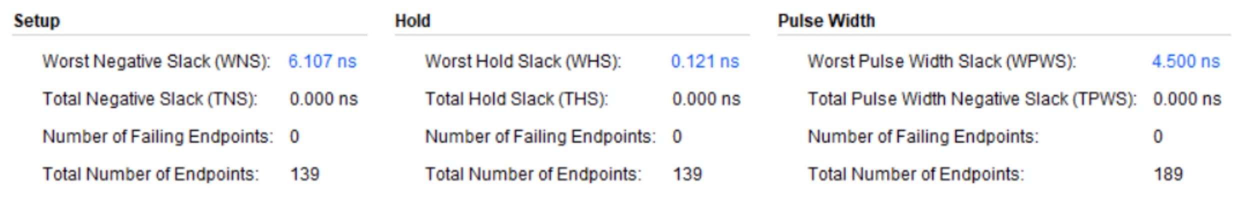
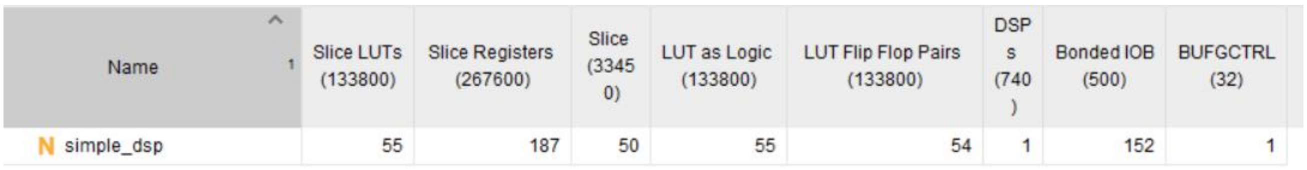


Figure 26 Devices Snippet



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.107 ns	Worst Hold Slack (WHS):	0.121 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:	139	Total Number of Endpoints:	139	Total Number of Endpoints:	189

Figure 27 Timing Summary



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)
simple_dsp	55	187	50	55	54	1	152	1

Figure 28 Utilization Summary

Q3: Time Division Mux

RTL Code:

```verilog
module time_mux(in0,
                in1,
                in2,
                in3,
                clk,
                rst,
                out);

    input [1 : 0]   in0,
                    in1,
                    in2,
                    in3;
    input   rst,
            clk;
    output reg [1 : 0]  out;

    reg [1 : 0] counter_reg;

    always @(posedge clk) begin
        if(rst)
            counter_reg <= 0;
        else begin
            if(counter_reg >= 3)
                counter_reg <= 0;
            else
                counter_reg <= counter_reg + 1;
        end
    end

    always @(*) begin
        case(counter_reg)
            0:  out = in0;
            1:  out = in1;
            2:  out = in2;
            3:  out = in3;
        endcase
    end

endmodule
```

Testbench:

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

    time_mux DUT(in0,
                in1,
                in2,
                in3,
                clk,
                rst,
                out);

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

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

        rst = 0;
        repeat(1000) @(negedge clk);

        $stop;
    end
endmodule
```

Do File:

```
vlib work

vlog time_mux.v time_mux_tb.v

vsim -voptargs=+acc work.time_mux_tb

add wave *

run -all

quit -sim
```

Simulation Snippets:

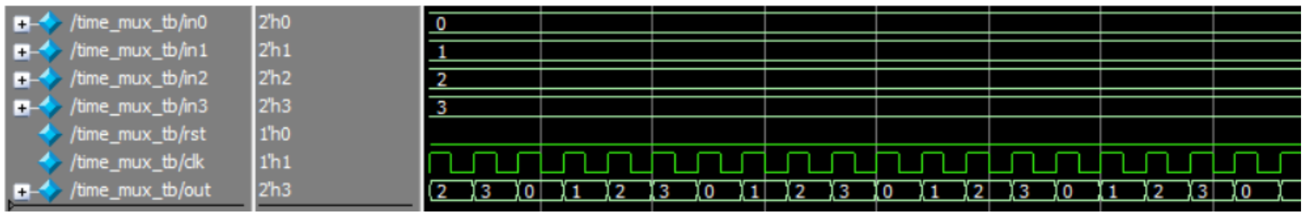


Figure 29 Output Snippet

Vivado:

Constraint File:

```
## 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]
```

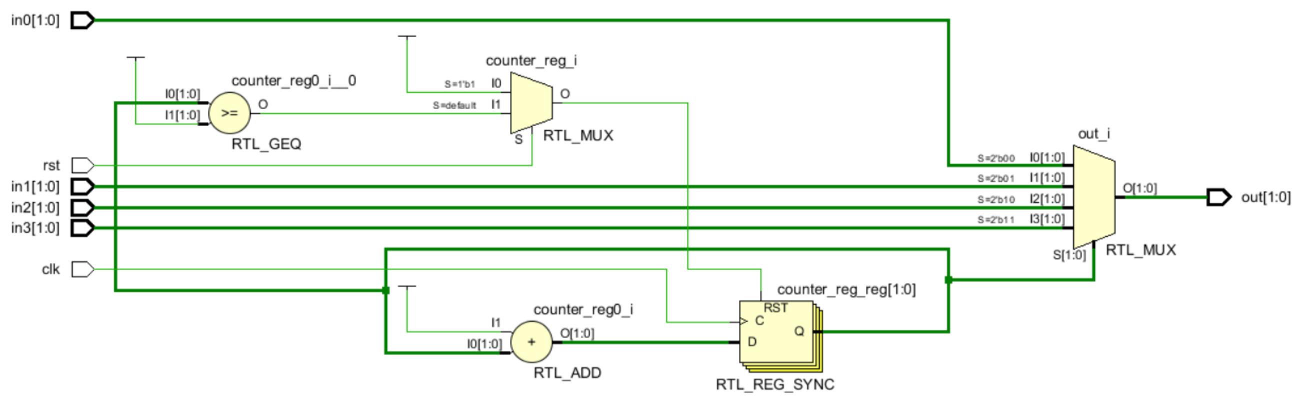
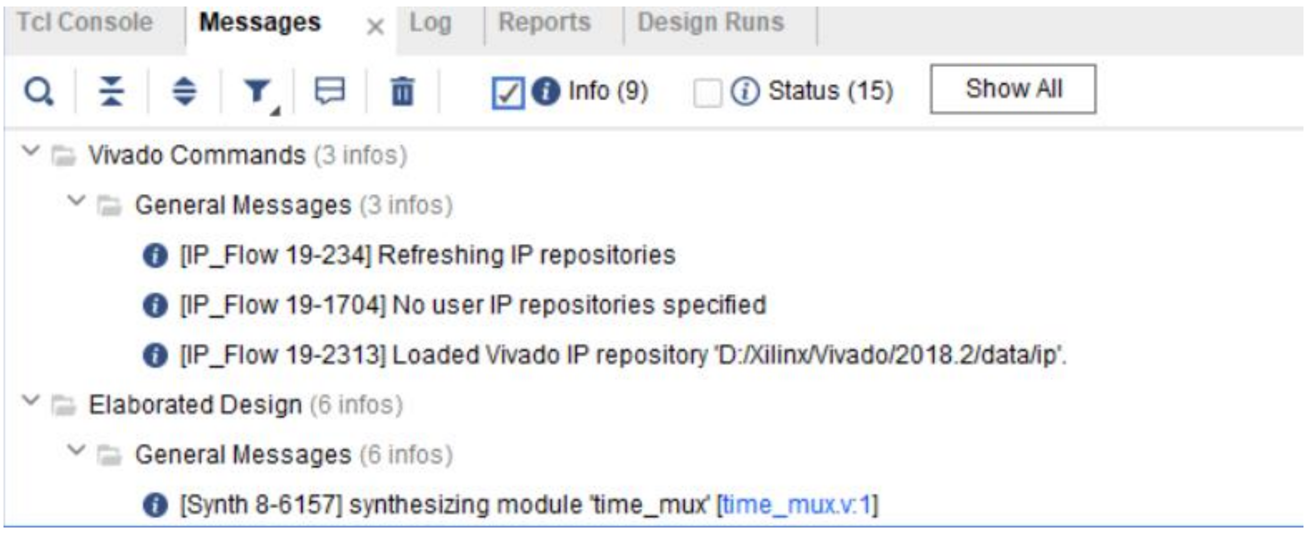


Figure 30 Elaboration Schematic



Vivado Commands (3 infos)
General Messages (3 infos)
[IP_Flow 19-234] Refreshing IP repositories
[IP_Flow 19-1704] No user IP repositories specified
[IP_Flow 19-2313] Loaded Vivado IP repository 'D:/Xilinx/Vivado/2018.2/data/ip'.
Elaborated Design (6 infos)
General Messages (6 infos)
[Synth 8-6157] synthesizing module 'time_mux' [time_mux.v:1]

Figure 31 No Critical Messages or Errors

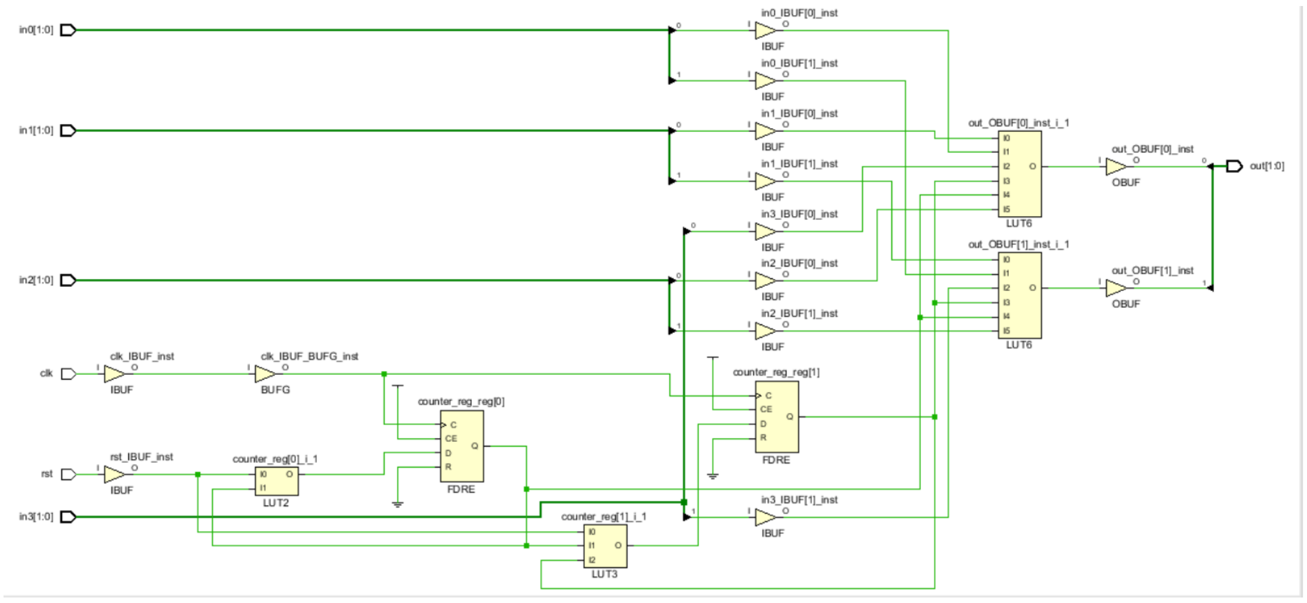
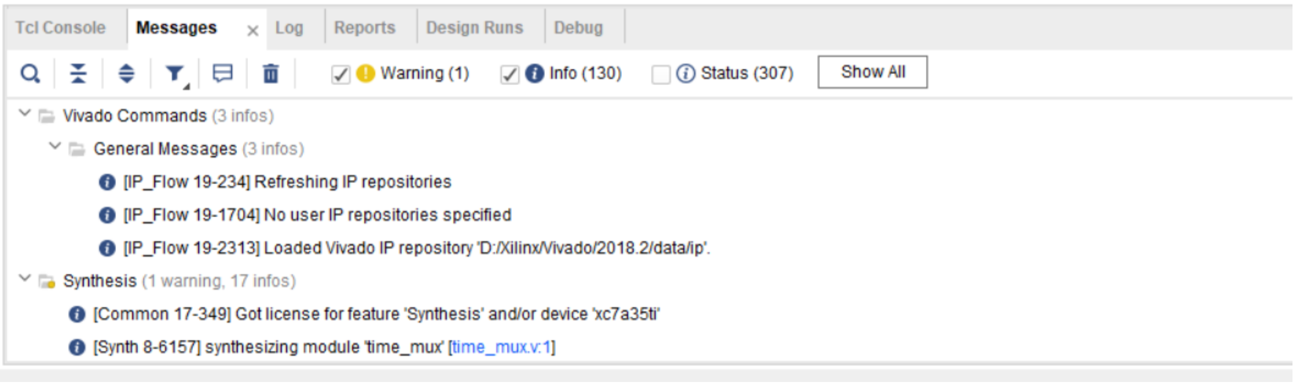
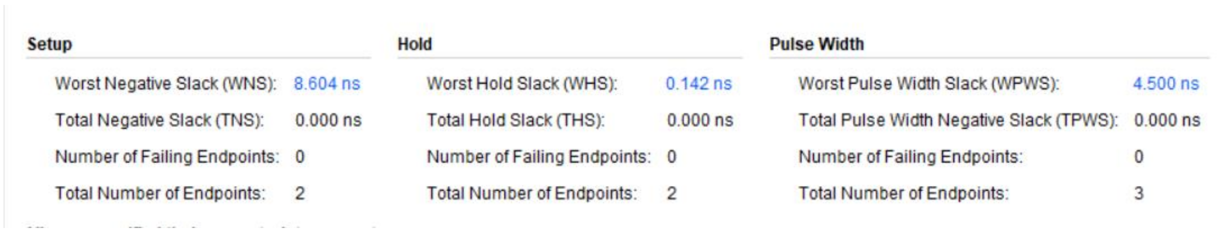


Figure 32 Synthesis Schematic



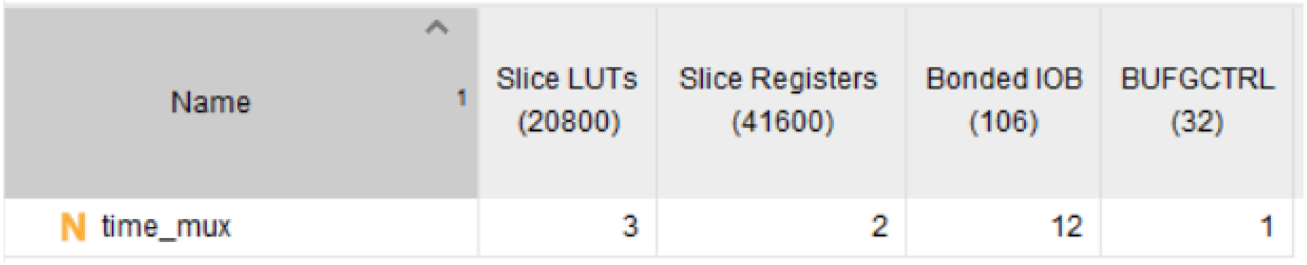
```
Vivado Commands (3 infos)
    General Messages (3 infos)
        [IP_Flow 19-234] Refreshing IP repositories
        [IP_Flow 19-1704] No user IP repositories specified
        [IP_Flow 19-2313] Loaded Vivado IP repository 'D:/Xilinx/Vivado/2018.2/data/ip'.
Synthesis (1 warning, 17 infos)
    [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a35ti'
    [Synth 8-6157] synthesizing module 'time_mux' [time_mux.v:1]
```

Figure 33 No Critical Warnings or Errors



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	8.604 ns	Worst Hold Slack (WHS):	0.142 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

Figure 34 Timing Summary



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

Figure 35 Utilization Summary

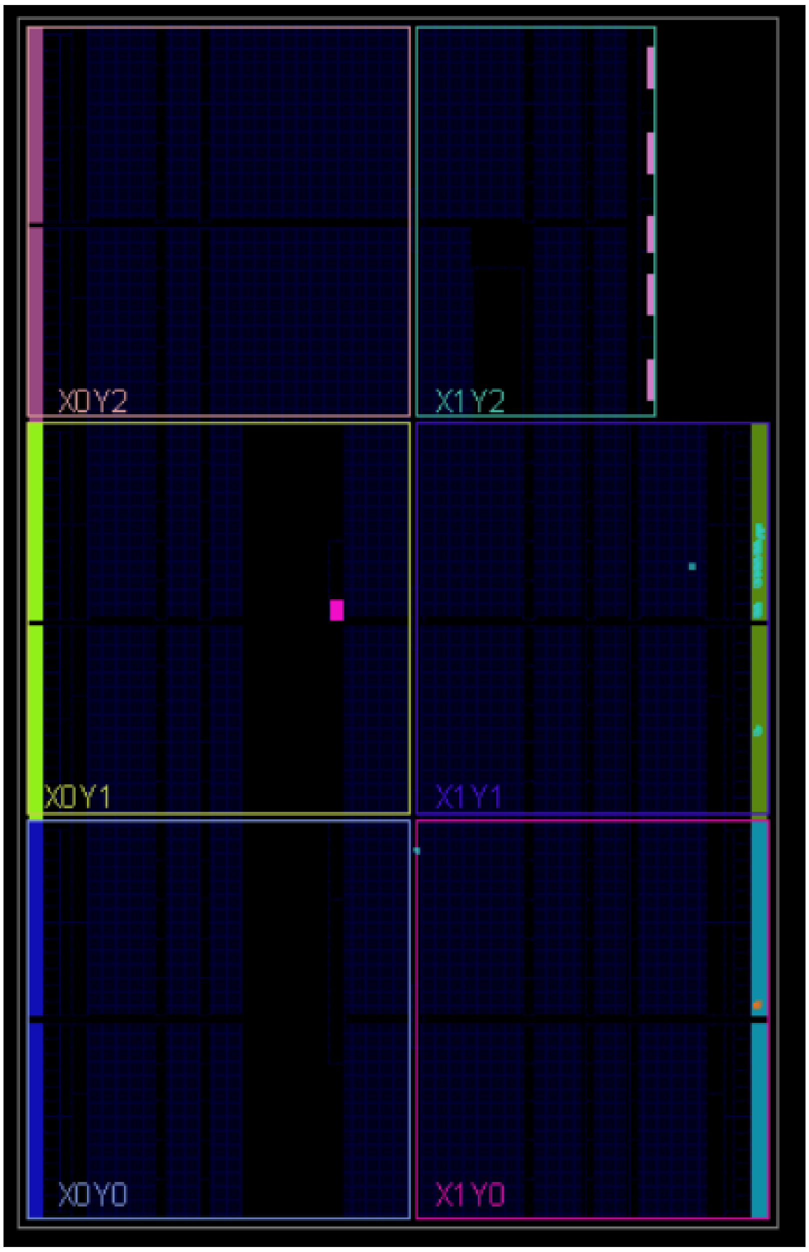
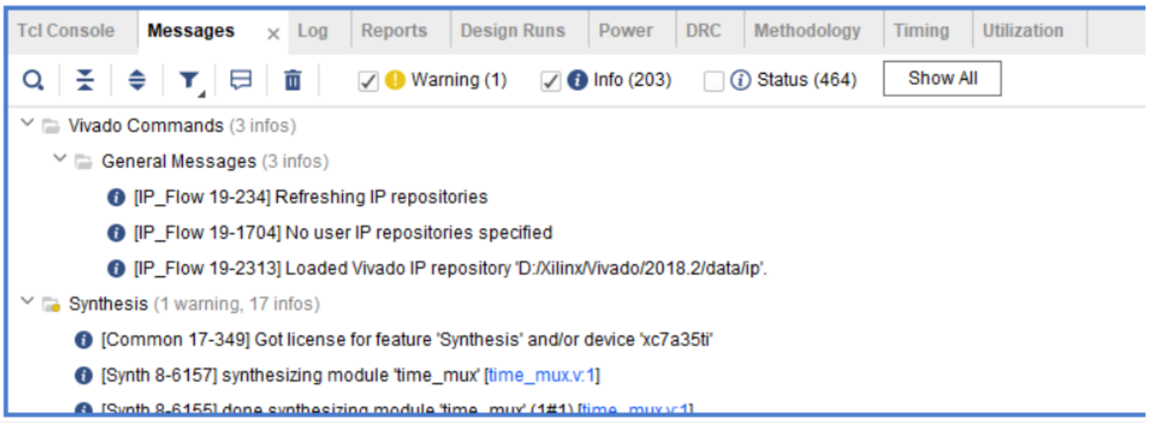
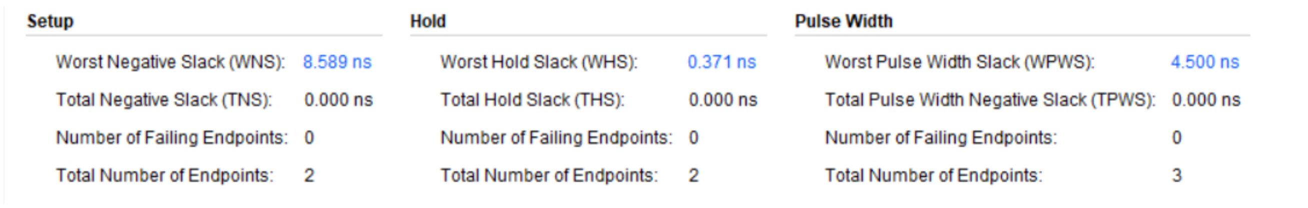


Figure 36 Device Snippet



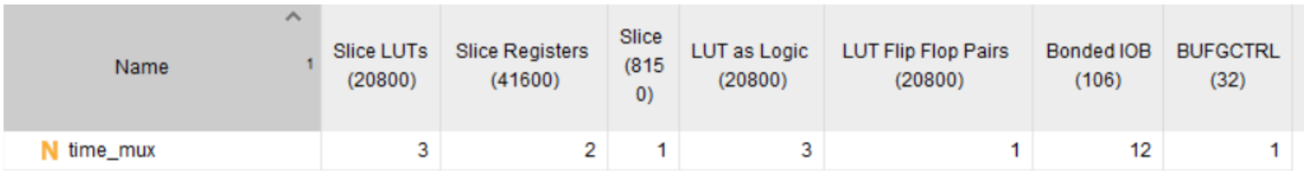
```
Vivado Commands (3 infos)
    General Messages (3 infos)
        [IP_Flow 19-234] Refreshing IP repositories
        [IP_Flow 19-1704] No user IP repositories specified
        [IP_Flow 19-2313] Loaded Vivado IP repository 'D:/Xilinx/Vivado/2018.2/data/ip'.
Synthesis (1 warning, 17 infos)
    [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a35ti'
    [Synth 8-6157] synthesizing module 'time_mux' [time_mux.v:1]
    [Synth 8-6155] done synthesizing module 'time_mux' (1#1) [time_mux.v:1]
```

Figure 37 No Critical Warnings or Errors



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	8.589 ns	Worst Hold Slack (WHS):	0.371 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

Figure 38 Timing Summary



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

Figure 39 Utilization Summary