1-RTL code:

```
module REG_MUX_pair(clk,rst,inp,CE,out);
parameter RESETTYPE="SYNC";
parameter SELECT=0;
parameter WIDTH=18;
input clk,rst,CE;
input [WIDTH-1:0] inp;
output  [WIDTH-1:0] out;
reg [WIDTH-1:0] out_comb;
reg [WIDTH-1:0] out_reg;
assign out = out_comb;
generate
    if(RESETTYPE=="ASYNC") begin
      always @(posedge clk or posedge rst) begin
        if(rst)
           out_reg<=0;
        else begin
            if(CE)
              out_reg<=inp;
        end
      end
    end
    else begin
      always @(posedge clk) begin
        if(rst)
           out_reg<=0;
        else begin
            if(CE)
              out_reg<=inp;
        end
        end
    end
endgenerate
always @(*) begin
    case (SELECT)
        0: out_comb=inp;
        default: out_comb=out_reg;
    endcase
end
endmodule
```

Figure 1: REG with MUX that will be instantiated in the DSP code

```
module DSP48A1 (A,B,C,D,OPMODE,BCIN,CEA,CEB,CEC,CED,CEM,CEP,CECARRYIN,CEOPMODE,PCIN,CLK,RSTA,RSTB,RSTC,RSTD,RSTM,RSTP,RSTCARRYIN,RSTOPMODE,
CARRYIN,M, P, CARRYOUT, CARRYOUTF, PCOUT, BCOUT);
input CEA,CEB,CEC,CED,CEM,CEP,CECARRYIN,CEOPMODE;
input CLK,RSTA,RSTB,RSTC,RSTD,RSTM,RSTP,RSTCARRYIN,RSTOPMODE;
input CARRYIN;
input [17:0] A,B,D,BCIN;
input [47:0] C,PCIN;
input [7:0] OPMODE;
output reg CARRYOUT,CARRYOUTF;
output reg [47:0] P,PCOUT;
output reg [35:0] M;
output reg [17:0] BCOUT;
wire [17:0] D_REG_OUT,B0_REG_OUT,A0_REG_OUT,B1_REG_OUT,A1_REG_OUT;
wire [47:0] C_REG_OUT,P_REG_OUT;
wire [35:0] M_REG_OUT;
wire [7:0]  OPMODE_REG_OUT;
wire CYI_OUT, CYO_OUT;
reg [17:0] B0_REG_inp,PRE_OUT,B1_reg_inp;
reg [47:0] POST_OUT,Z_OUT,X_OUT;
reg [35:0] multipler_out;
reg CYI_inp;
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 OPMODEREG = 1;
parameter CARRYINSEL = "OPMODE5";
parameter B_INPUT = "DIRECT";
parameter RSTTYPE = "SYNC";
REG_MUX_pair #(.WIDTH(8),.SELECT(OPMODEREG))  OPMODE_REG(CLK,RSTOPMODE,OPMODE,CEOPMODE,OPMODE_REG_OUT);
REG_MUX_pair #(.WIDTH(18),.SELECT(DREG))     D_REG(CLK,RSTD,D,CED,D_REG_OUT);
REG_MUX_pair #(.WIDTH(18),.SELECT(B0REG))    B0_REG(CLK,RSTB,B0_REG_inp,CEB,B0_REG_OUT);
REG_MUX_pair #(.WIDTH(18),.SELECT(A0REG))    A0_REG(CLK,RSTA,A,CEA,A0_REG_OUT);
REG_MUX_pair #(.WIDTH(48),.SELECT(CREG))     C_REG(CLK,RSTC,C,CEC,C_REG_OUT);
REG_MUX_pair #(.WIDTH(18),.SELECT(B1REG))    B1_REG(CLK,RSTB,B1_reg_inp,CEB,B1_REG_OUT);
REG_MUX_pair #(.WIDTH(18),.SELECT(A1REG))    A1_REG(CLK,RSTA,A0_REG_OUT,CEA,A1_REG_OUT);
REG_MUX_pair #(.WIDTH(36),.SELECT(MREG))     M_REG(CLK,RSTM,multipler_out,CEM,M_REG_OUT);
REG_MUX_pair #(.WIDTH(1),.SELECT(CARRYINREG)) CYI(CLK,RSTCARRYIN,CYI_inp,CECARRYIN,CYI_OUT);
REG_MUX_pair #(.WIDTH(1),.SELECT(CARRYOUTREG)) CYO(CLK,RSTCARRYIN,POST_OUT[47],CECARRYIN,CYO_OUT);
REG_MUX_pair #(.WIDTH(48),.SELECT(PREG))       P_REG(CLK,RSTP,POST_OUT,CEP,P_REG_OUT);
always @(*) begin
    case (B_INPUT)
        "DIRECT" : B0_REG_inp = B;
        "CASCADE": B0_REG_inp = BCIN;
        default  : B0_REG_inp = 0;
    endcase
    if (OPMODE_REG_OUT[6])
        PRE_OUT = D_REG_OUT - B0_REG_OUT;
    else
        PRE_OUT = D_REG_OUT + B0_REG_OUT;
    case (OPMODE_REG_OUT[4])
        0       : B1_reg_inp = B0_REG_OUT;
        default : B1_reg_inp = PRE_OUT;
    endcase
    BCOUT = B1_REG_OUT;
    multipler_out = B1_REG_OUT * A1_REG_OUT;
    // X input selection
    case (OPMODE_REG_OUT[1:0])
        2'b00   : X_OUT = 0;
        2'b01   : X_OUT = {12'b0, M_REG_OUT}; // zero-extend 36 bits to 48
        2'b10   : X_OUT = P;
        default : X_OUT = {D_REG_OUT[11:0], A0_REG_OUT, B0_REG_OUT}; // build custom 48-bit value
    endcase
    // Z input selection
    case (OPMODE_REG_OUT[3:2])
        2'b00   : Z_OUT = 0;
        2'b01   : Z_OUT = PCIN;
        2'b10   : Z_OUT = P;
        default : Z_OUT = C_REG_OUT;
    endcase
    if (OPMODE_REG_OUT[7])
        POST_OUT = Z_OUT - (X_OUT + CYI_OUT);
    else
        POST_OUT = Z_OUT + X_OUT + CYI_OUT;
    case (CARRYINSEL)
        "OPMODE5": CYI_inp = OPMODE_REG_OUT[5];
        "CARRYIN": CYI_inp = CARRYIN;
        default  : CYI_inp = 0;
    endcase
    PCOUT       = P_REG_OUT;
    P           = P_REG_OUT;
    M           = M_REG_OUT;
    CARRYOUT    = CYO_OUT;
    CARRYOUTF   = CYO_OUT;
end
endmodule
```

Figure 2: DSP RTL code

2-Testbench code:

```
module DSP48A1_tb();
reg [17:0] A,B,D,BCIN;
reg [47:0] C,PCIN;
reg [7:0] OPMODE;
reg CLK,CEA,CEB,CEC,CED,CEM,CEP,CECARRYIN,CEOPMODE,RSTA,RSTB,RSTC,RSTD,RSTM,RSTP,RSTCARRYIN,RSTOPMODE,CARRYIN;
wire [47:0] PCOUT_DUT,P_DUT;
wire [17:0] BCOUT_DUT;
wire [35:0] M_DUT;
wire CARRYOUT_DUT,CARRYOUTF_DUT;
reg [47:0] PCOUT_expected,P_expected;
reg [17:0] BCOUT_expected;
reg [35:0] M_expected;
reg CARRYOUT_expected,CARRYOUTF_expected;
DSP48A1 DUT(A,B,C,D,OPMODE,BCIN,CEA,CEB,CEC,CED,CEM,CEP,CECARRYIN,CEOPMODE,PCIN,CLK,RSTA,RSTB,RSTC,RSTD,RSTM,RSTP,RSTCARRYIN,RSTOPMODE,CARRYIN,M_DUT,P_DUT,CARRYOUT_DUT,CARRYOUTF_DUT,PCOUT_DUT,BCOUT_DUT);
initial begin
    CLK=0;
    forever
    #1 CLK=~CLK;
end
initial begin
    // verify reset operation
    RSTA=1;RSTB=1;RSTC=1;RSTD=1;RSTM=1;RSTP=1;RSTCARRYIN=1;RSTOPMODE=1;
    A=$random;B=$random;D=$random;BCIN=$random;
    C=$random;PCIN=$random;OPMODE=$random;CEA=$random;CEB=$random;CEC=$random;
    CED=$random;CEM=$random;CEP=$random;CECARRYIN=$random;CEOPMODE=$random;CARRYIN=$random;
    @(negedge CLK);
    PCOUT_expected=0;P_expected=0;BCOUT_expected=0;
    M_expected=0;CARRYOUT_expected=0;CARRYOUTF_expected=0;
    if (PCOUT_expected!=PCOUT_DUT) begin
      $display("Error in rst operation");
      $stop;
    end
    if (P_expected!=P_DUT) begin
      $display("Error in rst operation");
      $stop;
    end
    if (CARRYOUTF_expected!=CARRYOUTF_DUT) begin
      $display("Error in rst operation");
      $stop;
    end
    if (BCOUT_expected!=BCOUT_DUT) begin
      $display("Error in rst operation");
      $stop;
    end
    if (M_expected!=M_DUT) begin
      $display("Error in rst operation");
      $stop;
    end
    if (CARRYOUT_expected!=CARRYOUT_DUT) begin
      $display("Error in rst operation");
      $stop;
    end
    RSTA=0;RSTB=0;RSTC=0;RSTD=0;RSTM=0;RSTP=0;RSTCARRYIN=0;RSTOPMODE=0;
    CEA=1;CEB=1;CEC=1;CED=1;CEM=1;CEP=1;CECARRYIN=1;CEOPMODE=1;
    // verify DSP path1
    OPMODE=8'b11011101;
    A=20;B=10;C=350;D=25;
    PCIN=$random;BCIN=$random;CARRYIN=$random;
    repeat (4) @(negedge CLK);
    PCOUT_expected=48'h32;P_expected=48'h32;BCOUT_expected=18'hf;
    M_expected=36'h12c;CARRYOUT_expected=0;CARRYOUTF_expected=0;
    if (PCOUT_expected!=PCOUT_DUT) begin
      $display("Error in DSP path 1");
      $stop;
    end
    if (P_expected!=P_DUT) begin
      $display("Error in DSP path 1");
      $stop;
    end
    if (CARRYOUTF_expected!=CARRYOUTF_DUT) begin
      $display("Error in DSP path 1");
      $stop;
    end
    if (BCOUT_expected!=BCOUT_DUT) begin
      $display("Error in DSP path 1");
      $stop;
    end
    if (M_expected!=M_DUT) begin
      $display("Error in DSP path 1");
      $stop;
    end
    if (CARRYOUT_expected!=CARRYOUT_DUT) begin
      $display("Error in DSP path 1");
      $stop;
    end
```

Figure 3: Testbench code part 1

```
  // verify DSP path2
    OPMODE=8'b00010000;
    A=20;B=10;C=350;D=25;
    PCIN=$random;BCIN=$random;CARRYIN=$random;
    repeat (3) @(negedge CLK);
    PCOUT_expected=0;P_expected=0;BCOUT_expected=18'h23;
    M_expected=36'h2bc;CARRYOUT_expected=0;CARRYOUTF_expected=0;
    if (PCOUT_expected!=PCOUT_DUT) begin
      $display("Error in DSP path 2");
      $stop;
    end
    if (P_expected!=P_DUT) begin
      $display("Error in DSP path 2");
      $stop;
    end
    if (CARRYOUTF_expected!=CARRYOUTF_DUT) begin
      $display("Error in DSP path 2");
      $stop;
    end
    if (BCOUT_expected!=BCOUT_DUT) begin
      $display("Error in DSP path 2");
      $stop;
    end
    if (M_expected!=M_DUT) begin
      $display("Error in DSP path 2");
      $stop;
    end
    if (CARRYOUT_expected!=CARRYOUT_DUT) begin
      $display("Error in DSP path 2");
      $stop;
    end
    // verify DSP path3
    OPMODE=8'b00001010;
    A=20;B=10;C=350;D=25;
    PCIN=$random;BCIN=$random;CARRYIN=$random;
    repeat (3) @(negedge CLK);
    PCOUT_expected=0;P_expected=0;BCOUT_expected=18'ha;
    M_expected=36'hc8;CARRYOUT_expected=0;CARRYOUTF_expected=0;
    if (PCOUT_expected!=PCOUT_DUT) begin
      $display("Error in DSP path 3");
      $stop;
    end
    if (P_expected!=P_DUT) begin
      $display("Error in DSP path 3");
      $stop;
    end
    if (CARRYOUTF_expected!=CARRYOUTF_DUT) begin
      $display("Error in DSP path 3");
      $stop;
    end
    if (BCOUT_expected!=BCOUT_DUT) begin
      $display("Error in DSP path 3");
      $stop;
    end
    if (M_expected!=M_DUT) begin
      $display("Error in DSP path 3");
      $stop;
    end
    if (CARRYOUT_expected!=CARRYOUT_DUT) begin
      $display("Error in DSP path 3");
      $stop;
    end
    // verify DSP path4
    OPMODE=8'b10100111;
    A=5;B=6;C=350;D=25;
    PCIN=3000;BCIN=$random;CARRYIN=$random;
    repeat (3) @(negedge CLK);
    PCOUT_expected=48'hfe6fffec0bb1;P_expected=48'hfe6fffec0bb1;BCOUT_expected=18'h6;
    M_expected=36'h1e;CARRYOUT_expected=1;CARRYOUTF_expected=1;
    if (PCOUT_expected!=PCOUT_DUT) begin
      $display("Error in DSP path 4");
      $stop;
    end
    if (P_expected!=P_DUT) begin
      $display("Error in DSP path 4");
      $stop;
    end
    if (CARRYOUTF_expected!=CARRYOUTF_DUT) begin
      $display("Error in DSP path 4");
      $stop;
    end
    if (BCOUT_expected!=BCOUT_DUT) begin
      $display("Error in DSP path 4");
      $stop;
    end
    if (M_expected!=M_DUT) begin
      $display("Error in DSP path 4");
      $stop;
    end
    if (CARRYOUT_expected!=CARRYOUT_DUT) begin
      $display("Error in DSP path 4");
      $stop;
    end
    $display("function is correct");
    $stop;
  end
  endmodule
```

Figure 4: Testbench code part 2

3-Do file:

```
1   vlib work
2   vlog REG_MUX_pair.v DSP.v DSP_tb.v
3   vsim -voptargs=+acc work.DSP48A1_tb
4   add wave *
5   run -all
6   #quit -sim
```

Figure 5: DO file code

4-QuestaSim snippets:

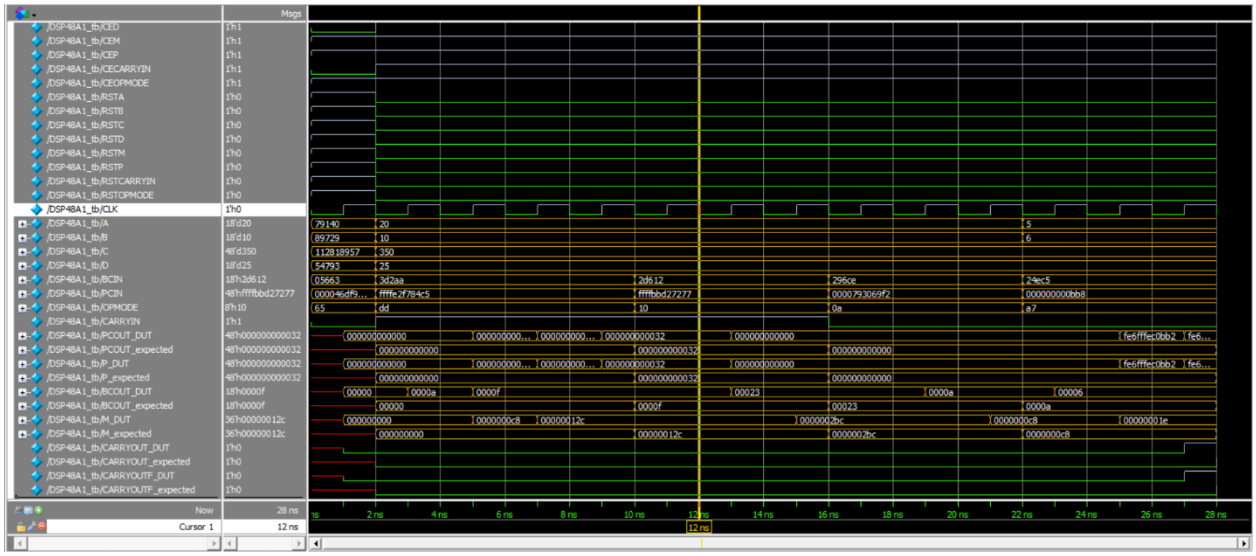


Figure 6: Simulation Waveform showing that all DUT outputs equal to expected outputs

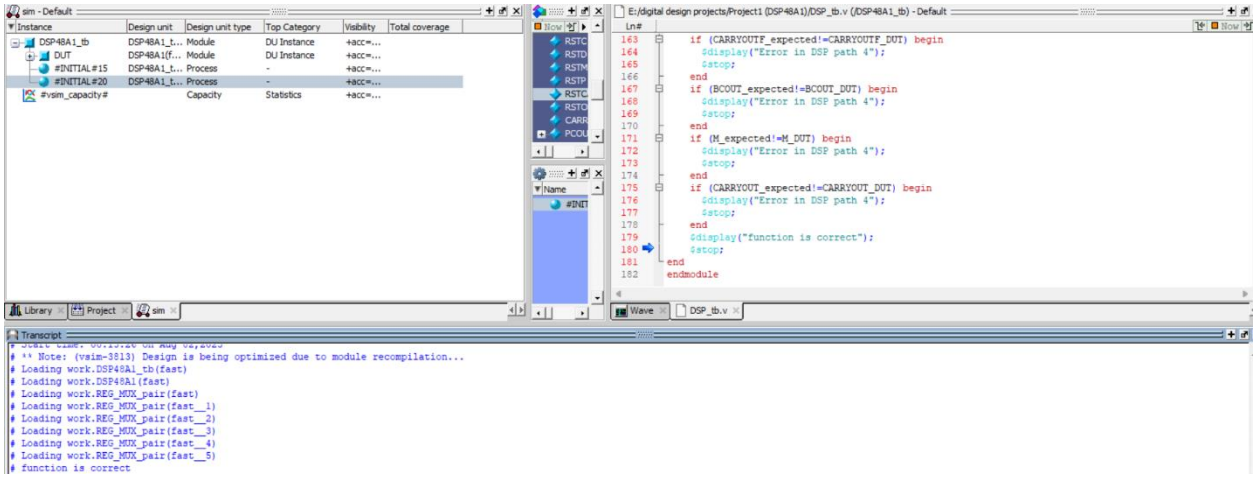


Figure 7: Transcript showing display message that function is correct

5-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 -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports CLK]


## Switches
#set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {sw[0]}]
#set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {sw[1]}]
#set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports {sw[2]}]
#set_property -dict { PACKAGE_PIN W17   IOSTANDARD LVCMOS33 } [get_ports {sw[3]}]
#set_property -dict { PACKAGE_PIN W15   IOSTANDARD LVCMOS33 } [get_ports {sw[4]}]
#set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports {sw[5]}]
#set_property -dict { PACKAGE_PIN W14   IOSTANDARD LVCMOS33 } [get_ports {sw[6]}]
#set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports {sw[7]}]
#set_property -dict { PACKAGE_PIN V2    IOSTANDARD LVCMOS33 } [get_ports {sw[8]}]
#set_property -dict { PACKAGE_PIN T3    IOSTANDARD LVCMOS33 } [get_ports {sw[9]}]
#set_property -dict { PACKAGE_PIN T2    IOSTANDARD LVCMOS33 } [get_ports {sw[10]}]
#set_property -dict { PACKAGE_PIN R3    IOSTANDARD LVCMOS33 } [get_ports {sw[11]}]
#set_property -dict { PACKAGE_PIN W2    IOSTANDARD LVCMOS33 } [get_ports {sw[12]}]
#set_property -dict { PACKAGE_PIN U1    IOSTANDARD LVCMOS33 } [get_ports {sw[13]}]
#set_property -dict { PACKAGE_PIN T1    IOSTANDARD LVCMOS33 } [get_ports {sw[14]}]
#set_property -dict { PACKAGE_PIN R2    IOSTANDARD LVCMOS33 } [get_ports {sw[15]}]
```

Figure 8: Constraint File code

6-Elaboration:

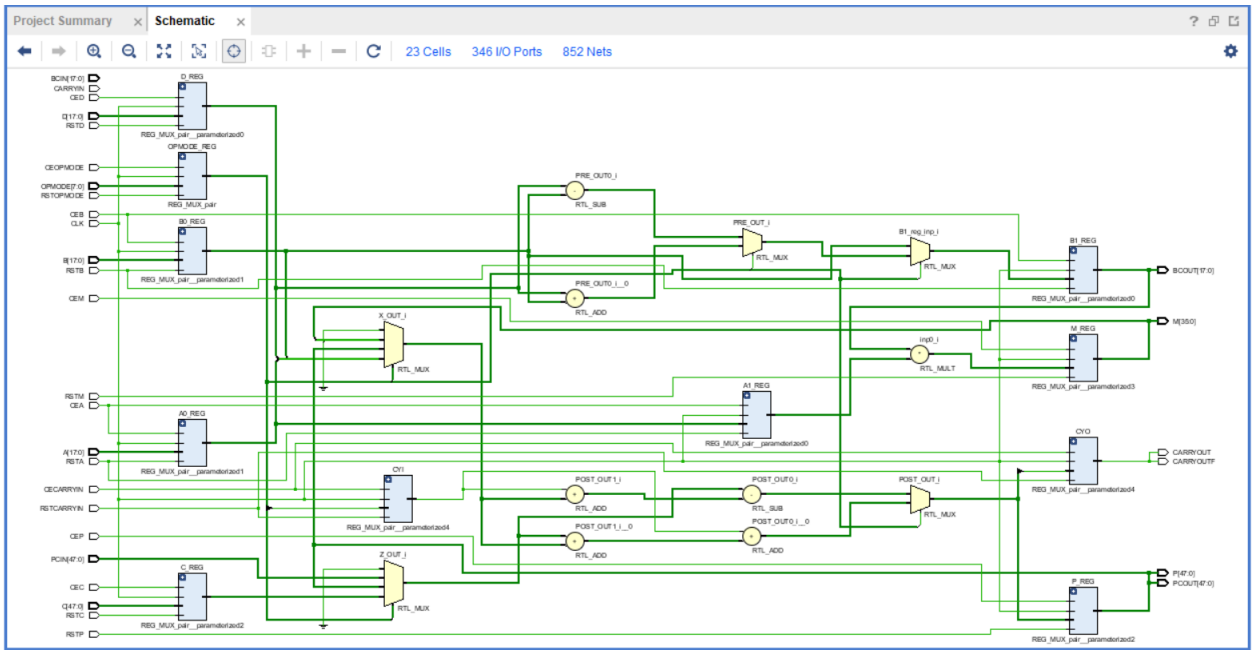


Figure 9: Elaboration schematic

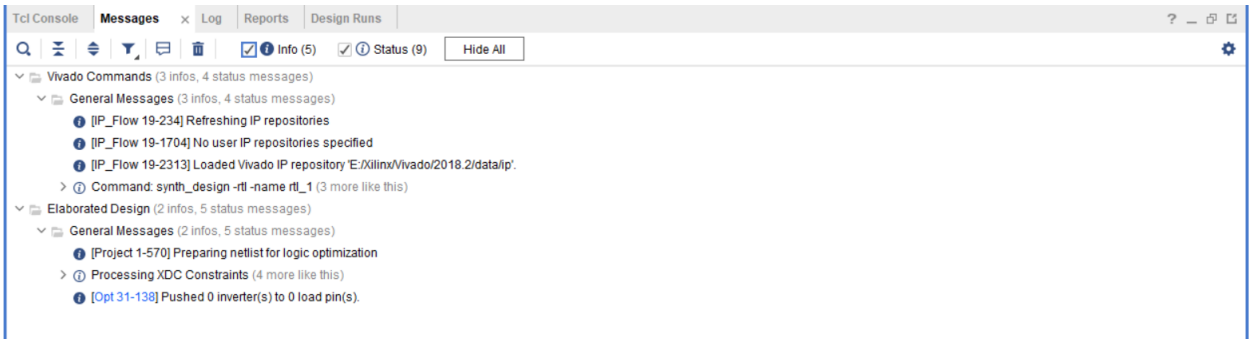


Figure 10: messages tab after Elaboration showing no Error

7-Synthesis:

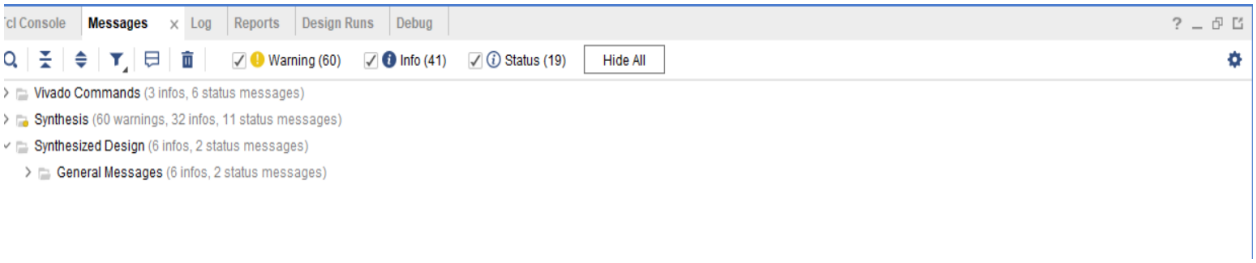


Figure 11:messages tab after Synthesis showing no Error

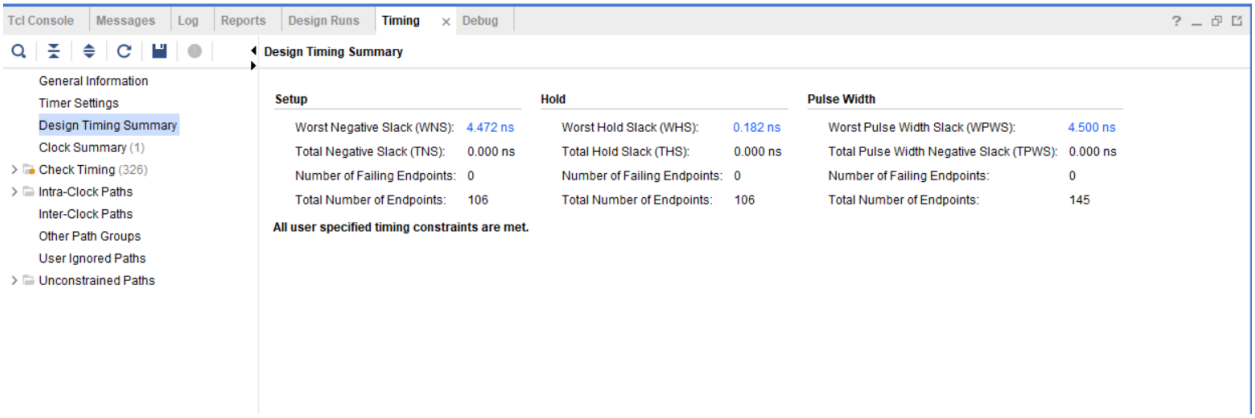


Figure 12: Timing report after synthesis run

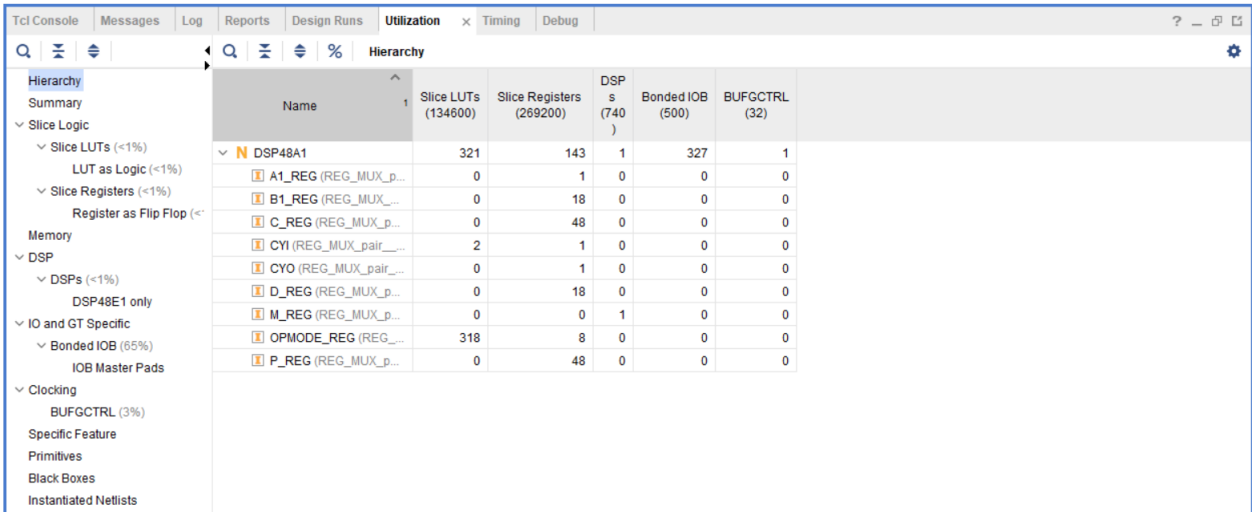


Figure 13:utilization report after synthesis run

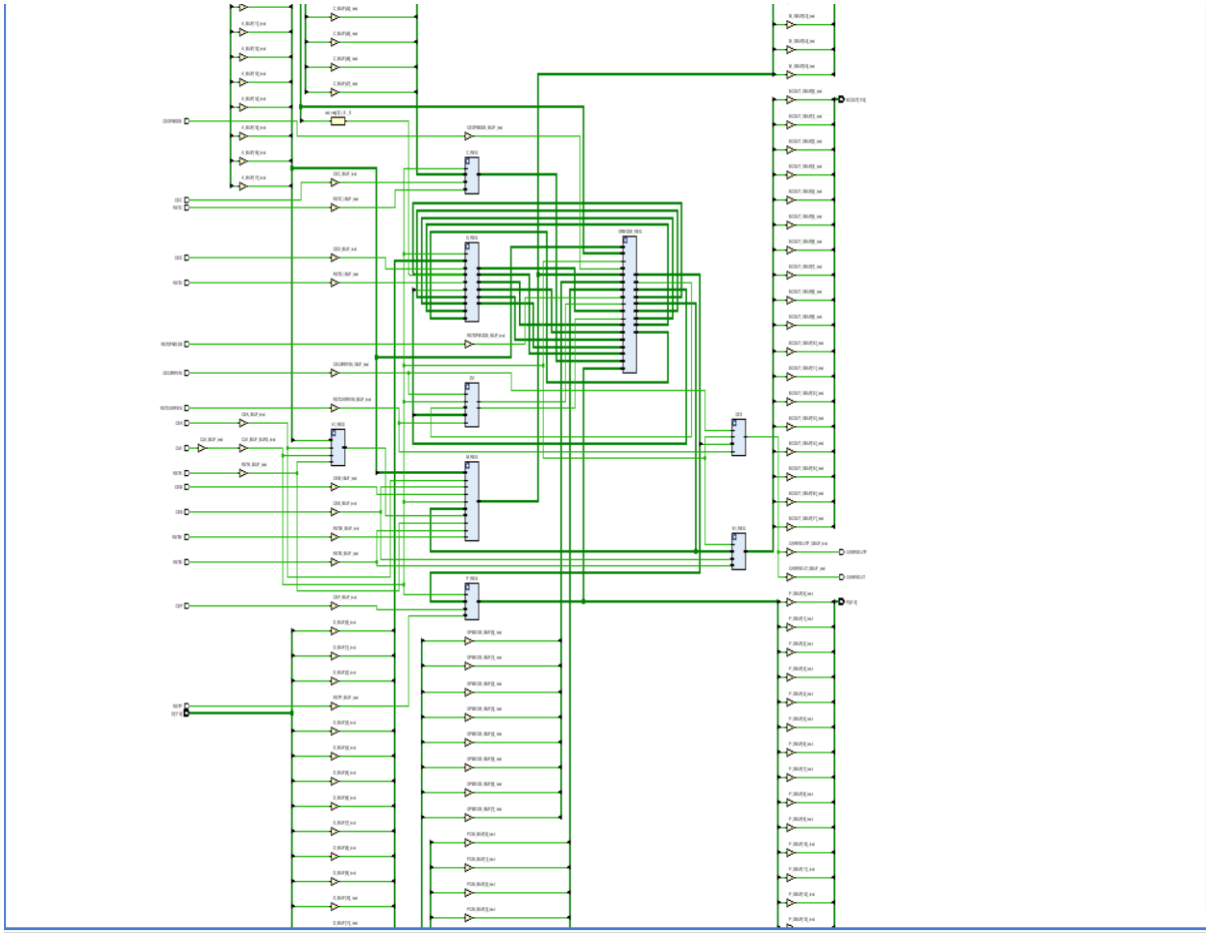


Figure 14: synthesis schematic

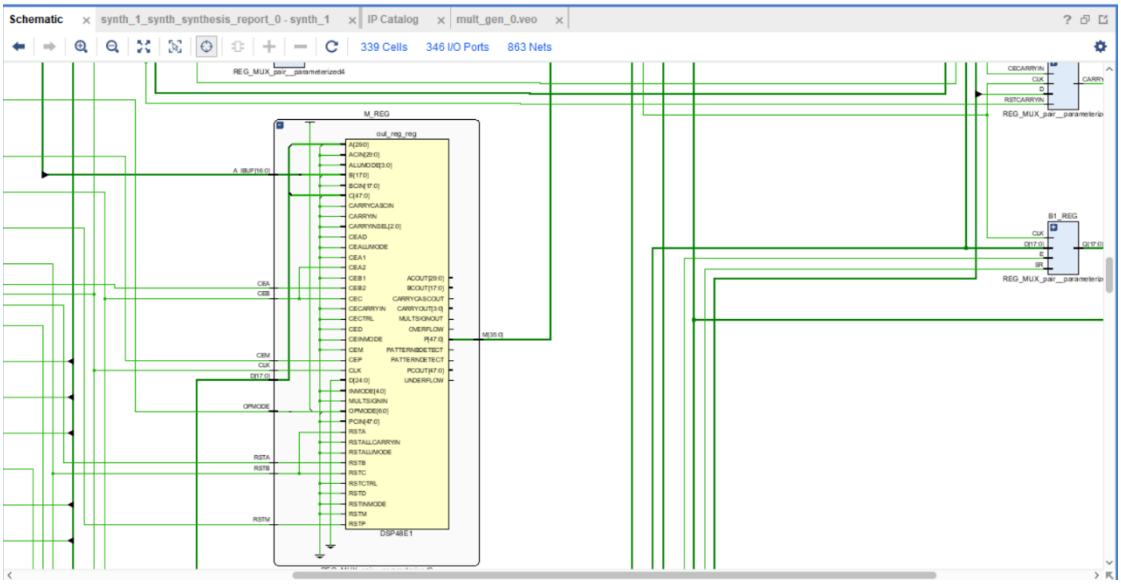


Figure 15: more obvious photo for synthesis schematic Showing DSP block at multiplier out

8-Implementation:

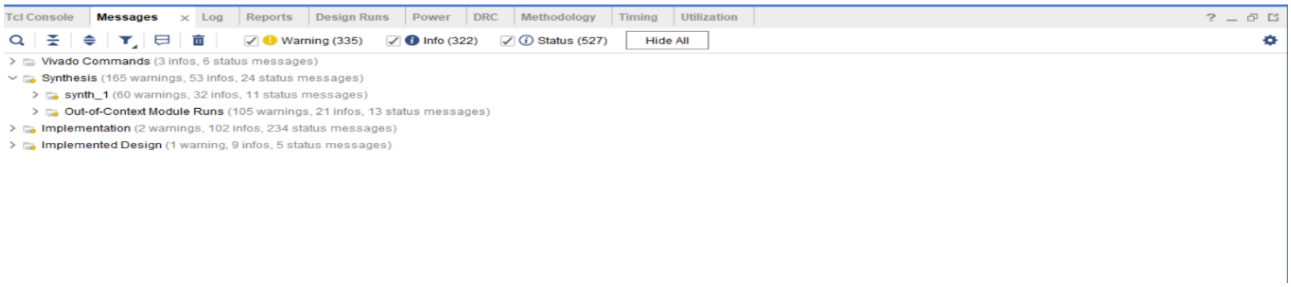


Figure 16: messages tab after implementation showing no Error

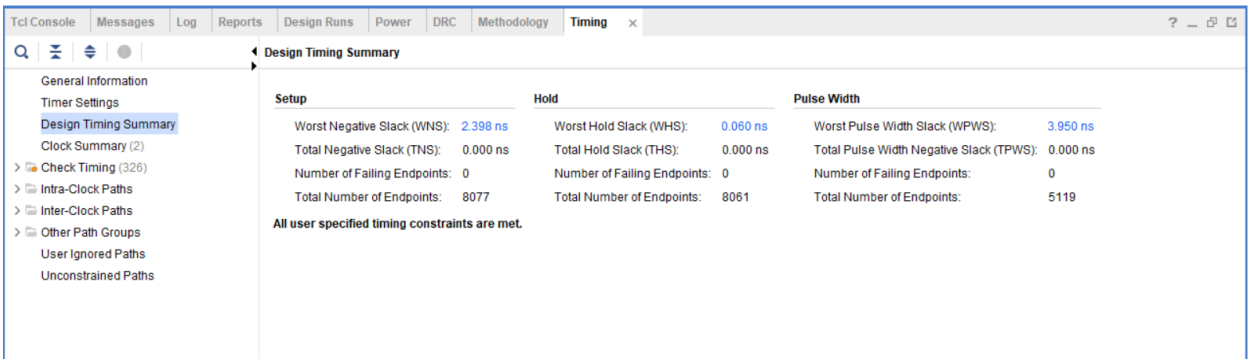
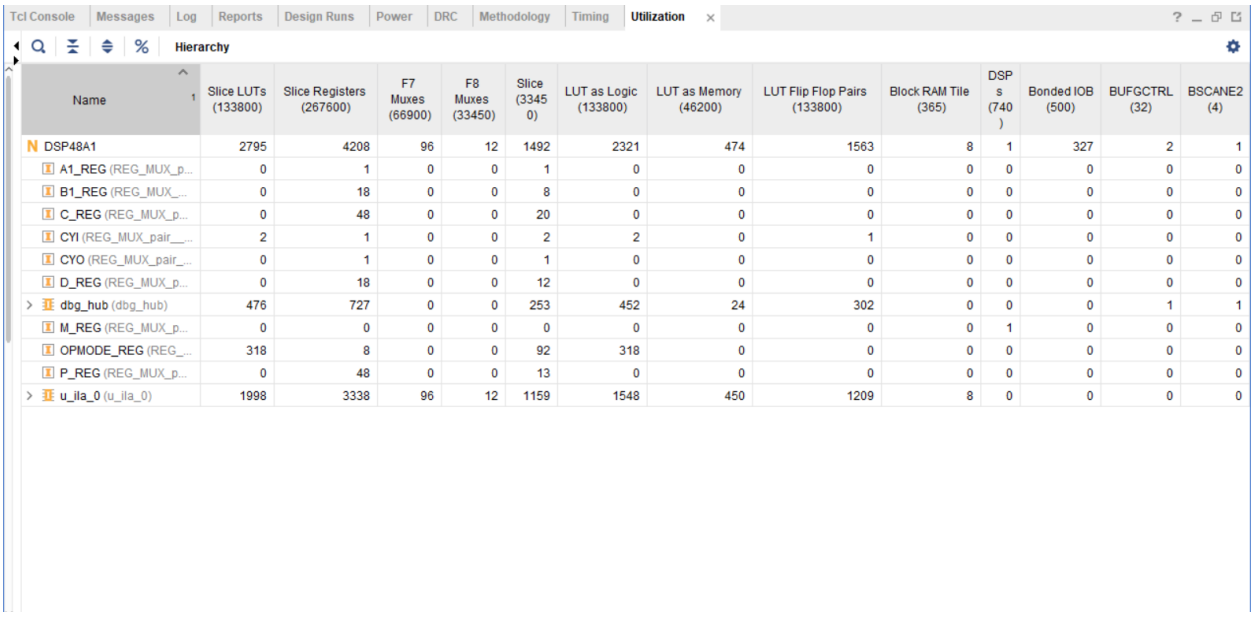


Figure 17: Timing report after implementation run



Name	Slice LUTs (133800)	Slice Registers (267600)	F7 Muxes (66900)	F8 Muxes (33450)	Slice (33450)	LUT as Logic (133800)	LUT as Memory (46200)	LUT Flip Flop Pairs (133800)	Block RAM Tile (365)	DSPs (740)	Bonded IOB (500)	BUFGCTRL (32)	BSCANE2 (4)
DSP48A1	2795	4208	96	12	1492	2321	474	1563	8	1	327	2	1
A1_REG (REG_MUX_p...	0	1	0	0	1	0	0	0	0	0	0	0	0
B1_REG (REG_MUX_...	0	18	0	0	8	0	0	0	0	0	0	0	0
C_REG (REG_MUX_p...	0	48	0	0	20	0	0	0	0	0	0	0	0
CYI (REG_MUX_pair_...	2	1	0	0	2	2	0	1	0	0	0	0	0
CYO (REG_MUX_pair_...	0	1	0	0	1	0	0	0	0	0	0	0	0
D_REG (REG_MUX_p...	0	18	0	0	12	0	0	0	0	0	0	0	0
dbg_hub (dbg_hub)	476	727	0	0	253	452	24	302	0	0	0	1	1
M_REG (REG_MUX_p...	0	0	0	0	0	0	0	0	0	1	0	0	0
OPMODE_REG (REG_...	318	8	0	0	92	318	0	0	0	0	0	0	0
P_REG (REG_MUX_p...	0	48	0	0	13	0	0	0	0	0	0	0	0
u_ila_0 (u_ila_0)	1998	3338	96	12	1159	1548	450	1209	8	0	0	0	0

Figure 18: utilization report after implementation run

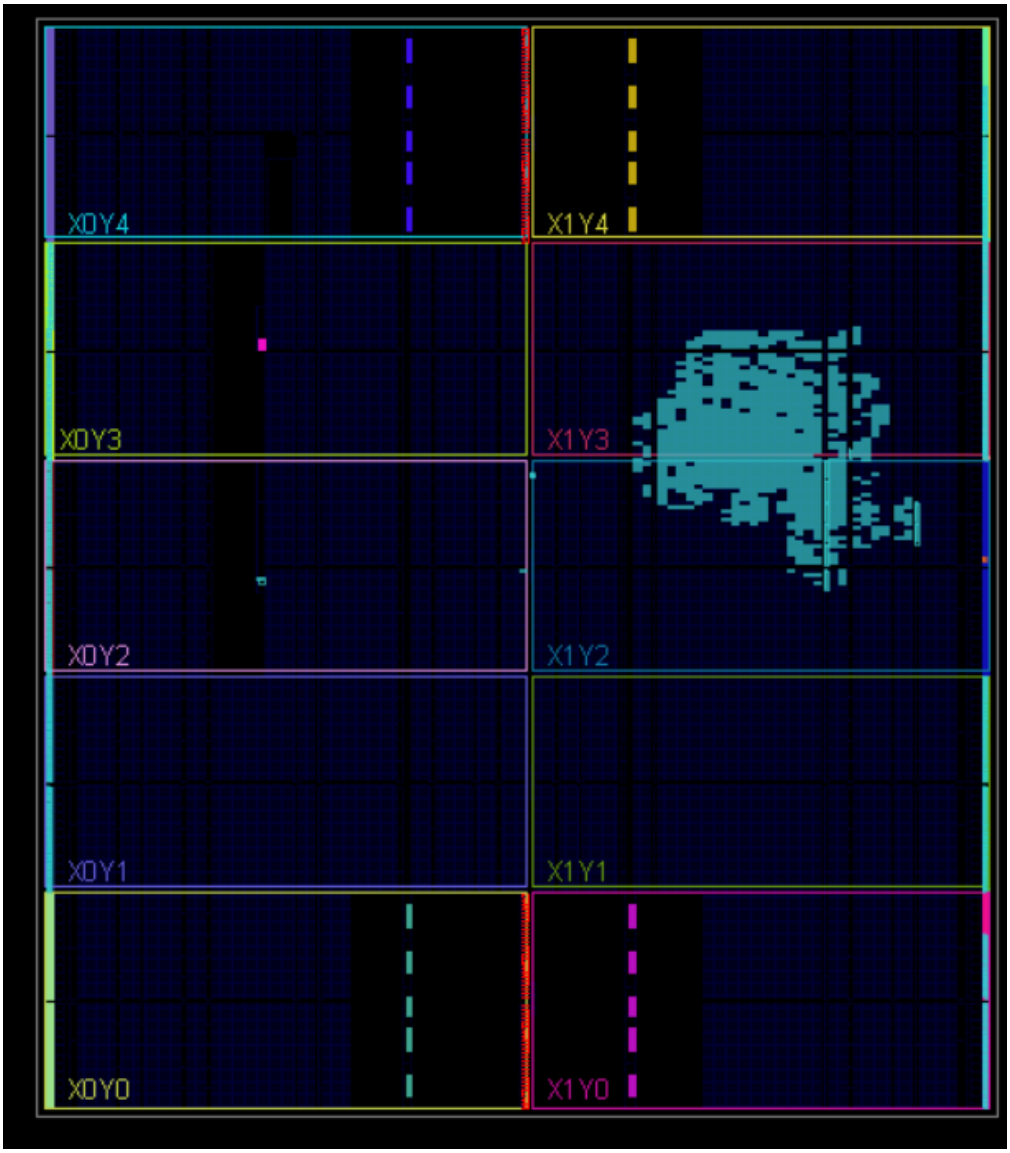


Figure 19: device

9-Linting:

Linting check showed just one warning as shown below:

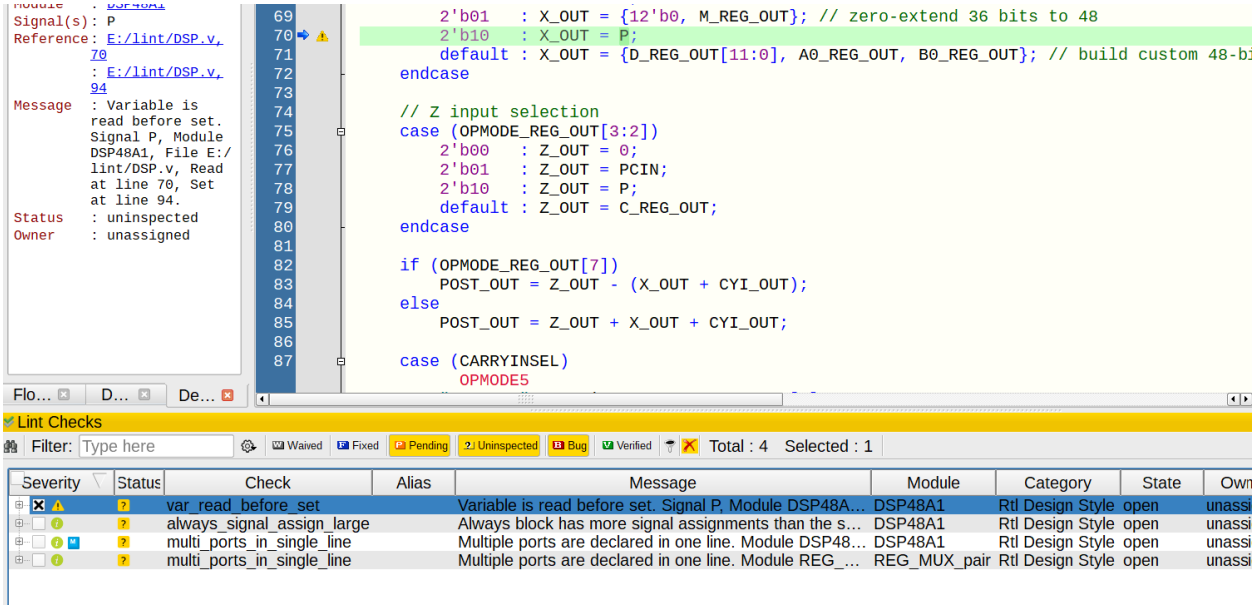


Figure 20: Warning in lint tool

This warning will not affect our design cause reset will

Be activated at the beginning so P will be equal to 0

And we want P as feedback for X input selection so,

We will just waive this warning.

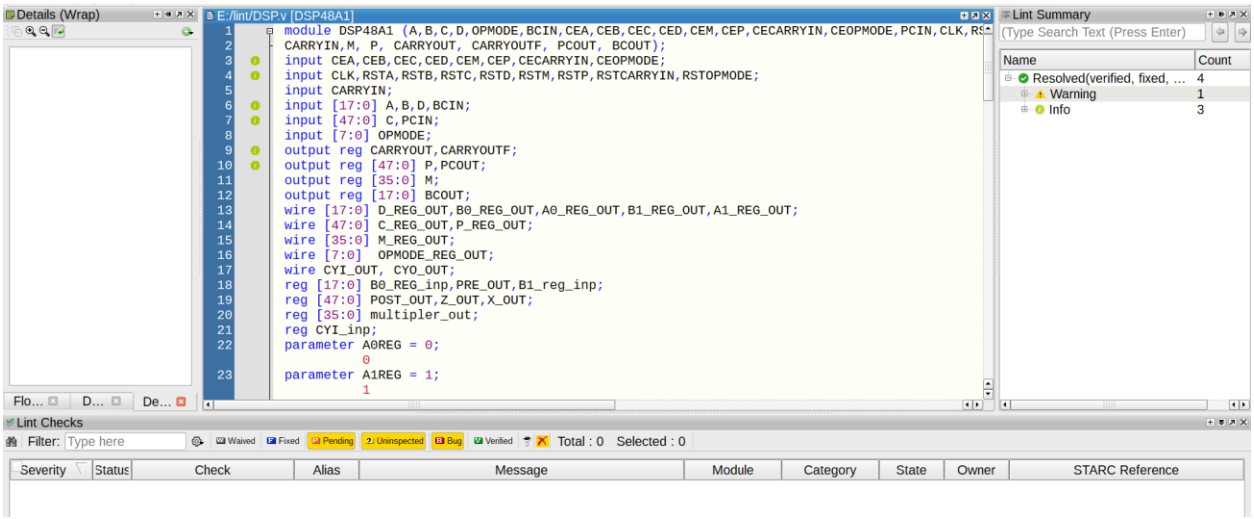


Figure 21: Lint snippet showing no Errors