**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;

            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;

        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:

A screen shot of a computer

AI-generated content may be incorrect.

Figure Testing Reset

A screen shot of a computer screen

AI-generated content may be incorrect.

Figure Opcode 0

A screenshot of a computer

AI-generated content may be incorrect.Figure Opcode 1

A screen shot of a computer

AI-generated content may be incorrect.

Figure Opcode 2

A screen shot of a computer

AI-generated content may be incorrect.

Figure Opcode 3

A green and white lines on a black background

AI-generated content may be incorrect.

Figure Invalid Case

Linting:

A screenshot of a computer

AI-generated content may be incorrect.

Figure 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]

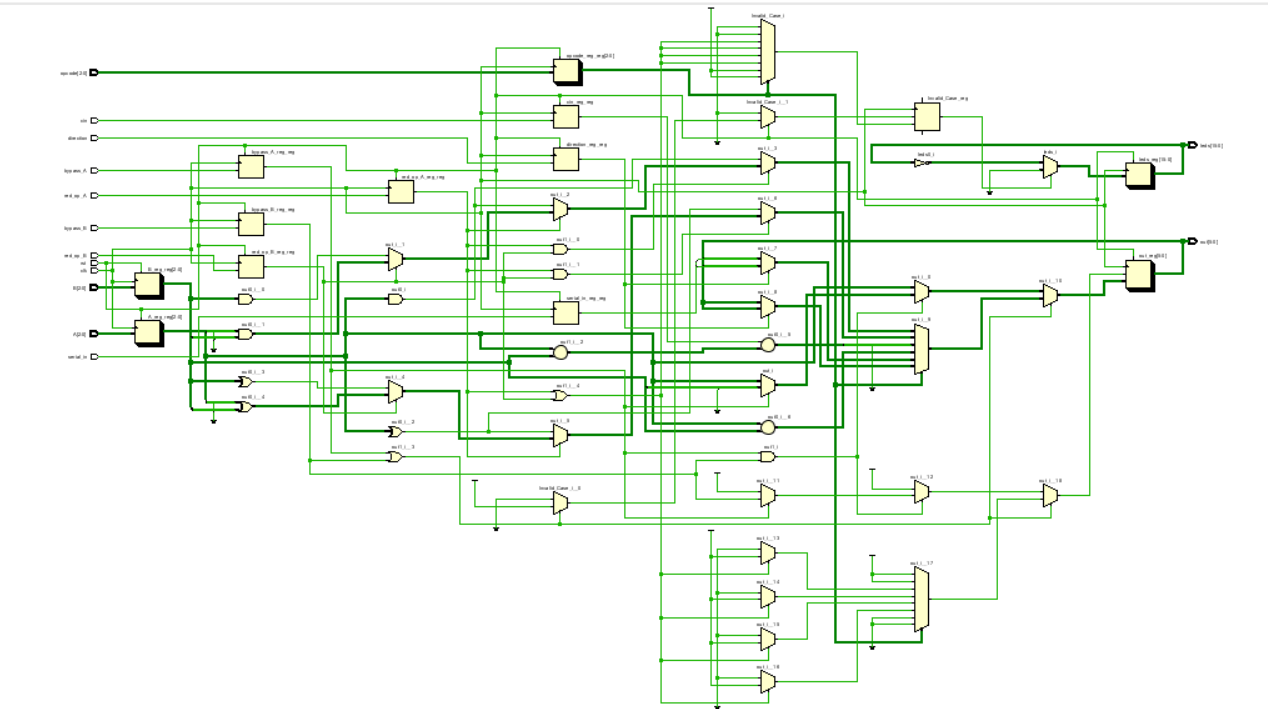


Figure Elaboration Schematic

A screenshot of a computer

AI-generated content may be incorrect.Figure No Critical Warnings

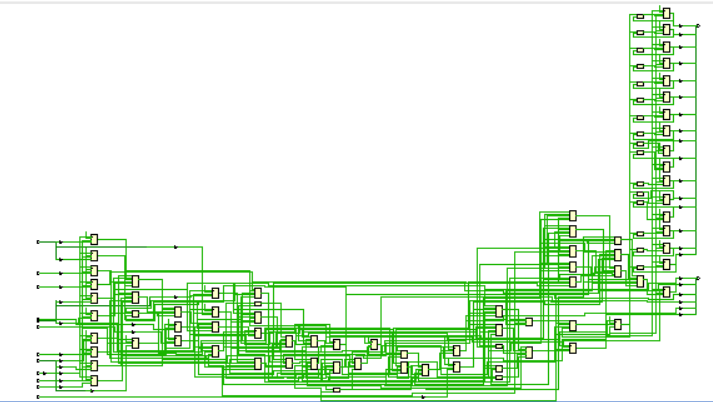


Figure Syntesis Schematic

A screenshot of a computer

AI-generated content may be incorrect.

Figure No Critical Warnings or Errors

A screenshot of a computer

AI-generated content may be incorrect.

Figure Syntehsis Timing Summary

A screenshot of a computer

AI-generated content may be incorrect.

Figure Syntehsis Utilization Summary

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure Device Snippet

A screenshot of a computer

AI-generated content may be incorrect.

Figure Messages

A screenshot of a computer

AI-generated content may be incorrect.

Figure Timing Summary

A screenshot of a computer

AI-generated content may be incorrect.

Figure 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

    // 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:

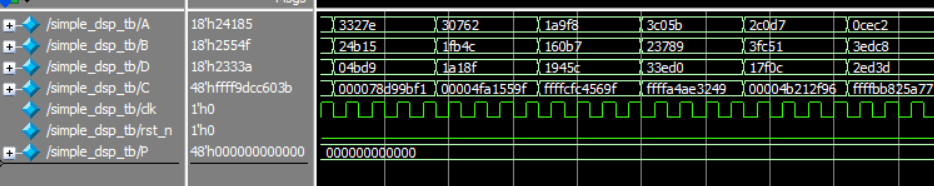


Figure Reset Test

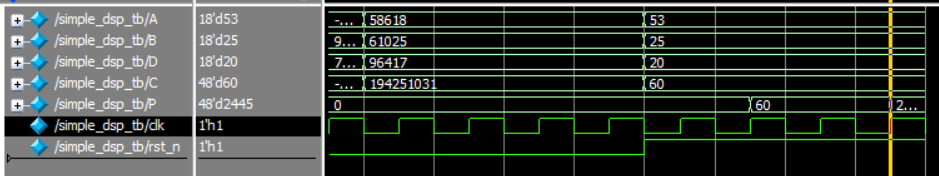


Figure Testing Path with fixed Value

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]

A diagram of a circuit

AI-generated content may be incorrect.

Figure Elaboration

A screenshot of a computer

AI-generated content may be incorrect.

Figure Nor Critical Warnings or Errors

A computer generated diagram of a staircase

AI-generated content may be incorrect.

Figure Synthesis

A screenshot of a computer

AI-generated content may be incorrect.

Figure No Critical Warnings or Errors

A screenshot of a computer

AI-generated content may be incorrect.

Figure Timing Summary

A screenshot of a computer

AI-generated content may be incorrect.

Figure Utilization Summary

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure Devices Snippet

A screenshot of a computer

AI-generated content may be incorrect.

Figure Timing Summary

A screenshot of a computer

AI-generated content may be incorrect.

Figure Utilization Summary

Q3: Time Division Mux

RTL Code:

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:

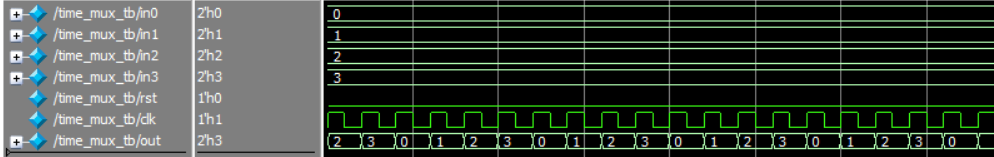


Figure Output Snippet

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]

A diagram of a circuit

AI-generated content may be incorrect.

Figure Elaboration Schematic

A screenshot of a computer

AI-generated content may be incorrect.

Figure No Critical Messages or Errors

A diagram of a computer

AI-generated content may be incorrect.

Figure Synthesis Schematic

A screenshot of a computer

AI-generated content may be incorrect.

Figure No Critical Warnings or Errors

A screenshot of a computer

AI-generated content may be incorrect.

Figure Timing Summary

A screenshot of a computer

AI-generated content may be incorrect.

Figure Utilization Summary

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure Device Snippet

A screenshot of a computer

AI-generated content may be incorrect.

Figure No Critical Warnings or Errors

A screenshot of a computer

AI-generated content may be incorrect.

Figure Timing Summary

A screenshot of a computer

AI-generated content may be incorrect.

Figure Utilization Summary