Project Digital Design

DSP48A1

Provided by

Saeed Nabawy

Submitted to

ENG / Kareem Waseem

Table of Contents

Introduction

In this project, we explore the DSP48A1 slice, which is based on the DSP48A slice used in Extended Spartan-3A FPGAs. This component is very useful for implementing digital signal processing (DSP) algorithms efficiently. It helps reduce the use of general-purpose FPGA logic, which means lower power consumption, better performance, and smarter use of the FPGA's resources.

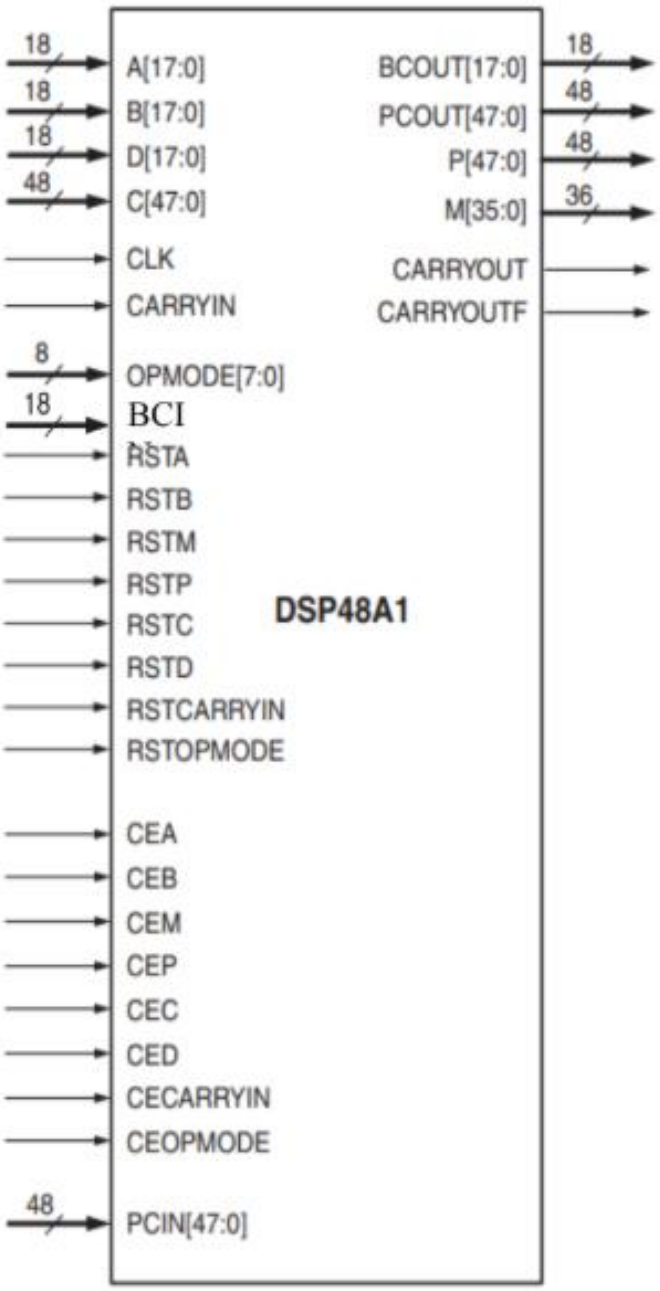
The DSP48A1 slice mainly includes three parts: an 18-bit input pre-adder, an 18x18 two's complement multiplier, and a 48-bit adder/subtracter/accumulator. These functions are essential in many DSP applications. It also has some advanced features like programmable pipelining and a 48-bit internal bus, which allows it to connect with other DSP slices for higher throughput and scalability.

One of the standout features is that the output from one DSP48A1 slice can be directly connected to the next without going through the FPGA's general routing. This makes it especially powerful for building DSP filters. Also, the slice supports cascading input streams and using multiple inputs (like from ports C and D) to perform more complex operations with fewer resources, which is great for optimizing performance in things like symmetric filters and multi-input arithmetic operations.

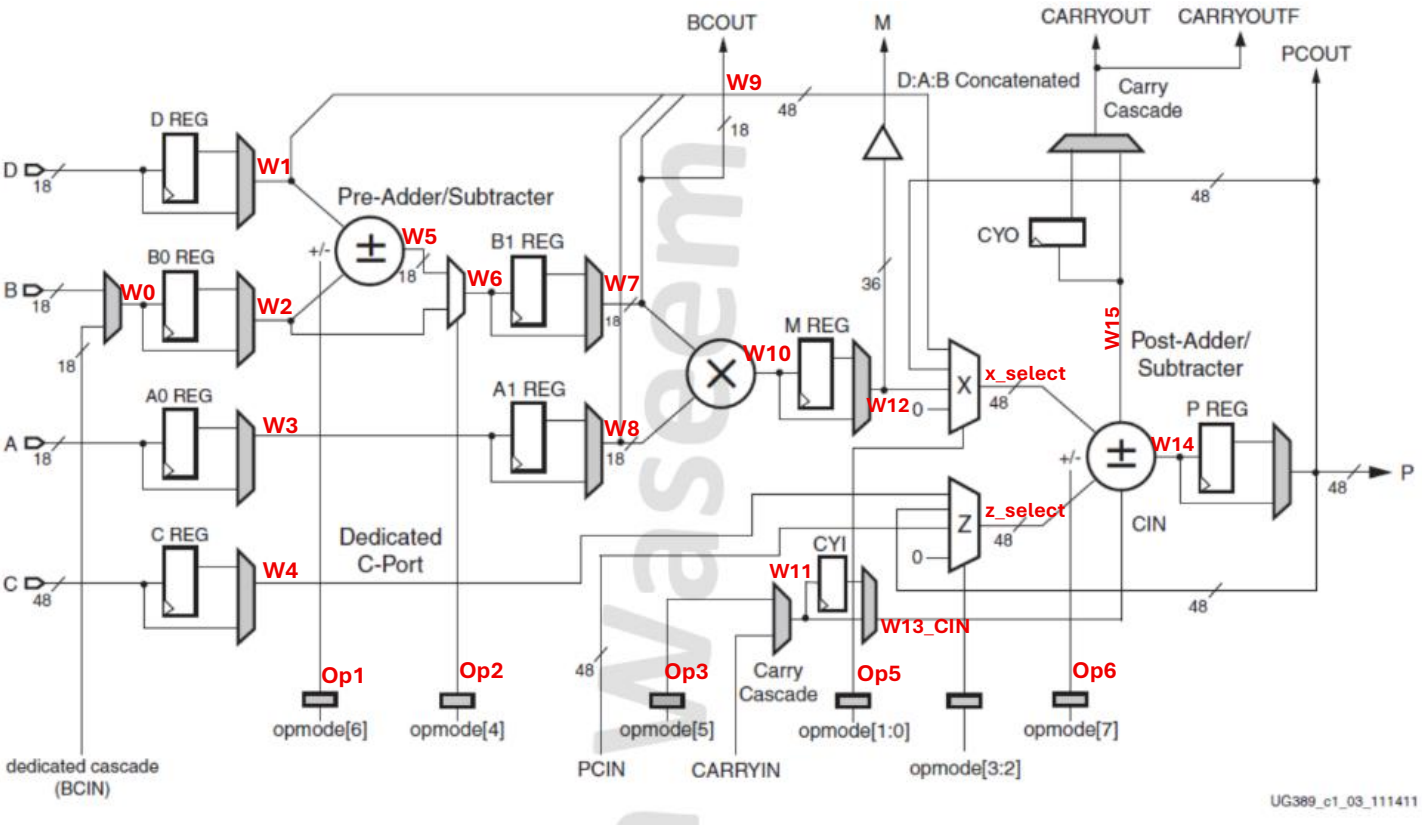
Project Description

The Spartan-6 family offers a high ratio of DSP48A1 slices to logic, making it ideal for math-intensive applications. Design DSP48A1 slice of the spartan6 FPGAs. The testbench for this design can be challenging so use directed test patterns whenever needed to verify the design and either have expected value to compare with in the testbench or check the result from the waveforms.

Use do file to run the Questasim flow.



``


BCOUT
M
CARRYOUT
CARRYOUTF
PCOUT
W9
D:A:B Concatenated
Carry Cascade
D REG
D
18
W1
Pre-Adder/Subtracter
W5
+/-
B1 REG
W6
W7
B0 REG
B
W0
W2
CYO
W15
Post-Adder/ Subtracter
M REG
W10
W12
X
x_select
A0 REG
A
W3
A1 REG
W8
P REG
W14
P
+/-
Z
z_select
CIN
C REG
C
48
W4
Dedicated C-Port
CYI
W11
W13_CIN
Op1
Op2
Op3
Op5
Op6
opmode[6]
opmode[4]
opmode[5]
opmode[1:0]
opmode[7]
opmode[3:2]
dedicated cascade (BCIN)
PCIN
CARRYIN
UG389_c1_03_111411

Parameter (Attributes)

Parameter	Function
A0REG, A1REG, B0REG, and B1REG	The A0REG, A1REG, B0REG, and B1REG attributes can take values of 0 or 1. these values define the number of pipeline registers in the A and B input paths. A0REG defaults to 0 (no register). A1REG defaults to 1 (register). B0REG defaults to 0 (no register) B1REG defaults to 1 (register). A0 and B0 are the first stages of the pipelines. A1 and B1 are the second stages of the pipelines
CREG, DREG, MREG, PREG, CARRYINREG, CARRYOUTREG, and OPMODEREG	These attributes can take a value of 0 or 1. The number defines the number of pipeline stages. Default: 1 (registered)
CARRYINSEL	The CARRYINSEL attribute is used in the carry cascade input, either the CARRYIN input will be considered or the value of opcode[5]. This attribute can be set to the string CARRYIN or OPMODE5. Default: OPMODE5. Tie the output of the mux to 0 if none of these string values exist.
B_INPUT	The B_INPUT attribute defines whether the input to the B port is routed from the B input (attribute = DIRECT) or the cascaded input (BCIN) from the previous DSP48A1 slice (attribute = CASCADE). Default: DIRECT. Tie the output of the mux to 0 if none of these string values exist.
RSTTYPE	The RSTTYPE attribute selects whether all resets for the DSP48A1 slice should have a synchronous or asynchronous reset capability. This attribute can be set to ASYNC or SYNC. Default: SYNC.

Data Ports

Signal Name	Function
A	18-bit data input to multiplier, and optionally to post-adder/subtracter depending on the value of OPMODE[1:0].
B	18-bit data input to pre-adder/subtracter, to multiplier depending on OPMODE[4], or to post-adder/subtracter depending on OPMODE[1:0].
C	48-bit data input to post-adder/subtracter.
D	18-bit data input to pre-adder/subtracter. D[11:0] are concatenated with A and B and optionally sent to post-adder/subtracter depending on the value of OPMODE[1:0].
CARRYIN	carry input to the post-adder/subtracter
M	36-bit buffered multiplier data output, routable to the FPGA logic. It is either the output of the M register (MREG = 1) or the direct output of the multiplier (MREG = 0).
P	Primary data output from the post-adder/subtracter. It is either the output of the P register (PREG = 1) or the direct output of the post-adder/subtracter (PREG = 0).
CARRYOUT	Cascade carry out signal from post-adder/subtracter. It can be registered in (CARRYOUTREG = 1) or unregistered (CARRYOUTREG = 0). This output is to be connected only to CARRYIN of adjacent DSP48A1 if multiple DSP blocks are used.
CARRYOUTF	Carry out signal from post-adder/subtracter for use in the FPGA logic. It is a copy of the CARRYOUT signal that can be routed to the user logic.

Control Input Ports

Signal Name	Function
CLK	DSP clock
OPMODE	Control input to select the arithmetic operations of the DSP48A1 slice.

Clock Enable Input Ports

Signal Name	Function
CEA	Clock enable for the A port registers: (A0REG & A1REG).
CEB	Clock enable for the B port registers: (B0REG & B1REG).
CEC	Clock enable for the C port registers (CREG).
CECARRYIN	Clock enable for the carry-in register (CYI) and the carry-out register (CYO).
CED	Clock enable for the D port register (DREG).
CEM	Clock enable for the multiplier register (MREG).
CEOPMODE	Clock enable for the opmode register (OPMODEREG).
CEP	Clock enable for the P output port registers (PREG = 1).

Reset Input Ports

All the resets are active high reset. They are either sync or async depending on the parameter RSTTYPE.

Signal Name	Function
RSTA	Reset for the A registers: (A0REG & A1REG).
RSTB	Reset for the B registers: (B0REG & B1REG).
RSTC	Reset for the C registers (CREG).
RSTCARRYIN	Reset for the carry-in register (CYI) and the carry-out register (CYO).
RSTD	Reset for the D register (DREG).
RSTM	Reset for the multiplier register (MREG).
RSTOPMODE	Reset for the opmode register (OPMODEREG).
RSTP	Reset for the P output registers (PREG = 1).

Cascade Ports

Signal Name	Function
BCOUT	Cascade output for Port B.
PCIN	Cascade input for Port P.
PCOUT	Cascade output for Port P.

OPMODE Pin Descriptions

Signal Name	Function
OPMODE[1:0]	**Specifies the source of the X input to the post-adder/subtracter** 0 – Specifies to place all zeros (disable the post-adder/subtracter and propagate the Z result to P) 1 – Use the multiplier product 2 – Use the P output signal (accumulator) 3 – Use the concatenated D:A:B input signals
OPMODE[3:2]	**Specifies the source of the Z input to the post-adder/subtracter** 0 – Specifies to place all zeros (disable the post-adder/subtracter and propagate the multiplier product or other X result to P) 1 – Use the PCIN 2 – Use the P output signal (accumulator) 3 – Use the C port
OPMODE[4]	**Specifies the use of the pre-adder/subtracter** 0 – Bypass the pre-adder supplying the data on port B directly to the multiplier 1 – Selects to use the pre-adder adding or subtracting the values on the B and D ports prior to the multiplier
OPMODE[5]	**Forces a value on the carry input of the carry-in register (CYI) or direct to the CIN to the post-adder. Only applicable when CARRYINSEL = OPMODE5**
OPMODE[6]	**Specifies whether the pre-adder/subtracter is an adder or subtracter** 0 – Specifies pre-adder/subtracter to perform an addition operation 1 – Specifies pre-adder/subtracter to perform a subtraction operation (D-B)
OPMODE[7]	**Specifies whether the post-adder/subtracter is an adder or subtracter** 0 – Specifies post-adder/subtracter to perform an addition operation 1 – Specifies post-adder/subtracter to perform a subtraction operation (Z-(X+CIN))

Modules

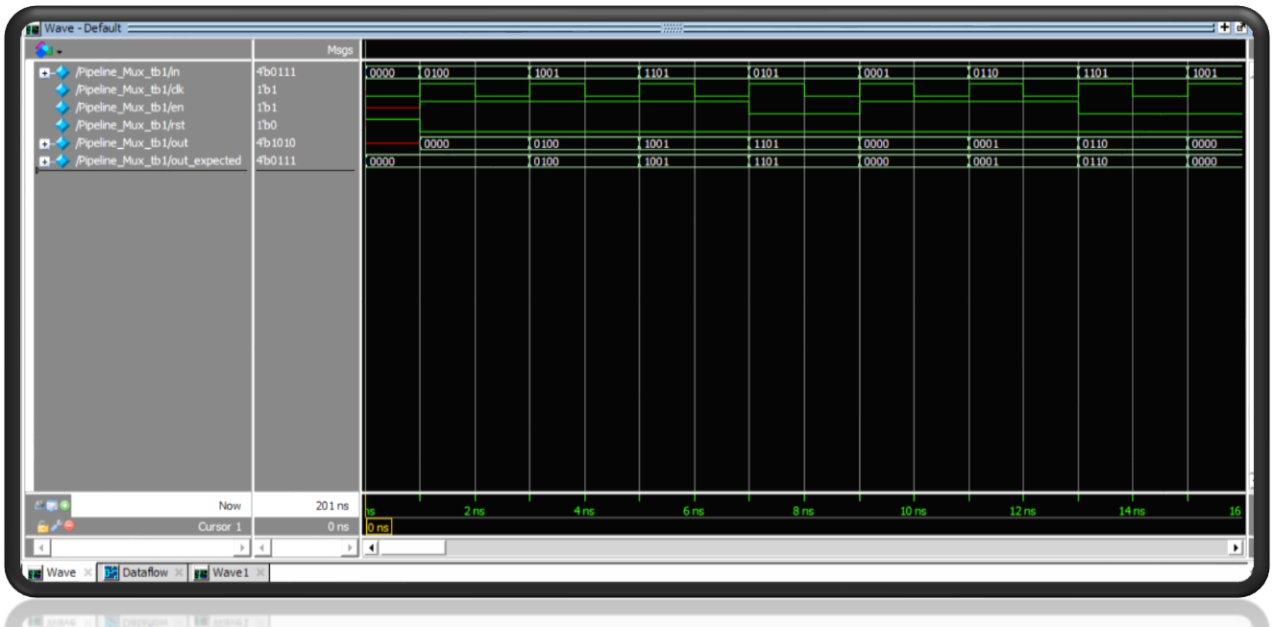
Pipeline_Mux

```
module Pipeline_Mux(in,clk,out,en,rst);

    parameter WIDTH = 18;
    parameter PIPELINE_ENABLE = 1;
    parameter RSTTYPE = "SYNC";

    input [WIDTH-1 : 0] in;
    input clk , en , rst;
    output reg [WIDTH-1 : 0] out;

    generate
        if(PIPELINE_ENABLE) begin
            if(RSTTYPE == "SYNC") begin
                always @(posedge clk) begin
                    if(rst) begin
                        out <= 0;
                    end
                    else if(en) begin
                        out <= in;
                    end
                end
            end
            else begin
                always @(posedge clk or posedge rst) begin
                    if(rst) begin
                        out <= 0;
                    end
                    else if(en) begin
                        out <= in;
                    end
                end
            end
        end
        else begin
            always @(*) begin
                out = in;
            end
        end
    endgenerate

endmodule
```

Pipeline_Mux Testbench 1

```
module Pipeline_Mux_tb1();

    reg [3 : 0] in;
    reg clk, en , rst;
    wire [3 : 0] out;
    reg [3 : 0] out_expected;

    Pipeline_Mux #(.WIDTH(4),.PIPELINE_ENABLE(1),.RSTTYPE("SYNC")) pm(.in(in),.clk(clk),.en(en) , .rst(rst),.out(out));

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

    integer i;
    initial begin
        rst = 1;
        @(posedge clk);
        rst = 0;
        for(i = 0 ; i < 100 ; i = i+1) begin
            in = $random;
            en = $random;
            @(posedge clk);
            if(en) begin
                out_expected = in;
            end
            else begin
                out_expected = 0;
            end
        end
        $stop;
    end

    always @(negedge clk) begin
        if(out != out_expected) begin
            $display("Error: out = %b, expected %b", out, out_expected);
            $stop;
        end
    end
endmodule
```

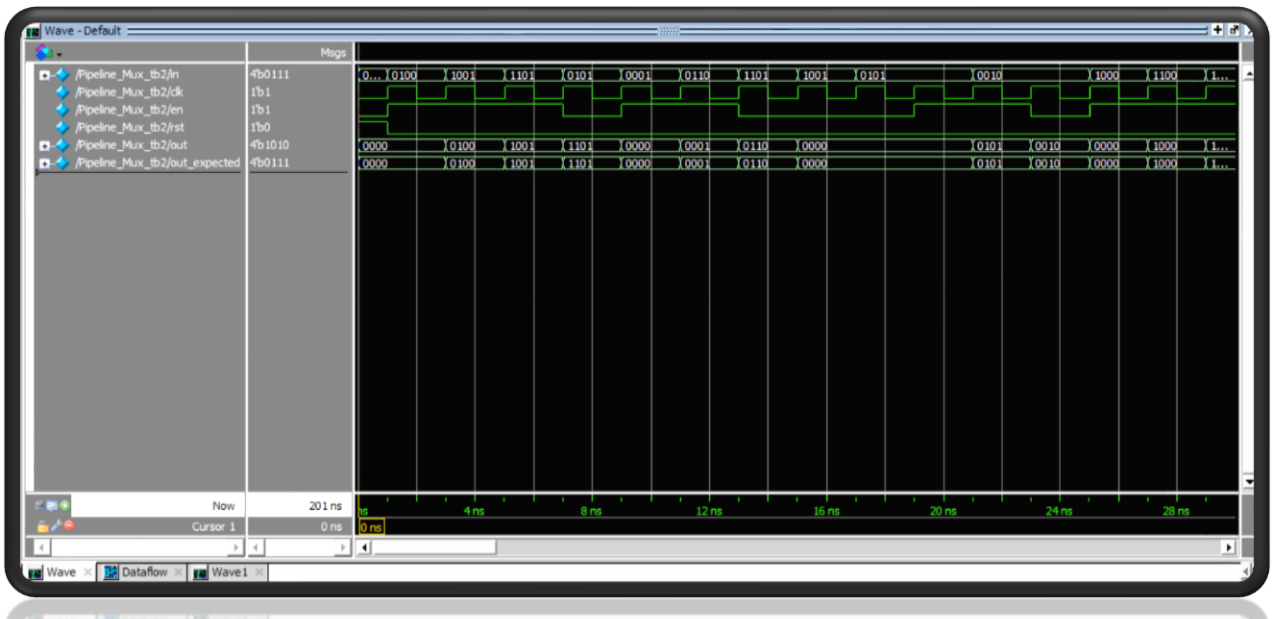
Simulation 1



Pipeline_Mux Testbench 2

```systemverilog
module Pipeline_Mux_tb2();

    reg [3 : 0] in;
    reg clk, en , rst;
    wire [3 : 0] out;
    reg [3 : 0] out_expected;

    Pipeline_Mux #(.WIDTH(4),.PIPELINE_ENABLE(1),.RSTTYPE("ASYNC")) pm(.in(in),.clk(clk),.en(en) , .rst(rst),.out(out));

    initial begin
        clk = 0;
        in = 0;
        out_expected = 0;
        en = 0;
        forever begin
            #1 clk = ~clk;
        end
    end

    integer i;
    initial begin
        rst = 1;
        @(posedge clk);
        rst = 0;
        for(i = 0 ; i < 100 ; i = i+1) begin
            in = $random;
            en = $random;
            @(posedge clk);
            if(en) begin
                out_expected = in;
            end
            else begin
                out_expected = 0;
            end
        end
        $stop;
    end

    always @(negedge clk) begin
        if(out != out_expected) begin
            $display("Error: out = %b, expected %b", out, out_expected);
            $stop;
        end
    end
endmodule
```

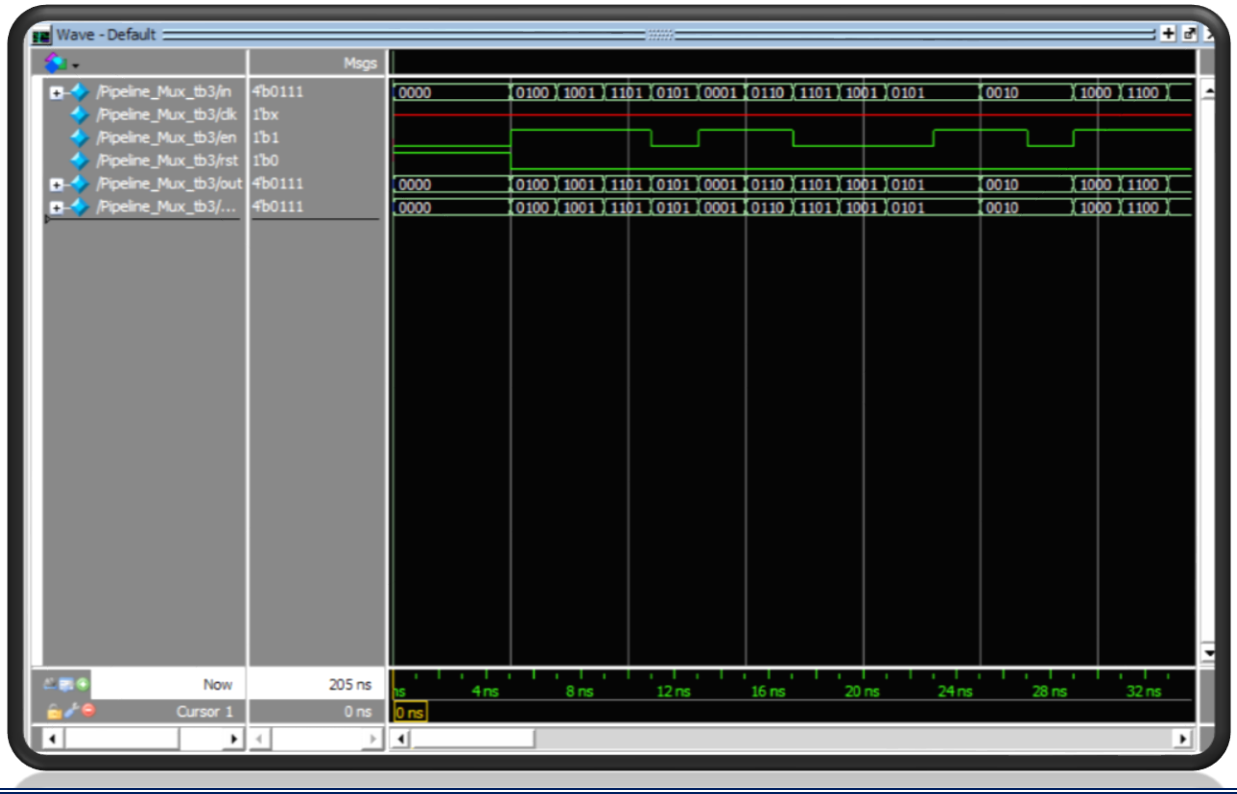
Simulation 2



Pipeline_Mux Testbench 3

```
module Pipeline_Mux_tb3();

    reg [3 : 0] in;
    reg clk, en , rst;
    wire [3 : 0] out;
    reg [3 : 0] out_expected;

    Pipeline_Mux #(.WIDTH(4),.PIPELINE_ENABLE(0),.RSTTYPE("SYNC")) pm(.in(in),.clk(clk),.en(en) , .rst(rst),.out(out));


    integer i;
    initial begin
        rst = 1;
        in = 0; en = 0; out_expected=0;
        #5;
        rst = 0;
        for(i = 0 ; i < 100 ; i = i+1) begin
            in = $random;
            en = $random;
            out_expected = in;
            #2;
        end
        $stop;
    end

    always @(in) begin
        #1;
        if(out != out_expected) begin
            $display("Error: out = %b, expected %b", out, out_expected);
            $stop;
        end
    end
endmodule
```

Simulation 3



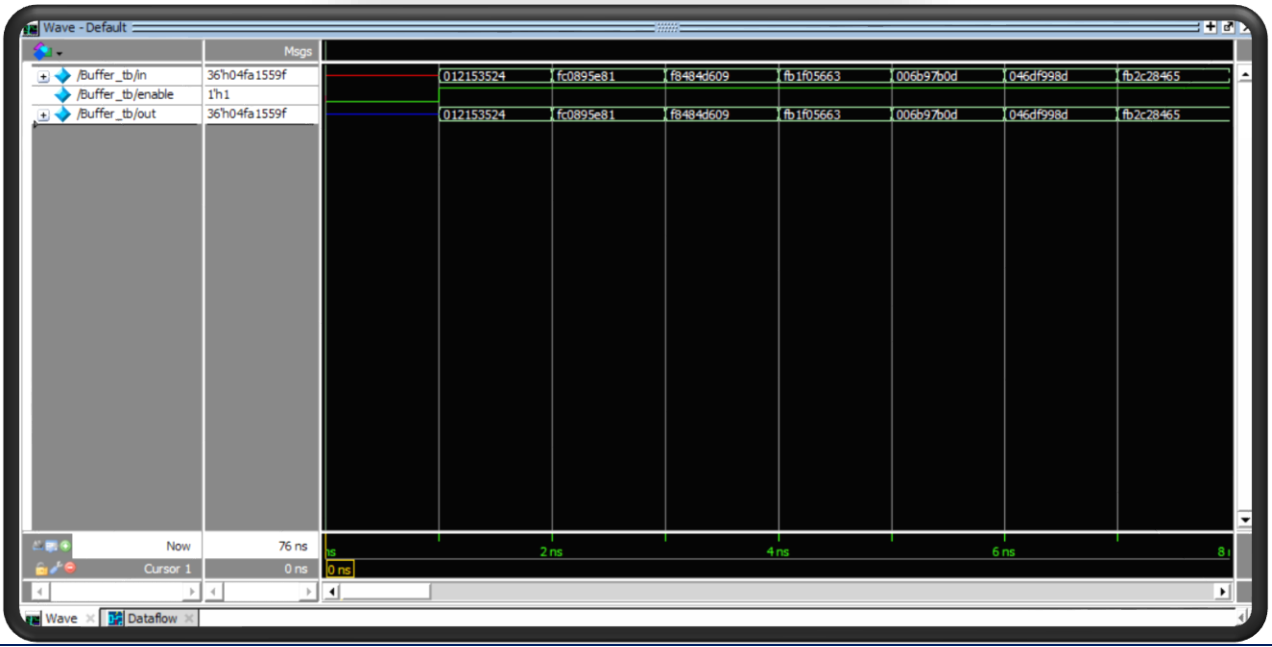
Buffer

```verilog
module Buffer(in,enable,out);

    parameter WIDTH = 36;

    input [WIDTH-1:0] in;
    input enable;
    output [WIDTH-1:0] out;

    assign out = enable ? in : {WIDTH{1'bz}};
endmodule
```

Buffer_Testbench

```verilog
module Buffer_tb();

    reg [35:0] in;
    reg enable;
    wire [35:0] out;

    Buffer bf(.in(in),.enable(enable),.out(out));

    integer i;
    initial begin
        enable = 0;
        #1;

        enable = 1;
        for(i = 0 ; i < 75 ; i = i + 1) begin
            in = $random;
            #1;
        end
        $stop;
    end

endmodule
```

Simulation



RTL Code

```
module DSP (A,B,D,C,PCIN,BCIN,CARRYIN,CARRYOUT,CARRYOUTF,
            clk, CEA, CEB, CEC, CECARRYIN, CED, CEM, CEOPMODE, CEP,
            RSTA, RSTB, RSTC, RSTCARRYIN, RSTD, RSTM, RSTOPMODE, RSTP,
            OPMODE,M,P,BCOUT,PCOUT
);

    parameter A0REG = 0; parameter A1REG = 1; parameter B0REG = 0; parameter B1REG = 1;
    parameter CREG = 1; parameter DREG = 1; parameter MREG = 1; parameter PREG = 1;
    parameter CARRYINREG = 1; parameter CARRYOUTREG = 1; parameter OPMODREG = 1;
    // OR --> "CARRYIN"
    parameter CARRYINSEL = "OPMOD5";
    // OR --> "CASCADE"
    parameter B_INPUT = "DIRECT";
    // OR --> "ASYNC"
    parameter RSTTYPE = "SYNC";


    input [17:0] A,B,D;
    input [47:0] C;
    input [47:0] PCIN;
    input [17:0] BCIN;
    input clk, CARRYIN,CEA, CEB, CEC, CECARRYIN, CED, CEM, CEOPMODE, CEP;
    input RSTA, RSTB, RSTC, RSTCARRYIN, RSTD, RSTM, RSTOPMODE, RSTP;
    input [7:0] OPMODE;
    output [35:0] M;
    output [47:0] P;
    output [17:0] BCOUT;
    output [47:0] PCOUT;
    output CARRYOUT,CARRYOUTF;

    wire [17:0] w0, w1, w2, w3, w7, w8;
    reg [17:0] w5, w6;
    reg [47:0] w9, x_select, z_select, w14;
    wire [35:0] w10,w12;
    wire [47:0] w4;
    wire op1, op2, op3, op6;
    wire [1:0] op4, op5;
    reg w15;
    wire w11,w13_CIN;

    generate
        // -------> Stage 0 <----------
        assign w0 = (B_INPUT == "DIRECT")? B : BCIN;

        // -------> Stage 1 <----------
        Pipeline_Mux #(.WIDTH(18), .PIPELINE_ENABLE(DREG), .RSTTYPE(RSTTYPE))
        D_REG (.in(D), .clk(clk), .en(CED), .rst(RSTD), .out(w1));

        Pipeline_Mux #(.WIDTH(18), .PIPELINE_ENABLE(B0REG), .RSTTYPE(RSTTYPE))
        B0_REG (.in(w0), .clk(clk), .en(CEB), .rst(RSTB), .out(w2));

        Pipeline_Mux #(.WIDTH(18), .PIPELINE_ENABLE(A0REG), .RSTTYPE(RSTTYPE))
        A0_REG (.in(A), .clk(clk), .en(CEA), .rst(RSTA), .out(w3));

        Pipeline_Mux #(.WIDTH(48), .PIPELINE_ENABLE(CREG), .RSTTYPE(RSTTYPE))
        C_REG (.in(C), .clk(clk), .en(CEC), .rst(RSTC), .out(w4));

        // -------> Buttom Stage <----------
        Pipeline_Mux #(.WIDTH(1), .PIPELINE_ENABLE(OPMODREG), .RSTTYPE(RSTTYPE))
        OP1_REG (.in(OPMODE[6]), .clk(clk), .en(CEOPMODE), .rst(RSTOPMODE), .out(op1));

        Pipeline_Mux #(.WIDTH(1), .PIPELINE_ENABLE(OPMODREG), .RSTTYPE(RSTTYPE))
        OP2_REG (.in(OPMODE[4]), .clk(clk), .en(CEOPMODE), .rst(RSTOPMODE), .out(op2));

        Pipeline_Mux #(.WIDTH(1), .PIPELINE_ENABLE(OPMODREG), .RSTTYPE(RSTTYPE))
        OP3_REG (.in(OPMODE[5]), .clk(clk), .en(CEOPMODE), .rst(RSTOPMODE), .out(op3));

        Pipeline_Mux #(.WIDTH(2), .PIPELINE_ENABLE(OPMODREG), .RSTTYPE(RSTTYPE))
        OP4_REG (.in(OPMODE[1:0]), .clk(clk), .en(CEOPMODE), .rst(RSTOPMODE), .out(op4));

        Pipeline_Mux #(.WIDTH(2), .PIPELINE_ENABLE(OPMODREG), .RSTTYPE(RSTTYPE))
        OP5_REG (.in(OPMODE[3:2]), .clk(clk), .en(CEOPMODE), .rst(RSTOPMODE), .out(op5));

        Pipeline_Mux #(.WIDTH(1), .PIPELINE_ENABLE(OPMODREG), .RSTTYPE(RSTTYPE))
        OP6_REG (.in(OPMODE[7]), .clk(clk), .en(CEOPMODE), .rst(RSTOPMODE), .out(op6));

        // -------> Stage 2 <----------
        always @(*) begin
            if(op1) begin
                w5 = w1 - w2;
            end
            else begin
                w5 = w1 + w2;
            end
        end
        always @(*) begin
            if(op2) begin
                w6 = w5;
            end
            else begin
                w6 = w2;
            end
```

```
        // -------> Stage 3 <----------
        Pipeline_Mux #(.WIDTH(18), .PIPELINE_ENABLE(B1REG), .RSTTYPE(RSTTYPE))
        B1_REG (.in(w6), .clk(clk), .en(CEB), .rst(RSTB), .out(w7));

        Pipeline_Mux #(.WIDTH(18), .PIPELINE_ENABLE(A1REG), .RSTTYPE(RSTTYPE))
        A1_REG (.in(w3), .clk(clk), .en(CEA), .rst(RSTA), .out(w8));

        always @(*) begin
            w9 = {w1[11:0],w8,w7};
        end

        assign w10 = w7 * w8;
        assign BCOUT = w7;

        assign w11 = (CARRYINSEL == "OPMOD5")? op3 : CARRYIN;

        // -------> Stage 4 <----------
        Pipeline_Mux #(.WIDTH(36), .PIPELINE_ENABLE(MREG), .RSTTYPE(RSTTYPE))
        M_REG (.in(w10), .clk(clk), .en(CEM), .rst(RSTM), .out(w12));

        Pipeline_Mux #(.WIDTH(1), .PIPELINE_ENABLE(CARRYINREG), .RSTTYPE(RSTTYPE))
        CYI_REG (.in(w11), .clk(clk), .en(CECARRYIN), .rst(RSTCARRYIN), .out(w13_CIN));

        Buffer M_BUF (.in(w12),.enable(1),.out(M));

        // -------> Stage 5 <----------
        always @(*) begin
            case(op4)
                2'b00 : x_select = 0;
                2'b01 : x_select = w12;
                2'b10 : x_select = P;
                2'b11 : x_select = w9;
            endcase
        end

        always @(*) begin
            case(op5)
                2'b00 : z_select = 0;
                2'b01 : z_select = PCIN;
                2'b10 : z_select = P;
                2'b11 : z_select = w4;
            endcase
        end

        // -------> Stage 6 <----------
        always @(*) begin
            if(op6) begin
                {w15,w14} = z_select - (x_select+w13_CIN);
            end
            else begin
                {w15,w14} = z_select + x_select + w13_CIN;
            end
        end

        Pipeline_Mux #(.WIDTH(1), .PIPELINE_ENABLE(CARRYOUTREG), .RSTTYPE(RSTTYPE))
        CYO_REG (.in(w15), .clk(clk), .en(CECARRYIN), .rst(RSTCARRYIN), .out(CARRYOUT));

        assign CARRYOUTF = CARRYOUT;

        Pipeline_Mux #(.WIDTH(48), .PIPELINE_ENABLE(PREG), .RSTTYPE(RSTTYPE))
        P_REG (.in(w14), .clk(clk), .en(CEP), .rst(RSTP), .out(P));

        assign PCOUT = P;

    endgenerate
endmodule
```

Testbench

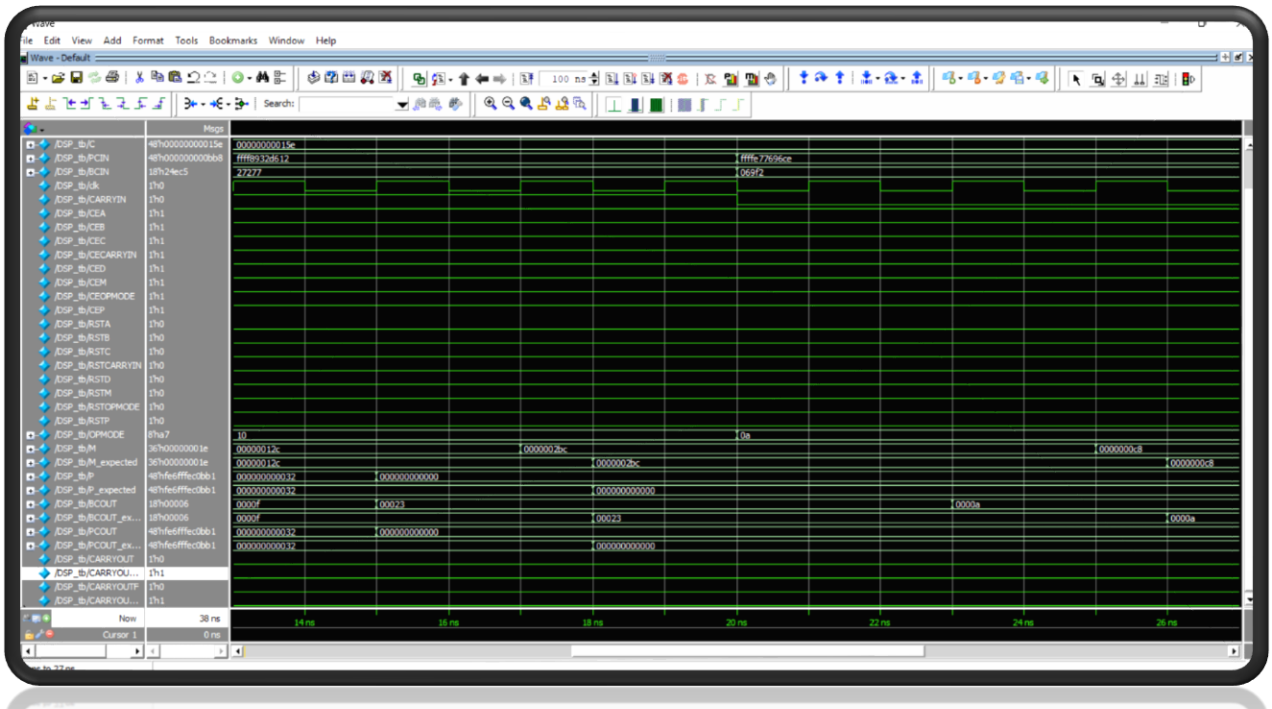
```
module DSP_tb();

    reg [17:0] A, B, D;
    reg [47:0] C;
    reg [47:0] PCIN;
    reg [17:0] BCIN;
    reg clk, CARRYIN;
    reg CEA, CEB, CEC, CECARRYIN, CED, CEM, CEOPMODE, CEP;
    reg RSTA, RSTB, RSTC, RSTCARRYIN, RSTD, RSTM, RSTOPMODE, RSTP;
    reg [7:0] OPMODE;
    wire [35:0] M;
    wire [47:0] P;
    wire [17:0] BCOUT;
    wire [47:0] PCOUT;
    wire CARRYOUT, CARRYOUTF;

    reg [35:0] M_expected;
    reg [47:0] P_expected;
    reg [17:0] BCOUT_expected;
    reg [47:0] PCOUT_expected;
    reg CARRYOUT_expected, CARRYOUTF_expected;

    DSP dut(
        .A(A), .B(B), .D(D), .C(C), .PCIN(PCIN), .BCIN(BCIN),
        .CARRYIN(CARRYIN), .CARRYOUT(CARRYOUT), .CARRYOUTF(CARRYOUTF),
        .clk(clk),
        .CEA(CEA), .CEB(CEB), .CEC(CEC), .CECARRYIN(CECARRYIN),
        .CED(CED), .CEM(CEM), .CEOPMODE(CEOPMODE), .CEP(CEP),
        .RSTA(RSTA), .RSTB(RSTB), .RSTC(RSTC), .RSTCARRYIN(RSTCARRYIN),
        .RSTD(RSTD), .RSTM(RSTM), .RSTOPMODE(RSTOPMODE), .RSTP(RSTP),
        .OPMODE(OPMODE), .M(M), .P(P), .BCOUT(BCOUT), .PCOUT(PCOUT)
    );

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

    initial begin
        // Test 1
        RSTA = 1; RSTB = 1; RSTC = 1;
        RSTD = 1; RSTM = 1; RSTP = 1;
        RSTOPMODE = 1; RSTCARRYIN = 1;

        A = $random; B = $random; C = $random; D = $random;
        CARRYIN = $random; PCIN = $random; BCIN = $random;
        CEA = $random; CEB = $random; CEC = $random; CECARRYIN = $random;
        CED = $random; CEM = $random; CEOPMODE = $random; CEP = $random;
        OPMODE = $random;

        M_expected = 0; P_expected = 0;
        BCOUT_expected = 0; PCOUT_expected = 0;
        CARRYOUT_expected = 0; CARRYOUTF_expected = 0;

        @(negedge clk);

        if ({M_expected, P_expected, BCOUT_expected, PCOUT_expected, CARRYOUT_expected, CARRYOUTF_expected} !=
            {M, P, BCOUT, PCOUT, CARRYOUT, CARRYOUTF}) begin
            $display("ERROR: Initial values not zero");
            $stop;
        end

        // Test 2
        RSTA = 0; RSTB = 0; RSTC = 0;
        RSTD = 0; RSTM = 0; RSTP = 0;
        RSTOPMODE = 0; RSTCARRYIN = 0;

        CEA = 1; CEB = 1; CEC = 1; CECARRYIN = 1; CED = 1; CEM = 1;
        CEOPMODE = 1; CEP = 1;

        OPMODE = 8'b11011101;
        A = 20; B = 10; C = 350; D = 25;
        BCIN = $random; PCIN = $random; CARRYIN = $random;
        @(negedge clk);
        @(negedge clk);
        @(negedge clk);
        @(negedge clk);
        M_expected = 'h12c; P_expected = 'h32;
        BCOUT_expected = 'hf; PCOUT_expected = 'h32;
        CARRYOUT_expected = 0; CARRYOUTF_expected = 0;
        @(negedge clk);

        if ({M_expected, P_expected, BCOUT_expected, PCOUT_expected, CARRYOUT_expected , CARRYOUTF_expected} !=
            {M, P, BCOUT, PCOUT, CARRYOUT , CARRYOUTF}) begin
            $display("ERROR: Initial values not zero");
            $stop;
        end
```

```
        // Test 3
        OPMODE = 8'b00010000;
        A = 20; B = 10; C = 350; D = 25;
        BCIN = $random; PCIN = $random; CARRYIN = $random;
        @(negedge clk);
        @(negedge clk);
        @(negedge clk);
        M_expected = 'h2bc; P_expected = 0;
        BCOUT_expected = 'h23; PCOUT_expected = 0;
        CARRYOUT_expected = 0; CARRYOUTF_expected = 0;
        @(negedge clk);

        if({M_expected,P_expected,BCOUT_expected,PCOUT_expected,CARRYOUT_expected,CARRYOUTF_expected} != {M,P,BCOUT,PCOUT,CARRYOUT,CARRYOUTF}) begin
            $display("ERROR: Initial values not zero");
            $stop;
        end

        // Test 4
        OPMODE = 8'b00001010;
        A = 20; B = 10; C = 350; D = 25;
        BCIN = $random; PCIN = $random; CARRYIN = $random;
        @(negedge clk);
        @(negedge clk);
        @(negedge clk);
        M_expected = 'hc8; P_expected = 0;
        BCOUT_expected = 'ha; PCOUT_expected = 0;
        CARRYOUT_expected = 0; CARRYOUTF_expected = 0;
        @(negedge clk);

        if({M_expected,P_expected,BCOUT_expected,PCOUT_expected,CARRYOUT_expected,CARRYOUTF_expected} != {M,P,BCOUT,PCOUT,CARRYOUT,CARRYOUTF}) begin
            $display("ERROR: Initial values not zero");
            $stop;
        end

        // Test 5
        OPMODE = 8'b10100111;
        A = 5; B = 6; C = 350; D = 25; PCIN = 3000;
        BCIN = $random; CARRYIN = $random;
        @(negedge clk);
        @(negedge clk);
        @(negedge clk);
        @(negedge clk);
        M_expected = 'h1e; P_expected = 'hfe6fffec0bb1;
        BCOUT_expected = 'h6; PCOUT_expected = 'hfe6fffec0bb1;
        CARRYOUT_expected = 1; CARRYOUTF_expected = 1;
        @(negedge clk);

        if({M_expected,P_expected,BCOUT_expected,PCOUT_expected,CARRYOUT_expected,CARRYOUTF_expected} != {M,P,BCOUT,PCOUT,CARRYOUT,CARRYOUTF}) begin
            $display("ERROR: Initial values not zero");
            $stop;
        end

        $stop;
    end
endmodule
```

Do File

```
vlib work
vlog Buffer.v Pipeline_Mux.v DSP.v DSP_tb.v
vsim -voptargs=+acc work.DSP_tb
add wave *
run -all
#quit -sim
```

QuestaSim

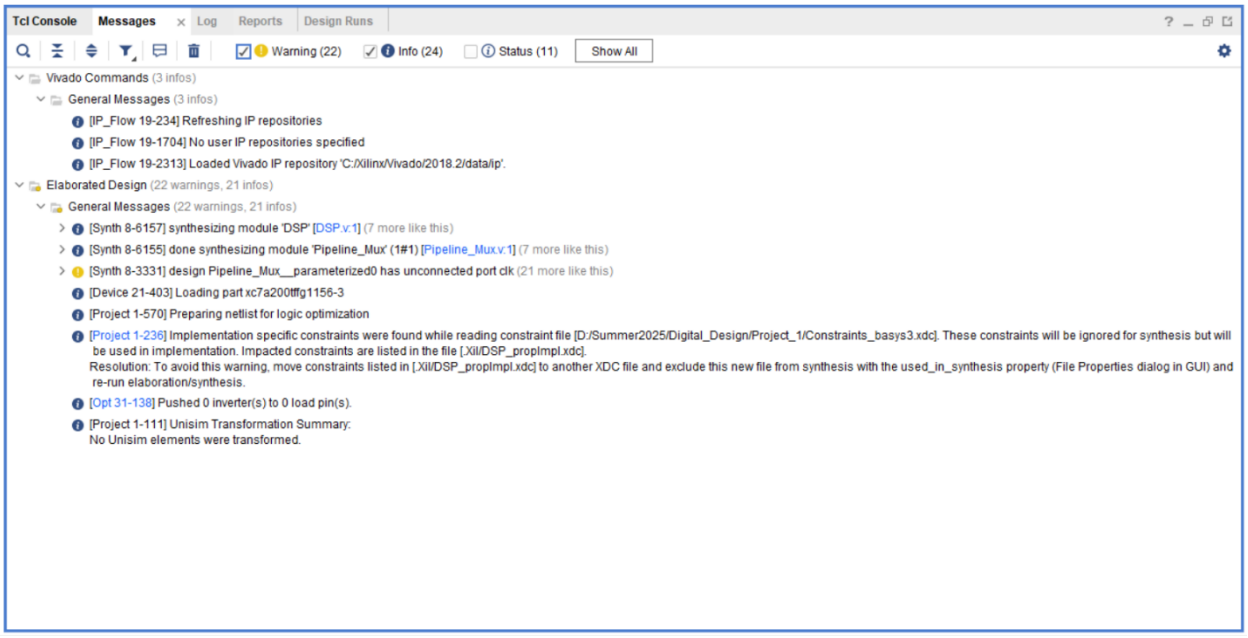


Constrain File

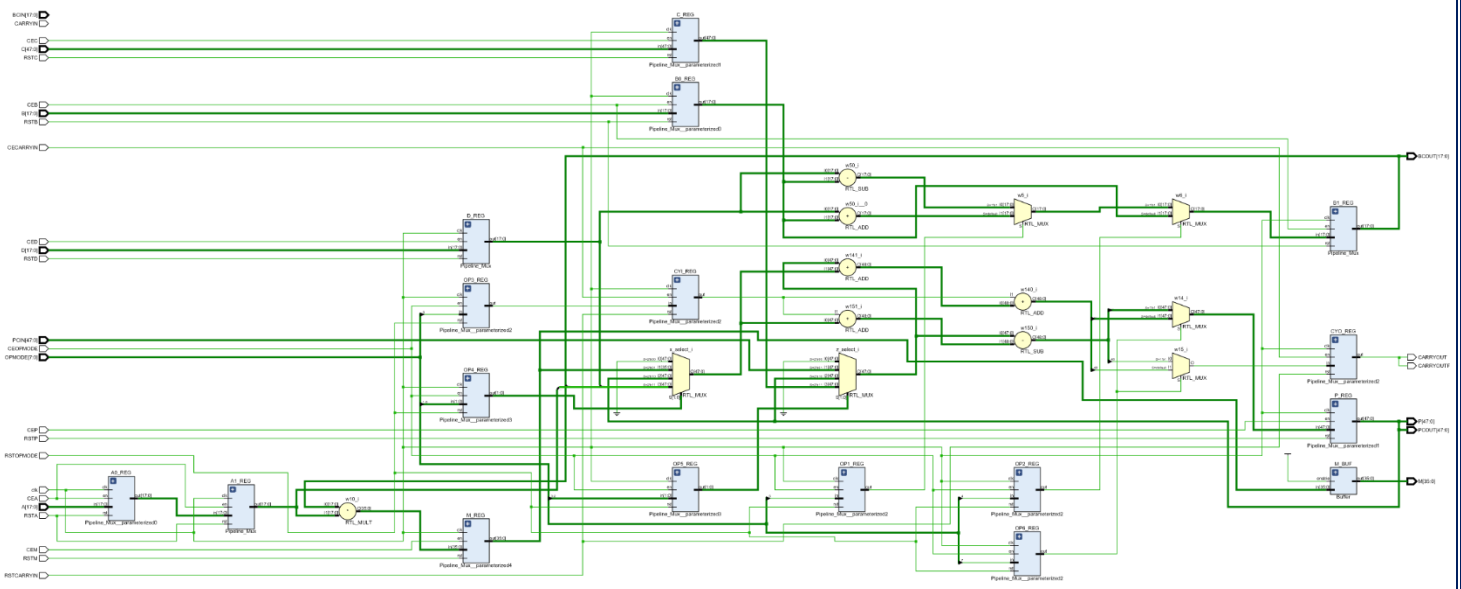
```
## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## 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]

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 48 [get_debug_ports u_ila_0/probe0]
connect_debug_port u_ila_0/probe0 [get_nets [list {PCOUT_OBUF[0]} {PCOUT_OBUF[1]} {PCOUT_OBUF[2]} {PCOUT_OBUF[3]} {PCOUT_OBUF[4]} {PCOUT_OBUF[5]} {PCOUT_OBUF[6]}
{PCOUT_OBUF[7]} {PCOUT_OBUF[8]} {PCOUT_OBUF[9]} {PCOUT_OBUF[10]} {PCOUT_OBUF[11]} {PCOUT_OBUF[12]} {PCOUT_OBUF[13]} {PCOUT_OBUF[14]} {PCOUT_OBUF[15]}
{PCOUT_OBUF[16]} {PCOUT_OBUF[17]} {PCOUT_OBUF[18]} {PCOUT_OBUF[19]} {PCOUT_OBUF[20]} {PCOUT_OBUF[21]} {PCOUT_OBUF[22]} {PCOUT_OBUF[23]} {PCOUT_OBUF[24]}
{PCOUT_OBUF[25]} {PCOUT_OBUF[26]} {PCOUT_OBUF[27]} {PCOUT_OBUF[28]} {PCOUT_OBUF[29]} {PCOUT_OBUF[30]} {PCOUT_OBUF[31]} {PCOUT_OBUF[32]} {PCOUT_OBUF[33]}
{PCOUT_OBUF[34]} {PCOUT_OBUF[35]} {PCOUT_OBUF[36]} {PCOUT_OBUF[37]} {PCOUT_OBUF[38]} {PCOUT_OBUF[39]} {PCOUT_OBUF[40]} {PCOUT_OBUF[41]} {PCOUT_OBUF[42]}
{PCOUT_OBUF[43]} {PCOUT_OBUF[44]} {PCOUT_OBUF[45]} {PCOUT_OBUF[46]} {PCOUT_OBUF[47]}]]
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 18 [get_debug_ports u_ila_0/probe1]
connect_debug_port u_ila_0/probe1 [get_nets [list {A_IBUF[0]} {A_IBUF[1]} {A_IBUF[2]} {A_IBUF[3]} {A_IBUF[4]} {A_IBUF[5]} {A_IBUF[6]} {A_IBUF[7]} {A_IBUF[8]}
{A_IBUF[9]} {A_IBUF[10]} {A_IBUF[11]} {A_IBUF[12]} {A_IBUF[13]} {A_IBUF[14]} {A_IBUF[15]} {A_IBUF[16]} {A_IBUF[17]}]]
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 36 [get_debug_ports u_ila_0/probe2]
connect_debug_port u_ila_0/probe2 [get_nets [list {M_OBUF[0]} {M_OBUF[1]} {M_OBUF[2]} {M_OBUF[3]} {M_OBUF[4]} {M_OBUF[5]} {M_OBUF[6]} {M_OBUF[7]} {M_OBUF[8]}
{M_OBUF[9]} {M_OBUF[10]} {M_OBUF[11]} {M_OBUF[12]} {M_OBUF[13]} {M_OBUF[14]} {M_OBUF[15]} {M_OBUF[16]} {M_OBUF[17]} {M_OBUF[18]} {M_OBUF[19]} {M_OBUF[20]}
{M_OBUF[21]} {M_OBUF[22]} {M_OBUF[23]} {M_OBUF[24]} {M_OBUF[25]} {M_OBUF[26]} {M_OBUF[27]} {M_OBUF[28]} {M_OBUF[29]} {M_OBUF[30]} {M_OBUF[31]} {M_OBUF[32]}
{M_OBUF[33]} {M_OBUF[34]} {M_OBUF[35]}]]
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 18 [get_debug_ports u_ila_0/probe3]
connect_debug_port u_ila_0/probe3 [get_nets [list {B_IBUF[0]} {B_IBUF[1]} {B_IBUF[2]} {B_IBUF[3]} {B_IBUF[4]} {B_IBUF[5]} {B_IBUF[6]} {B_IBUF[7]} {B_IBUF[8]}
{B_IBUF[9]} {B_IBUF[10]} {B_IBUF[11]} {B_IBUF[12]} {B_IBUF[13]} {B_IBUF[14]} {B_IBUF[15]} {B_IBUF[16]} {B_IBUF[17]}]]
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 18 [get_debug_ports u_ila_0/probe4]
connect_debug_port u_ila_0/probe4 [get_nets [list {BCOUT_OBUF[0]} {BCOUT_OBUF[1]} {BCOUT_OBUF[2]} {BCOUT_OBUF[3]} {BCOUT_OBUF[4]} {BCOUT_OBUF[5]} {BCOUT_OBUF
[6]} {BCOUT_OBUF[7]} {BCOUT_OBUF[8]} {BCOUT_OBUF[9]} {BCOUT_OBUF[10]} {BCOUT_OBUF[11]} {BCOUT_OBUF[12]} {BCOUT_OBUF[13]} {BCOUT_OBUF[14]} {BCOUT_OBUF[15]}
{BCOUT_OBUF[16]} {BCOUT_OBUF[17]}]]
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 48 [get_debug_ports u_ila_0/probe5]
connect_debug_port u_ila_0/probe5 [get_nets [list {C_IBUF[0]} {C_IBUF[1]} {C_IBUF[2]} {C_IBUF[3]} {C_IBUF[4]} {C_IBUF[5]} {C_IBUF[6]} {C_IBUF[7]} {C_IBUF[8]}
{C_IBUF[9]} {C_IBUF[10]} {C_IBUF[11]} {C_IBUF[12]} {C_IBUF[13]} {C_IBUF[14]} {C_IBUF[15]} {C_IBUF[16]} {C_IBUF[17]} {C_IBUF[18]} {C_IBUF[19]} {C_IBUF[20]}
{C_IBUF[21]} {C_IBUF[22]} {C_IBUF[23]} {C_IBUF[24]} {C_IBUF[25]} {C_IBUF[26]} {C_IBUF[27]} {C_IBUF[28]} {C_IBUF[29]} {C_IBUF[30]} {C_IBUF[31]} {C_IBUF[32]}
{C_IBUF[33]} {C_IBUF[34]} {C_IBUF[35]} {C_IBUF[36]} {C_IBUF[37]} {C_IBUF[38]} {C_IBUF[39]} {C_IBUF[40]} {C_IBUF[41]} {C_IBUF[42]} {C_IBUF[43]} {C_IBUF[44]}
{C_IBUF[45]} {C_IBUF[46]} {C_IBUF[47]}]]
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 8 [get_debug_ports u_ila_0/probe6]
connect_debug_port u_ila_0/probe6 [get_nets [list {OPMODE_IBUF[0]} {OPMODE_IBUF[1]} {OPMODE_IBUF[2]} {OPMODE_IBUF[3]} {OPMODE_IBUF[4]} {OPMODE_IBUF[5]}
{OPMODE_IBUF[6]} {OPMODE_IBUF[7]}]]
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 48 [get_debug_ports u_ila_0/probe7]
connect_debug_port u_ila_0/probe7 [get_nets [list {PCIN_IBUF[0]} {PCIN_IBUF[1]} {PCIN_IBUF[2]} {PCIN_IBUF[3]} {PCIN_IBUF[4]} {PCIN_IBUF[5]} {PCIN_IBUF[6]}
{PCIN_IBUF[7]} {PCIN_IBUF[8]} {PCIN_IBUF[9]} {PCIN_IBUF[10]} {PCIN_IBUF[11]} {PCIN_IBUF[12]} {PCIN_IBUF[13]} {PCIN_IBUF[14]} {PCIN_IBUF[15]} {PCIN_IBUF[16]}
{PCIN_IBUF[17]} {PCIN_IBUF[18]} {PCIN_IBUF[19]} {PCIN_IBUF[20]} {PCIN_IBUF[21]} {PCIN_IBUF[22]} {PCIN_IBUF[23]} {PCIN_IBUF[24]} {PCIN_IBUF[25]} {PCIN_IBUF
[26]} {PCIN_IBUF[27]} {PCIN_IBUF[28]} {PCIN_IBUF[29]} {PCIN_IBUF[30]} {PCIN_IBUF[31]} {PCIN_IBUF[32]} {PCIN_IBUF[33]} {PCIN_IBUF[34]} {PCIN_IBUF[35]}
{PCIN_IBUF[36]} {PCIN_IBUF[37]} {PCIN_IBUF[38]} {PCIN_IBUF[39]} {PCIN_IBUF[40]} {PCIN_IBUF[41]} {PCIN_IBUF[42]} {PCIN_IBUF[43]} {PCIN_IBUF[44]} {PCIN_IBUF
[45]} {PCIN_IBUF[46]} {PCIN_IBUF[47]}]]
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 18 [get_debug_ports u_ila_0/probe8]
connect_debug_port u_ila_0/probe8 [get_nets [list {D_IBUF[0]} {D_IBUF[1]} {D_IBUF[2]} {D_IBUF[3]} {D_IBUF[4]} {D_IBUF[5]} {D_IBUF[6]} {D_IBUF[7]} {D_IBUF[8]}
{D_IBUF[9]} {D_IBUF[10]} {D_IBUF[11]} {D_IBUF[12]} {D_IBUF[13]} {D_IBUF[14]} {D_IBUF[15]} {D_IBUF[16]} {D_IBUF[17]}]]
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 CARRYOUTF_OBUF]]
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 CEA_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 CEB_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 CEC_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 CECARRYIN_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe14]
set_property port_width 1 [get_debug_ports u_ila_0/probe14]
connect_debug_port u_ila_0/probe14 [get_nets [list CED_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe15]
set_property port_width 1 [get_debug_ports u_ila_0/probe15]
connect_debug_port u_ila_0/probe15 [get_nets [list CEM_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe16]
set_property port_width 1 [get_debug_ports u_ila_0/probe16]
connect_debug_port u_ila_0/probe16 [get_nets [list CEOPMODE_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe17]
set_property port_width 1 [get_debug_ports u_ila_0/probe17]
connect_debug_port u_ila_0/probe17 [get_nets [list CEP_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe18]
set_property port_width 1 [get_debug_ports u_ila_0/probe18]
connect_debug_port u_ila_0/probe18 [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/probe19]
set_property port_width 1 [get_debug_ports u_ila_0/probe19]
connect_debug_port u_ila_0/probe19 [get_nets [list RSTA_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe20]
set_property port_width 1 [get_debug_ports u_ila_0/probe20]
connect_debug_port u_ila_0/probe20 [get_nets [list RSTA_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA AND TRIGGER [get_debug_ports u_ila_0/probe21]
set_property port_width 1 [get_debug_ports u_ila_0/probe21]
connect_debug_port u_ila_0/probe21 [get_nets [list RSTC_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe22]
set_property port_width 1 [get_debug_ports u_ila_0/probe22]
connect_debug_port u_ila_0/probe22 [get_nets [list RSTCARRYIN_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe23]
set_property port_width 1 [get_debug_ports u_ila_0/probe23]
connect_debug_port u_ila_0/probe23 [get_nets [list RSTD_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE TYPE DATA AND TRIGGER [get debug ports u ila 0/probe24]
set_property port_width 1 [get_debug_ports u_ila_0/probe24]
connect_debug_port u_ila_0/probe24 [get_nets [list RSTM_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe25]
set_property port_width 1 [get_debug_ports u_ila_0/probe25]
connect_debug_port u_ila_0/probe25 [get_nets [list RSTOPMODE_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe26]
set_property port_width 1 [get_debug_ports u_ila_0/probe26]
connect_debug_port u_ila_0/probe26 [get_nets [list RSTP_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]
```

Elaboration

Messages

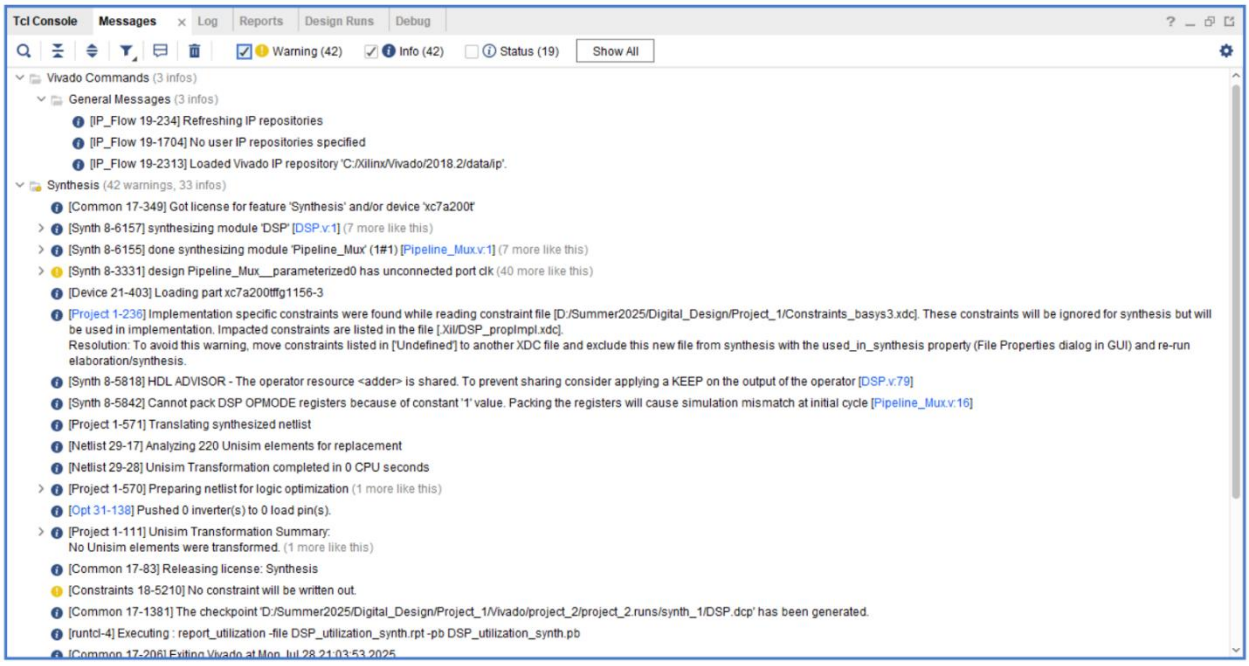


Schematic

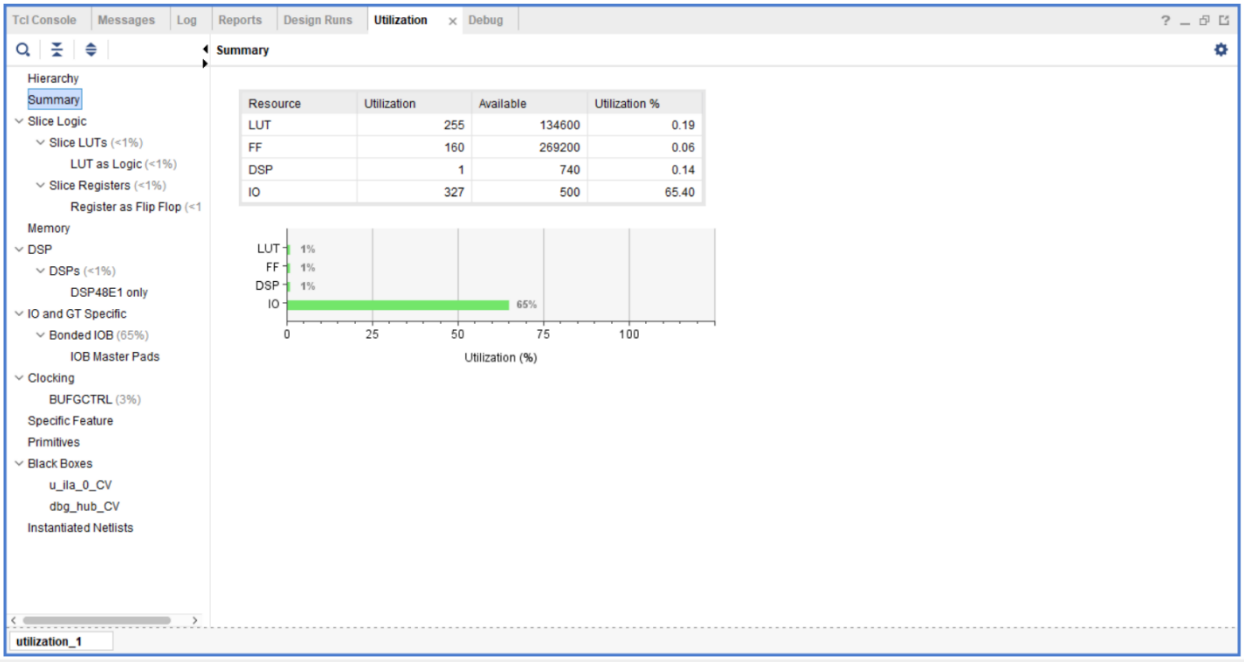


Synthesis

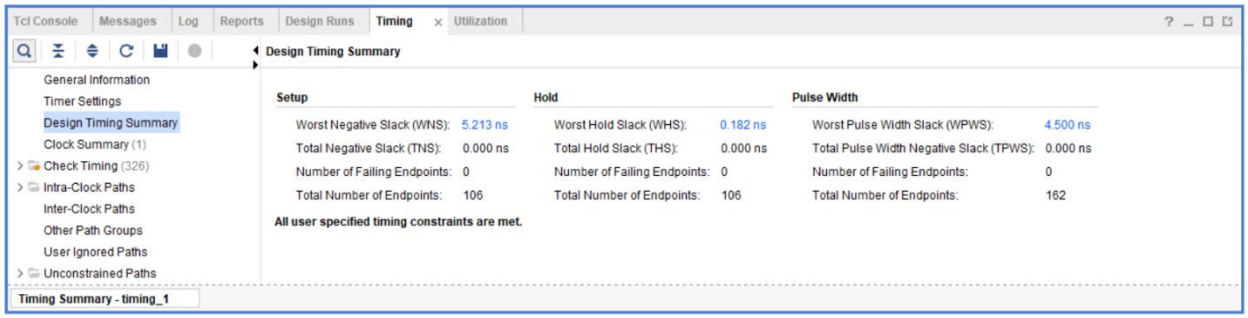
Messages



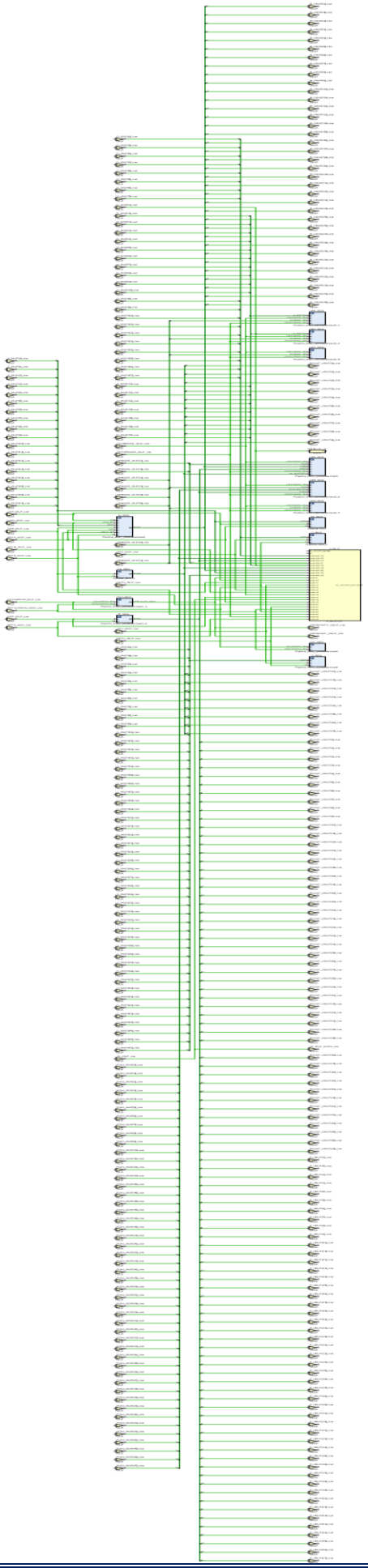
Utilization Report



Timing Report

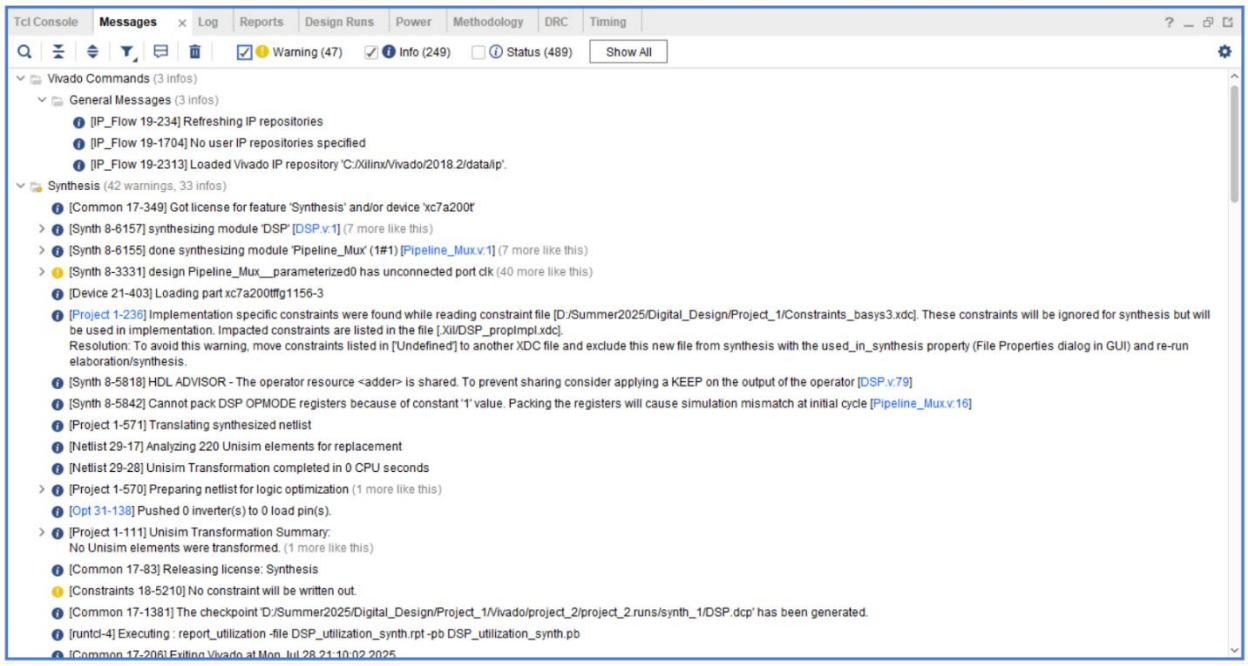


Schematic

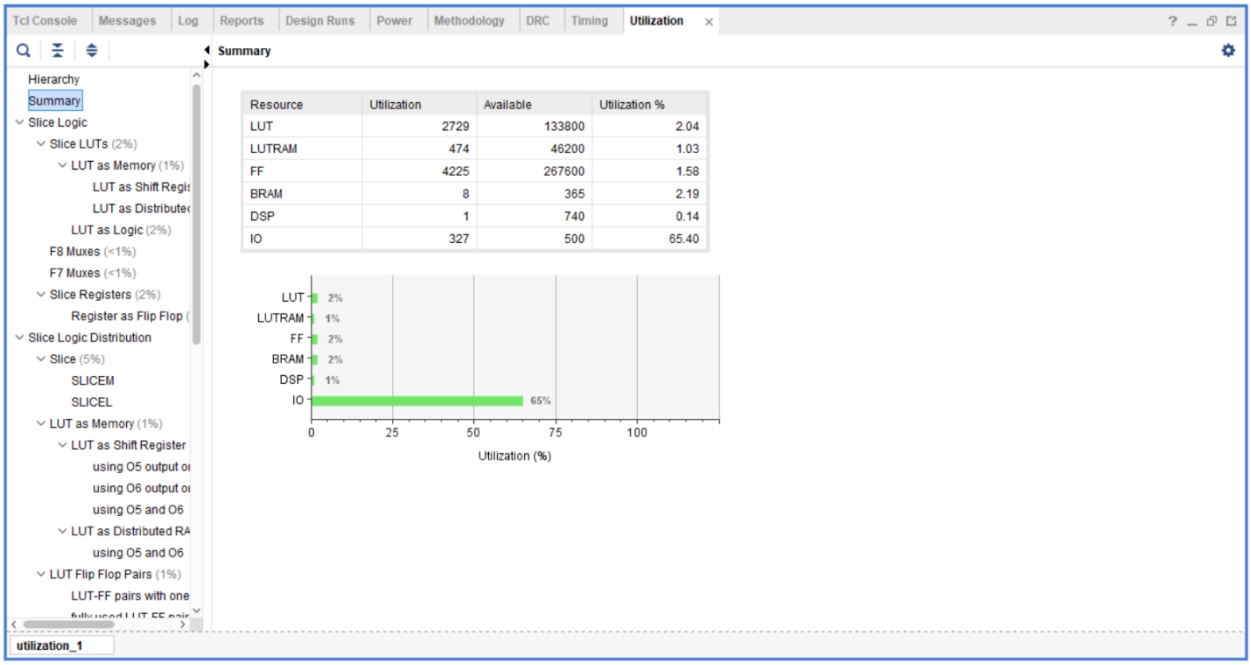


Implementation

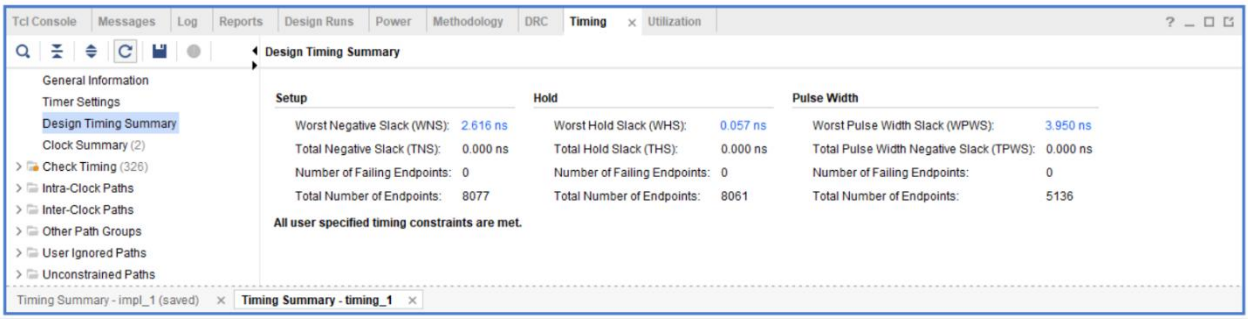
Messages



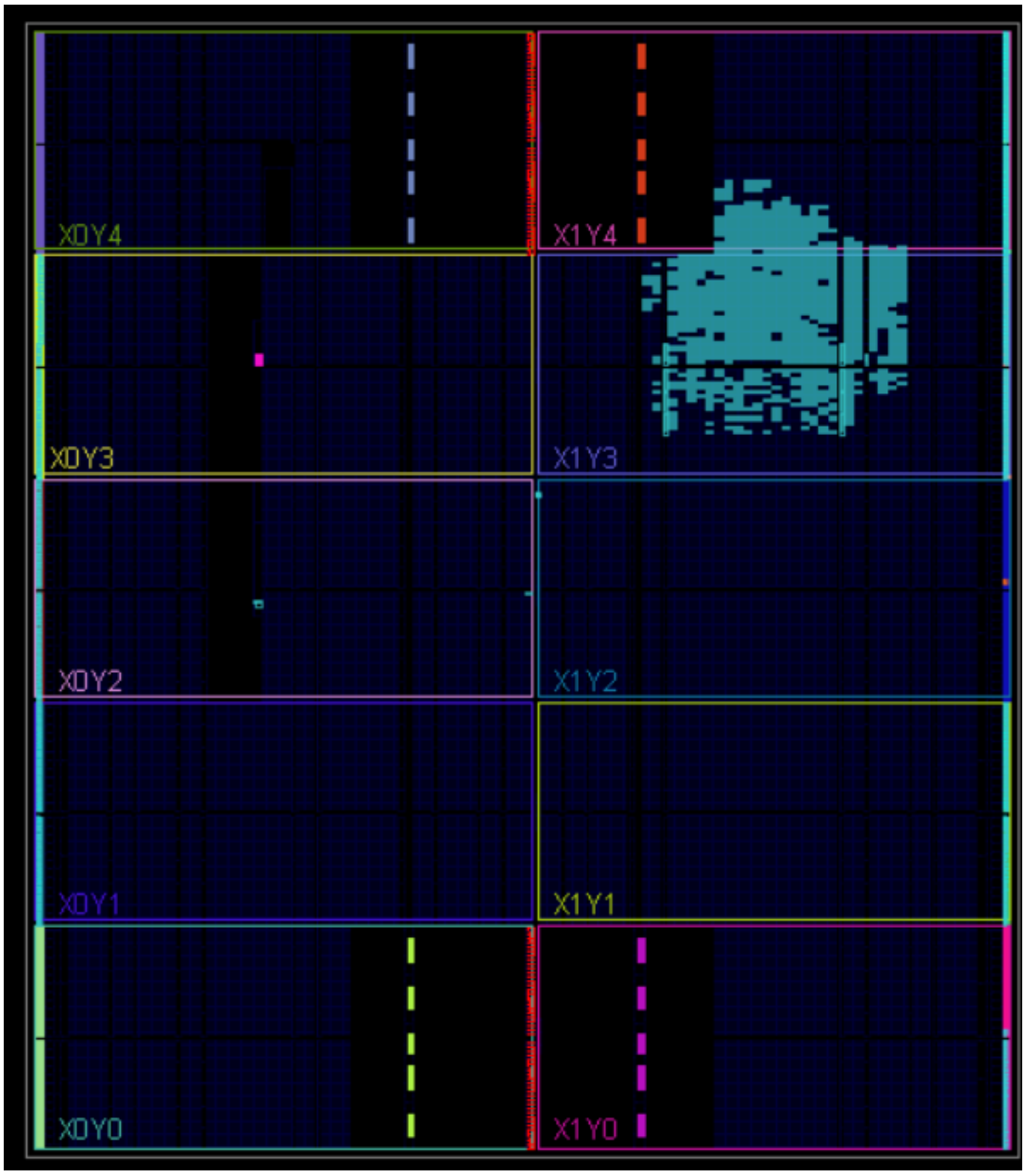
Utilization Report



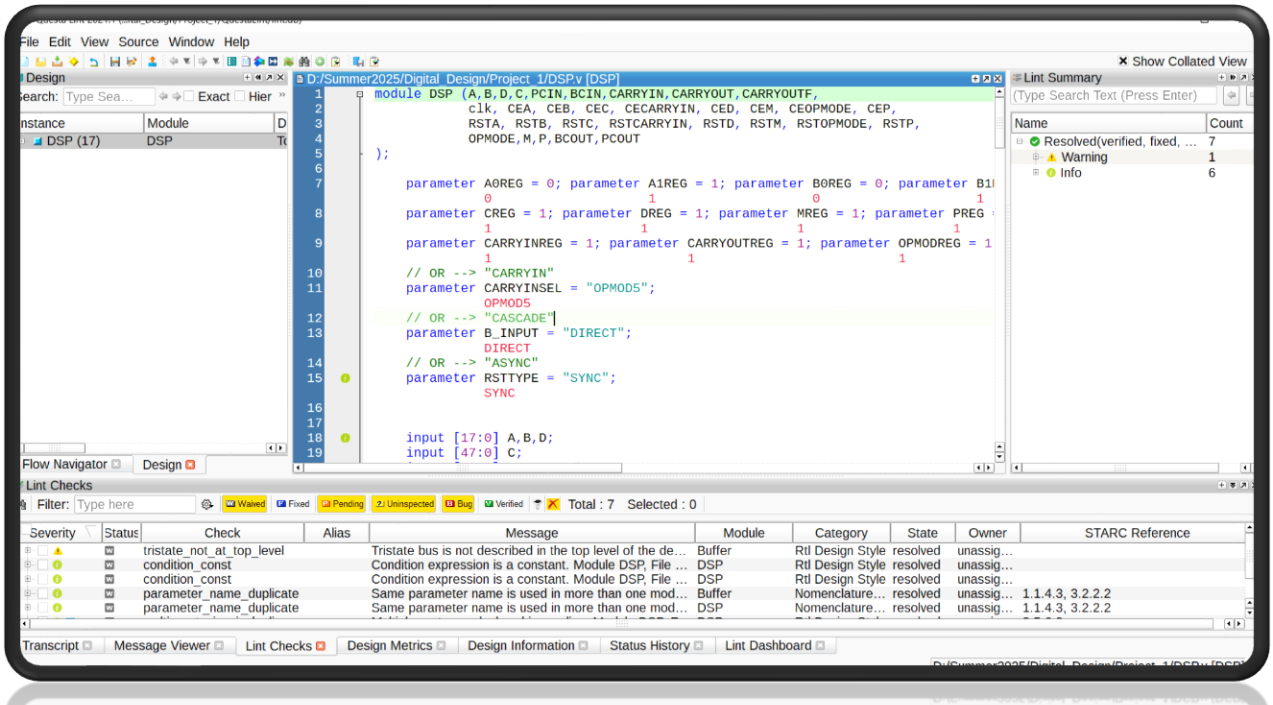
Timing Report



Device



Linting


File Edit View Source Window Help
Show Collated View
Design
Search: Type Sea... Exact Hier
Instance Module
DSP (17) DSP
D:/Summer2025/Digital_Design/Project_1/DSP.v [DSP]
module DSP (A,B,D,C,PCIN,BCIN,CARRYIN,CARRYOUT,CARRYOUTF,
clk, CEA, CEB, CEC, CECARRYIN, CED, CEM, CEOPMODE, CEP,
RSTA, RSTB, RSTC, RSTCARRYIN, RSTD, RSTM, RSTOPMODE, RSTP,
OPMODE,M,P,BCOUT,PCOUT
);
parameter A0REG = 0; parameter A1REG = 1; parameter B0REG = 0; parameter B1
parameter CREG = 1; parameter DREG = 1; parameter MREG = 1; parameter PREG
parameter CARRYINREG = 1; parameter CARRYOUTREG = 1; parameter OPMODREG = 1
// OR --> "CARRYIN"
parameter CARRYINSEL = "OPMOD5";
// OR --> "CASCADE"
parameter B_INPUT = "DIRECT";
// OR --> "ASYNC"
parameter RSTTYPE = "SYNC";
input [17:0] A,B,D;
input [47:0] C;
Lint Summary
(Type Search Text (Press Enter))
Name Count
Resolved(verified, fixed, ... 7
Warning 1
Info 6
Flow Navigator Design
Lint Checks
Filter: Type here
Waived Fixed Pending Uninspected Bug Verified
Total : 7 Selected : 0
Severity Status Check Alias Message Module Category State Owner STARC Reference
tristate_not_at_top_level Tristate bus is not described in the top level of the de... Buffer Rtl Design Style resolved unassig...
condition_const Condition expression is a constant. Module DSP, File ... DSP Rtl Design Style resolved unassig...
condition_const Condition expression is a constant. Module DSP, File ... DSP Rtl Design Style resolved unassig...
parameter_name_duplicate Same parameter name is used in more than one mod... Buffer Nomenclature... resolved unassig... 1.1.4.3, 3.2.2.2
parameter_name_duplicate Same parameter name is used in more than one mod... DSP Nomenclature... resolved unassig... 1.1.4.3, 3.2.2.2
Transcript Message Viewer Lint Checks Design Metrics Design Information Status History Lint Dashboard