

1-RTL code:

```
1  module REG_MUX_pair(clk,rst,inp,CE,out);
2  parameter RESETTYPE="SYNC";
3  parameter SELECT=0;
4  parameter WIDTH=18;
5  input  clk,rst,CE;
6  input  [WIDTH-1:0] inp;
7  output [WIDTH-1:0] out;
8  reg [WIDTH-1:0] out_comb;
9  reg [WIDTH-1:0] out_reg;
10 assign out = out_comb;
11 generate
12     if(RESETTYPE=="ASYNC") begin
13         always @(posedge clk or posedge rst) begin
14             if(rst)
15                 out_reg<=0;
16             else begin
17                 if(CE)
18                     out_reg<=inp;
19             end
20         end
21     end
22     else begin
23         always @(posedge clk) begin
24             if(rst)
25                 out_reg<=0;
26             else begin
27                 if(CE)
28                     out_reg<=inp;
29             end
30         end
31     end
32 endgenerate
33 always @(*) begin
34     case (SELECT)
35         0: out_comb=inp;
36         default: out_comb=out_reg;
37     endcase
38 end
39 endmodule
```

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

```

1 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,
2 CARRYIN,M,P,CARRYOUT,CARRYOUTF,PCOUT,BCOUT);
3 input CEA,CEB,CEC,CED,CEM,CEP,CECARRYIN,CEOPMODE;
4 input CLK,RSTA,RSTB,RSTC,RSTD,RSTM,RSTP,RSTCARRYIN,RSTOPMODE;
5 input CARRYIN;
6 input [17:0] A,B,D,BCIN;
7 input [47:0] C,PCIN;
8 input [7:0] OPMODE;
9 output reg CARRYOUT,CARRYOUTF;
10 output reg [47:0] P,PCOUT;
11 output reg [35:0] M;
12 output reg [17:0] BCOUT;
13 wire [17:0] D_REG_OUT,B0_REG_OUT,A0_REG_OUT,B1_REG_OUT,A1_REG_OUT;
14 wire [47:0] C_REG_OUT,P_REG_OUT;
15 wire [35:0] M_REG_OUT;
16 wire [7:0] OPMODE_REG_OUT;
17 wire CYI_OUT,CYO_OUT;
18 reg [17:0] B0_REG_in,PRE_OUT,B1_reg_in;
19 reg [47:0] POST_OUT,Z_OUT,X_OUT;
20 reg [35:0] multiplier_out;
21 reg CYI_in;
22 parameter A0REG = 0;
23 parameter A1REG = 1;
24 parameter B0REG = 0;
25 parameter B1REG = 1;
26 parameter CREG = 1;
27 parameter DREG = 1;
28 parameter MREG = 1;
29 parameter PREG = 1;
30 parameter CARRYINREG = 1;
31 parameter CARRYOUTREG = 1;
32 parameter OPMODEREG = 1;
33 parameter CARRYINSEL = "OPMODES";
34 parameter B_INPUT = "DIRECT";
35 parameter RSTTYPE = "SYNC";
36 REG_MUX_pair #(.WIDTH(8),.SELECT(OPMODEREG)) OPMODE_REG(CLK,RSTOPMODE,OPMODE,CEOPMODE,OPMODE_REG_OUT);
37 REG_MUX_pair #(.WIDTH(18),.SELECT(DREG)) D_REG(CLK,RSTD,D,CED,D_REG_OUT);
38 REG_MUX_pair #(.WIDTH(18),.SELECT(B0REG)) B0_REG(CLK,RSTB,B0_REG_in,CEB,B0_REG_OUT);
39 REG_MUX_pair #(.WIDTH(18),.SELECT(A0REG)) A0_REG(CLK,RSTA,A,CEA,A0_REG_OUT);
40 REG_MUX_pair #(.WIDTH(48),.SELECT(CREG)) C_REG(CLK,RSTC,C,CEC,C_REG_OUT);
41 REG_MUX_pair #(.WIDTH(18),.SELECT(B1REG)) B1_REG(CLK,RSTB,B1_reg_in,CEB,B1_REG_OUT);
42 REG_MUX_pair #(.WIDTH(18),.SELECT(A1REG)) A1_REG(CLK,RSTA,A0_REG_OUT,CEA,A1_REG_OUT);
43 REG_MUX_pair #(.WIDTH(36),.SELECT(MREG)) M_REG(CLK,RSTM,multiplier_out,CEM,M_REG_OUT);
44 REG_MUX_pair #(.WIDTH(1),.SELECT(CARRYINREG)) CYI(CLK,RSTCARRYIN,CYI_in,CECARRYIN,CYI_OUT);
45 REG_MUX_pair #(.WIDTH(1),.SELECT(CARRYOUTREG)) CYO(CLK,RSTCARRYIN,POST_OUT[47],CECARRYIN,CYO_OUT);
46 REG_MUX_pair #(.WIDTH(48),.SELECT(PREG)) P_REG(CLK,RSTP,POST_OUT,CEP,P_REG_OUT);
47 always @(*) begin
48     case (B_INPUT)
49         "DIRECT" : B0_REG_in = B;
50         "CASCADE": B0_REG_in = BCIN;
51         default : B0_REG_in = 0;
52     endcase
53     if (OPMODE_REG_OUT[6])
54         PRE_OUT = D_REG_OUT - B0_REG_OUT;
55     else
56         PRE_OUT = D_REG_OUT + B0_REG_OUT;
57     case (OPMODE_REG_OUT[4])
58         0 : B1_reg_in = B0_REG_OUT;
59         default : B1_reg_in = PRE_OUT;
60     endcase
61     BCOUT = B1_REG_OUT;
62     multiplier_out = B1_REG_OUT * A1_REG_OUT;
63     // X input selection
64     case (OPMODE_REG_OUT[1:0])
65         2'b00 : X_OUT = 0;
66         2'b01 : X_OUT = {12'b0, M_REG_OUT}; // zero-extend 36 bits to 48
67         2'b10 : X_OUT = P;
68         default : X_OUT = {D_REG_OUT[11:0], A0_REG_OUT, B0_REG_OUT}; // build custom 48-bit value
69     endcase
70     // Z input selection
71     case (OPMODE_REG_OUT[3:2])
72         2'b00 : Z_OUT = 0;
73         2'b01 : Z_OUT = PCIN;
74         2'b10 : Z_OUT = P;
75         default : Z_OUT = C_REG_OUT;
76     endcase
77     if (OPMODE_REG_OUT[7])
78         POST_OUT = Z_OUT - (X_OUT + CYI_OUT);
79     else
80         POST_OUT = Z_OUT + X_OUT + CYI_OUT;
81     case (CARRYINSEL)
82         "OPMODES": CYI_in = OPMODE_REG_OUT[5];
83         "CARRYIN": CYI_in = CARRYIN;
84         default : CYI_in = 0;
85     endcase
86     PCOUT = P_REG_OUT;
87     P = P_REG_OUT;
88     M = M_REG_OUT;
89     CARRYOUT = CYO_OUT;
90     CARRYOUTF = CYO_OUT;
91 end
92 endmodule

```

Figure 2: DSP RTL code

2-Testbench code:

```
1 module DSP48A1_tb();
2 reg [17:0] A,B,D,BCIN;
3 reg [47:0] C,PCIN;
4 reg [7:0] OPMODE;
5 reg CLK,CEA,CEB,CEC,CEB,CEM,CEP,CECARRYIN,CEOPMODE,RSTA,RSTB,RSTC,RSTD,RSTM,RSTP,RSTCARRYIN,RSTOPMODE,CARRYIN;
6 wire [47:0] PCOUT_OUT,P_OUT;
7 wire [17:0] BCOUT_OUT;
8 wire [35:0] M_OUT;
9 wire CARRYOUT_OUT,CARRYOUTF_OUT;
10 reg [47:0] PCOUT_expected,P_expected;
11 reg [17:0] BCOUT_expected;
12 reg [35:0] M_expected;
13 reg CARRYOUT_expected,CARRYOUTF_expected;
14 DSP48A1 DUT(A,B,C,D,OPMODE,BCIN,CEA,CEB,CEC,CEM,CEP,CECARRYIN,CEOPMODE,PCIN,CLK,RSTA,RSTB,RSTC,RSTD,RSTM,RSTP,RSTCARRYIN,RSTOPMODE,CARRYIN,M_OUT,P_OUT,CARRYOUT_OUT,CARRYOUTF_OUT,PCOUT_OUT,BCOUT_OUT);
15 initial begin
16     CLK=0;
17     forever
18         #1 CLK=~CLK;
19 end
20 Initial begin
21     // verify reset operation
22     RSTA=1;RSTB=1;RSTC=1;RSTD=1;RSTM=1;RSTP=1;RSTCARRYIN=1;RSTOPMODE=1;
23     A=$random;B=$random;D=$random;BCIN=$random;
24     C=$random;PCIN=$random;OPMODE=$random;CEA=$random;CEB=$random;CEC=$random;
25     CEM=$random;CEM=$random;CEP=$random;CECARRYIN=$random;CEOPMODE=$random;CARRYIN=$random;
26     @(negedge CLK);
27     PCOUT_expected=P_expected=0;BCOUT_expected=0;
28     M_expected=0;CARRYOUT_expected=0;CARRYOUTF_expected=0;
29     if (PCOUT_expected!=PCOUT_OUT) begin
30         $display("Error in rst operation");
31         $stop;
32     end
33     if (P_expected!=P_OUT) begin
34         $display("Error in rst operation");
35         $stop;
36     end
37     if (CARRYOUTF_expected!=CARRYOUTF_OUT) begin
38         $display("Error in rst operation");
39         $stop;
40     end
41     if (BCOUT_expected!=BCOUT_OUT) begin
42         $display("Error in rst operation");
43         $stop;
44     end
45     if (M_expected!=M_OUT) begin
46         $display("Error in rst operation");
47         $stop;
48     end
49     if (CARRYOUT_expected!=CARRYOUT_OUT) begin
50         $display("Error in rst operation");
51         $stop;
52     end
53     RSTA=0;RSTB=0;RSTC=0;RSTD=0;RSTM=0;RSTP=0;RSTCARRYIN=0;RSTOPMODE=0;
54     CEA=1;CEB=1;CEC=1;CEM=1;CEP=1;CECARRYIN=1;CEOPMODE=1;
55     // verify DSP path1
56     OPMODE=0'b1111101;
57     A=20;B=10;C=350;D=25;
58     PCIN=$random;BCIN=$random;CARRYIN=$random;
59     repeat (4) @(negedge CLK);
60     PCOUT_expected=48'h32;P_expected=48'h32;BCOUT_expected=18'hf;
61     M_expected=36'h12c;CARRYOUT_expected=0;CARRYOUTF_expected=0;
62     if (PCOUT_expected!=PCOUT_OUT) begin
63         $display("Error in DSP path 1");
64         $stop;
65     end
66     if (P_expected!=P_OUT) begin
67         $display("Error in DSP path 1");
68         $stop;
69     end
70     if (CARRYOUTF_expected!=CARRYOUTF_OUT) begin
71         $display("Error in DSP path 1");
72         $stop;
73     end
74     if (BCOUT_expected!=BCOUT_OUT) begin
75         $display("Error in DSP path 1");
76         $stop;
77     end
78     if (M_expected!=M_OUT) begin
79         $display("Error in DSP path 1");
80         $stop;
81     end
82     if (CARRYOUT_expected!=CARRYOUT_OUT) begin
83         $display("Error in DSP path 1");
84         $stop;
85     end
86 end
```

Figure 3: Testbench code part 1

```

1 // verify DSP path2
2 OPMODE=8'b00010000;
3 A=20;B=10;C=350;D=25;
4 PCIN=$random;BCIN=$random;CARRYIN=$random;
5 repeat (3) @(negedge CLK);
6 PCOUT_expected=0;P_expected=0;BCOUT_expected=18'h23;
7 M_expected=36'h2bc;CARRYOUT_expected=0;CARRYOUTF_expected=0;
8 if (PCOUT_expected!=PCOUT_DUT) begin
9     $display("Error in DSP path 2");
10    $stop;
11 end
12 if (P_expected!=P_DUT) begin
13     $display("Error in DSP path 2");
14    $stop;
15 end
16 if (CARRYOUTF_expected!=CARRYOUTF_DUT) begin
17     $display("Error in DSP path 2");
18    $stop;
19 end
20 if (BCOUT_expected!=BCOUT_DUT) begin
21     $display("Error in DSP path 2");
22    $stop;
23 end
24 if (M_expected!=M_DUT) begin
25     $display("Error in DSP path 2");
26    $stop;
27 end
28 if (CARRYOUT_expected!=CARRYOUT_DUT) begin
29     $display("Error in DSP path 2");
30    $stop;
31 end
32 // verify DSP path3
33 OPMODE=8'b00001010;
34 A=20;B=10;C=350;D=25;
35 PCIN=$random;BCIN=$random;CARRYIN=$random;
36 repeat (3) @(negedge CLK);
37 PCOUT_expected=0;P_expected=0;BCOUT_expected=18'ha;
38 M_expected=36'hc8;CARRYOUT_expected=0;CARRYOUTF_expected=0;
39 if (PCOUT_expected!=PCOUT_DUT) begin
40     $display("Error in DSP path 3");
41    $stop;
42 end
43 if (P_expected!=P_DUT) begin
44     $display("Error in DSP path 3");
45    $stop;
46 end
47 if (CARRYOUTF_expected!=CARRYOUTF_DUT) begin
48     $display("Error in DSP path 3");
49    $stop;
50 end
51 if (BCOUT_expected!=BCOUT_DUT) begin
52     $display("Error in DSP path 3");
53    $stop;
54 end
55 if (M_expected!=M_DUT) begin
56     $display("Error in DSP path 3");
57    $stop;
58 end
59 if (CARRYOUT_expected!=CARRYOUT_DUT) begin
60     $display("Error in DSP path 3");
61    $stop;
62 end
63 // verify DSP path4
64 OPMODE=8'b10100111;
65 A=5;B=6;C=350;D=25;
66 PCIN=3000;BCIN=$random;CARRYIN=$random;
67 repeat (3) @(negedge CLK);
68 PCOUT_expected=48'hfe6ffec0bb1;P_expected=48'hfe6ffec0bb1;BCOUT_expected=18'h6;
69 M_expected=36'h1e;CARRYOUT_expected=1;CARRYOUTF_expected=1;
70 if (PCOUT_expected!=PCOUT_DUT) begin
71     $display("Error in DSP path 4");
72    $stop;
73 end
74 if (P_expected!=P_DUT) begin
75     $display("Error in DSP path 4");
76    $stop;
77 end
78 if (CARRYOUTF_expected!=CARRYOUTF_DUT) begin
79     $display("Error in DSP path 4");
80    $stop;
81 end
82 if (BCOUT_expected!=BCOUT_DUT) begin
83     $display("Error in DSP path 4");
84    $stop;
85 end
86 if (M_expected!=M_DUT) begin
87     $display("Error in DSP path 4");
88    $stop;
89 end
90 if (CARRYOUT_expected!=CARRYOUT_DUT) begin
91     $display("Error in DSP path 4");
92    $stop;
93 end
94 $display("function is correct");
95 $stop;
96 end
97 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:

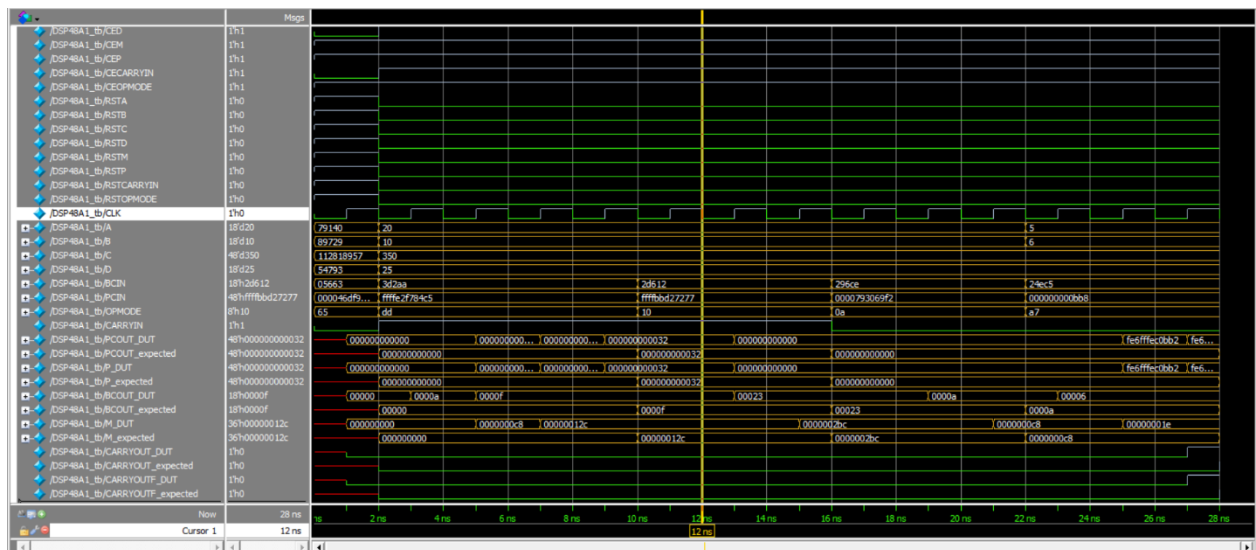


Figure 6: Simulation Waveform showing that all DUT outputs equal to expected outputs

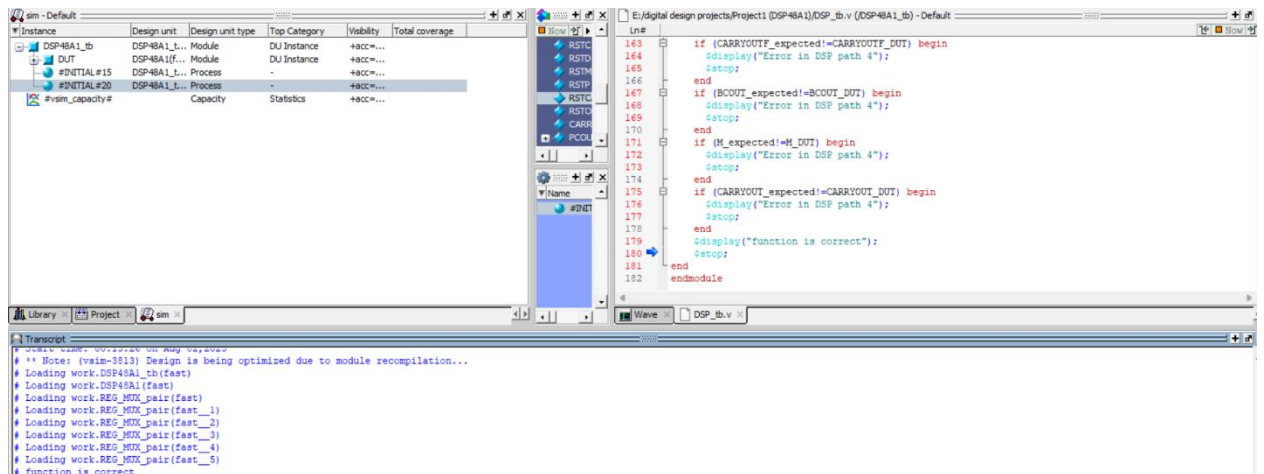


Figure 7: Transcript showing display message that function is correct

5-Constraint file:

```

1  ## This file is a general .xdc for the Basys3 rev B board
2  ## To use it in a project:
3  ## - uncomment the lines corresponding to used pins
4  ## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project
5
6  ## Clock signal
7  set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports CLK]
8  create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports CLK]
9
10
11 ## Switches
12 #set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {sw[0]}]
13 #set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {sw[1]}]
14 #set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports {sw[2]}]
15 #set_property -dict { PACKAGE_PIN W17   IOSTANDARD LVCMOS33 } [get_ports {sw[3]}]
16 #set_property -dict { PACKAGE_PIN W15   IOSTANDARD LVCMOS33 } [get_ports {sw[4]}]
17 #set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports {sw[5]}]
18 #set_property -dict { PACKAGE_PIN W14   IOSTANDARD LVCMOS33 } [get_ports {sw[6]}]
19 #set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports {sw[7]}]
20 #set_property -dict { PACKAGE_PIN V2    IOSTANDARD LVCMOS33 } [get_ports {sw[8]}]
21 #set_property -dict { PACKAGE_PIN T3    IOSTANDARD LVCMOS33 } [get_ports {sw[9]}]
22 #set_property -dict { PACKAGE_PIN T2    IOSTANDARD LVCMOS33 } [get_ports {sw[10]}]
23 #set_property -dict { PACKAGE_PIN R3    IOSTANDARD LVCMOS33 } [get_ports {sw[11]}]
24 #set_property -dict { PACKAGE_PIN W2    IOSTANDARD LVCMOS33 } [get_ports {sw[12]}]
25 #set_property -dict { PACKAGE_PIN U1    IOSTANDARD LVCMOS33 } [get_ports {sw[13]}]
26 #set_property -dict { PACKAGE_PIN T1    IOSTANDARD LVCMOS33 } [get_ports {sw[14]}]
27 #set_property -dict { PACKAGE_PIN R2    IOSTANDARD LVCMOS33 } [get_ports {sw[15]}]

```

Figure 8: Constraint File code

6-Elaboration:

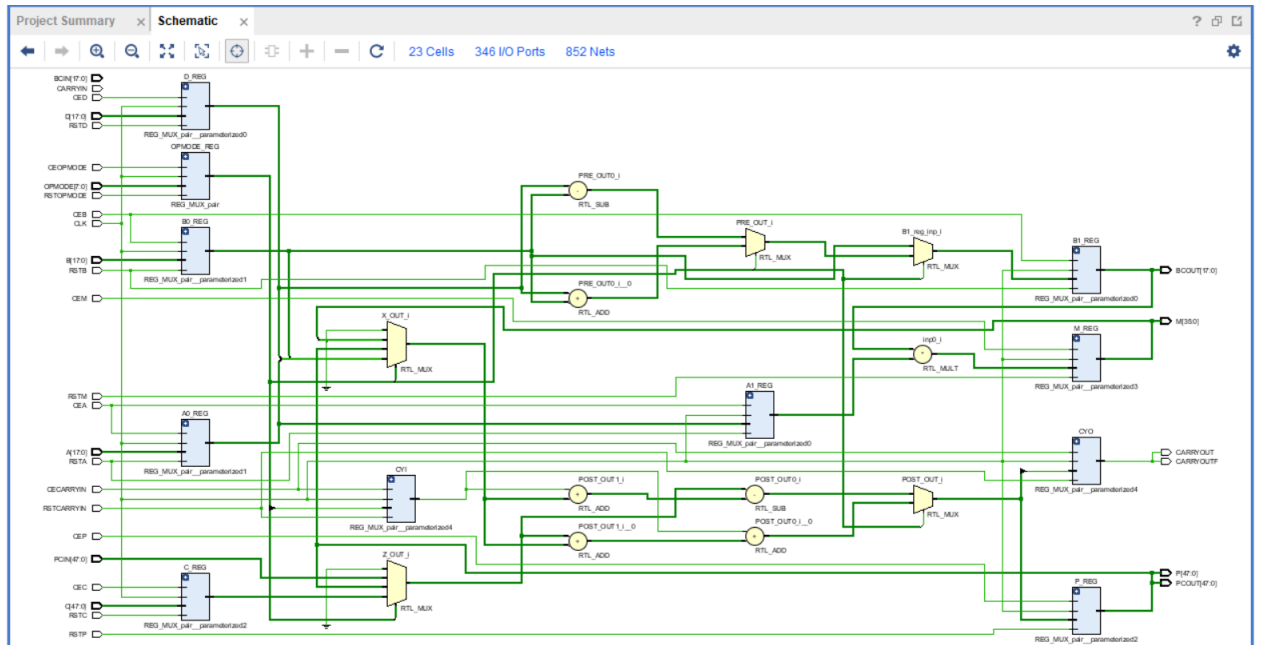


Figure 9: Elaboration schematic

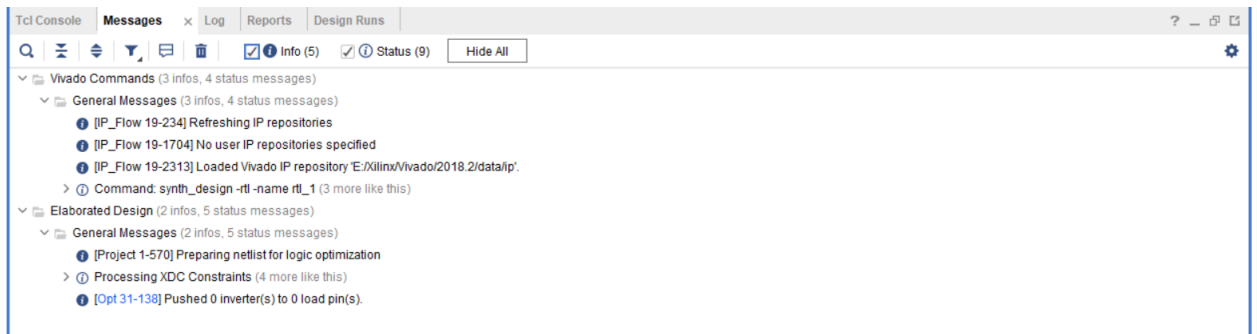


Figure 10: messages tab after Elaboration showing no Error

7-Synthesis:

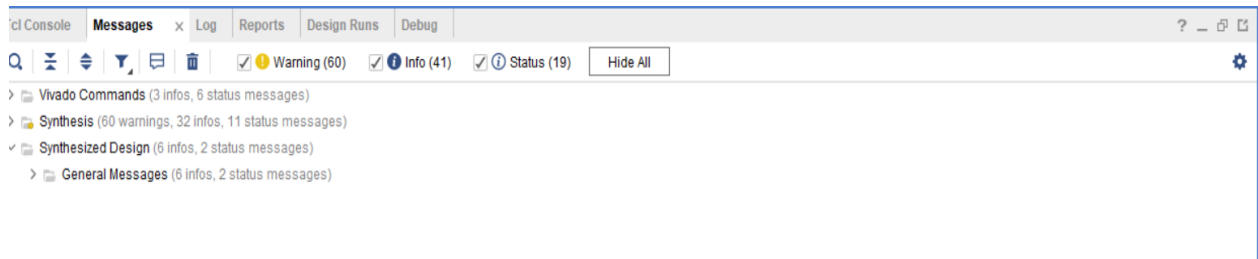


Figure 11: messages tab after Synthesis showing no Error

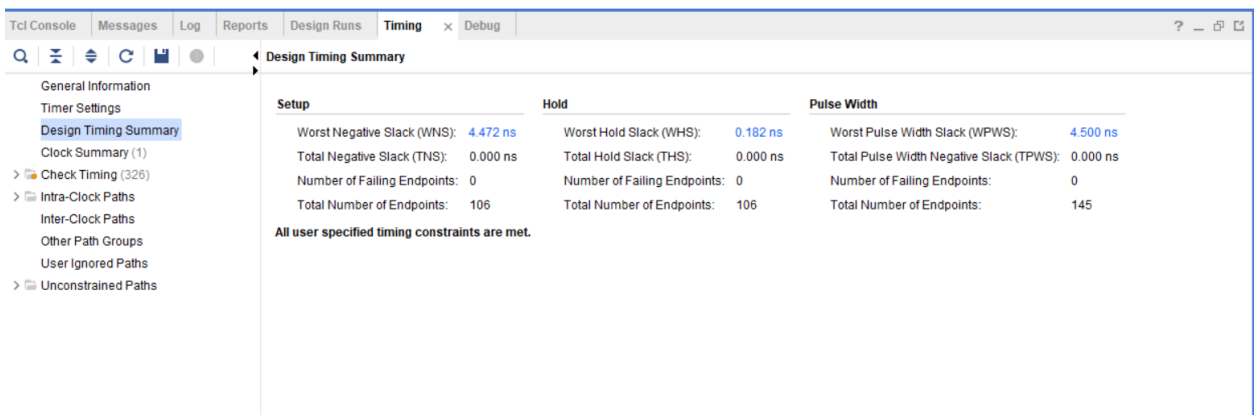


Figure 12: Timing report after synthesis run

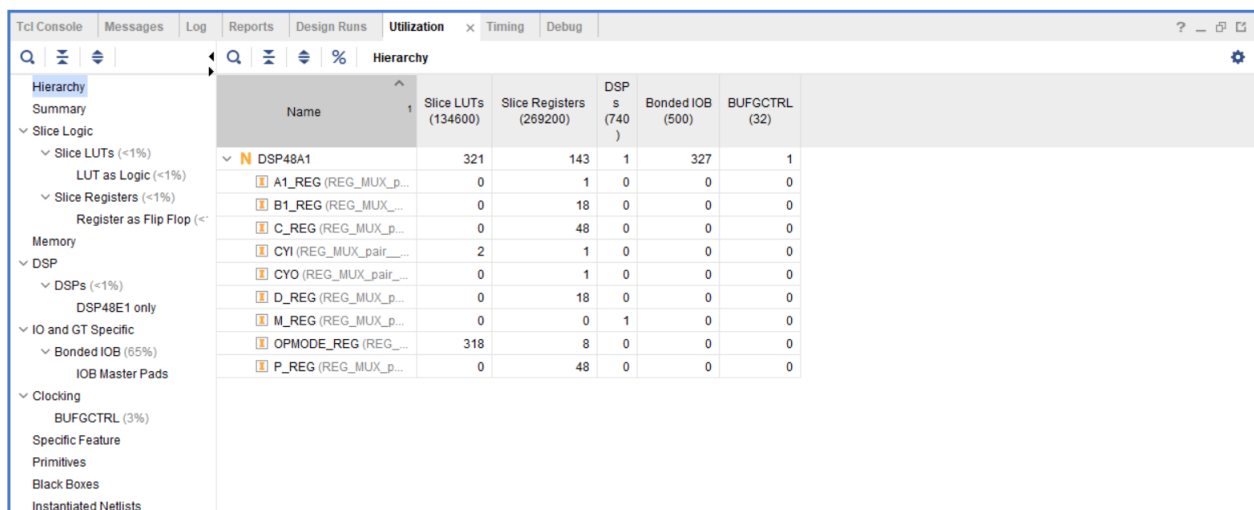


Figure 13:utilization report after synthesis run

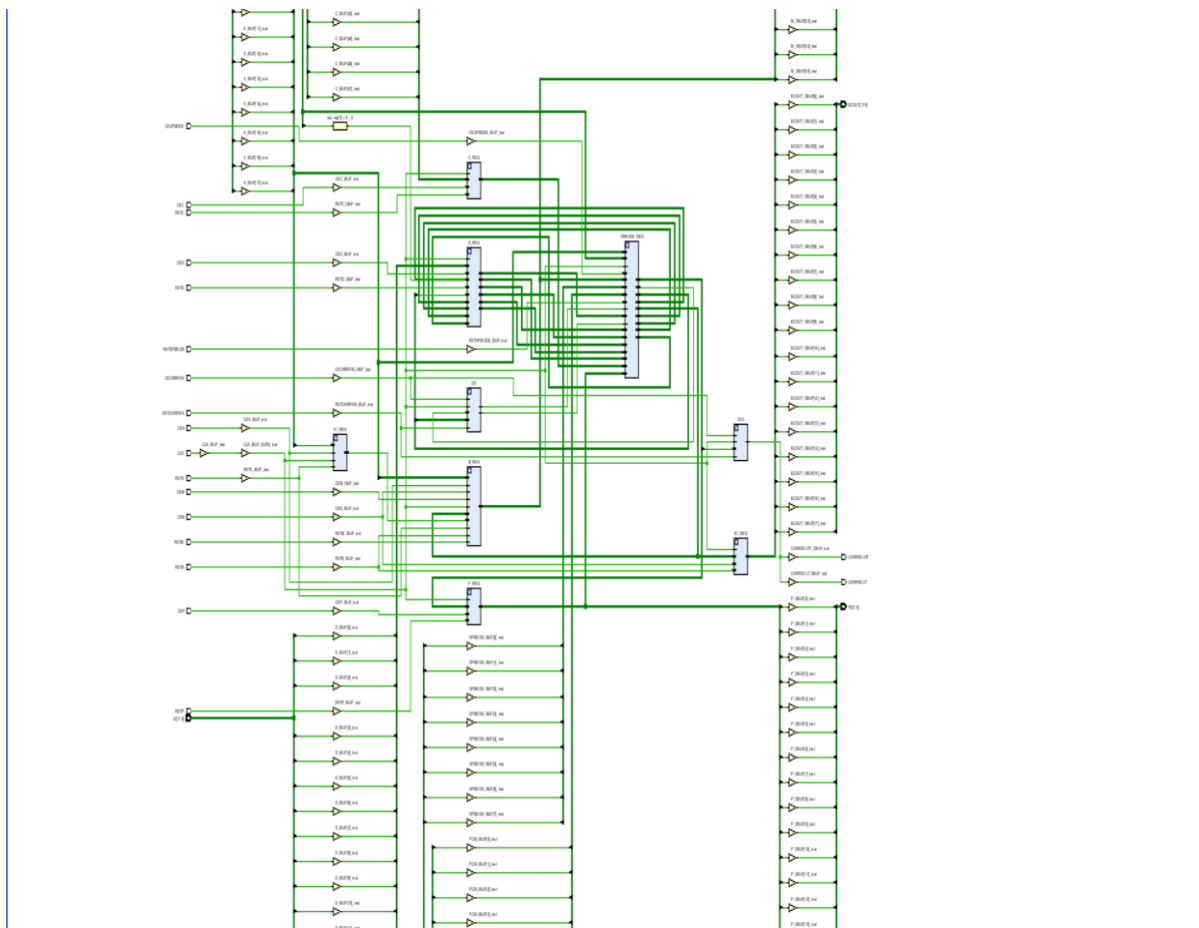


Figure 14: synthesis schematic

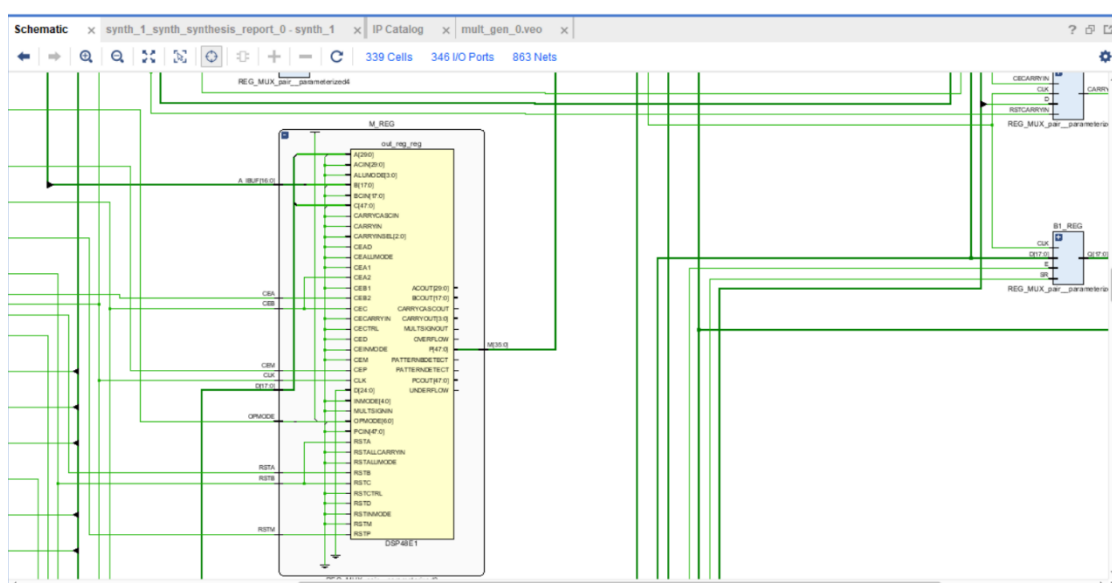


Figure 15: more obvious photo for synthesis schematic Showing DSP block at multiplier out

8-Implementation:

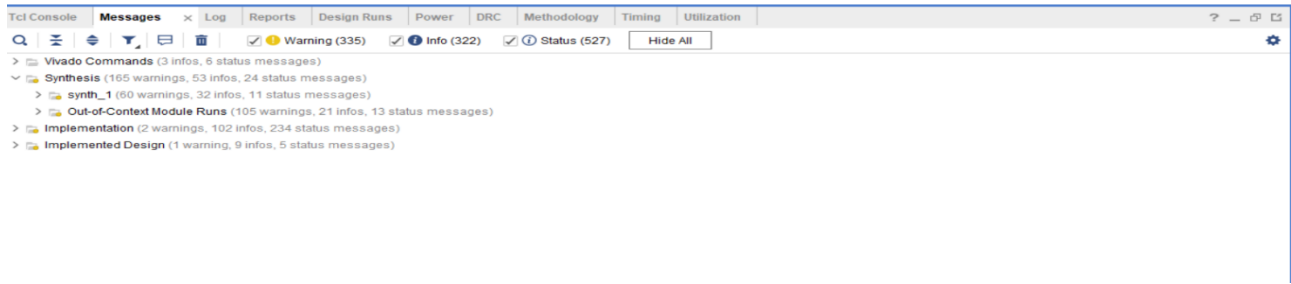


Figure 16: messages tab after implementation showing no Error

Setup			Hold			Pulse Width		
Worst Negative Slack (WNS):	2.398 ns		Worst Hold Slack (WHS):	0.060 ns		Worst Pulse Width Slack (WPWS):	3.950 ns	
Total Negative Slack (TNS):	0.000 ns		Total Hold Slack (THS):	0.000 ns		Total Pulse Width Negative Slack (TPWS):	0.000 ns	
Number of Failing Endpoints:	0		Number of Failing Endpoints:	0		Number of Failing Endpoints:	0	
Total Number of Endpoints:	8077		Total Number of Endpoints:	8061		Total Number of Endpoints:	5119	

All user specified timing constraints are met.

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)	BUFCTRL (32)	BSCAN2 (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

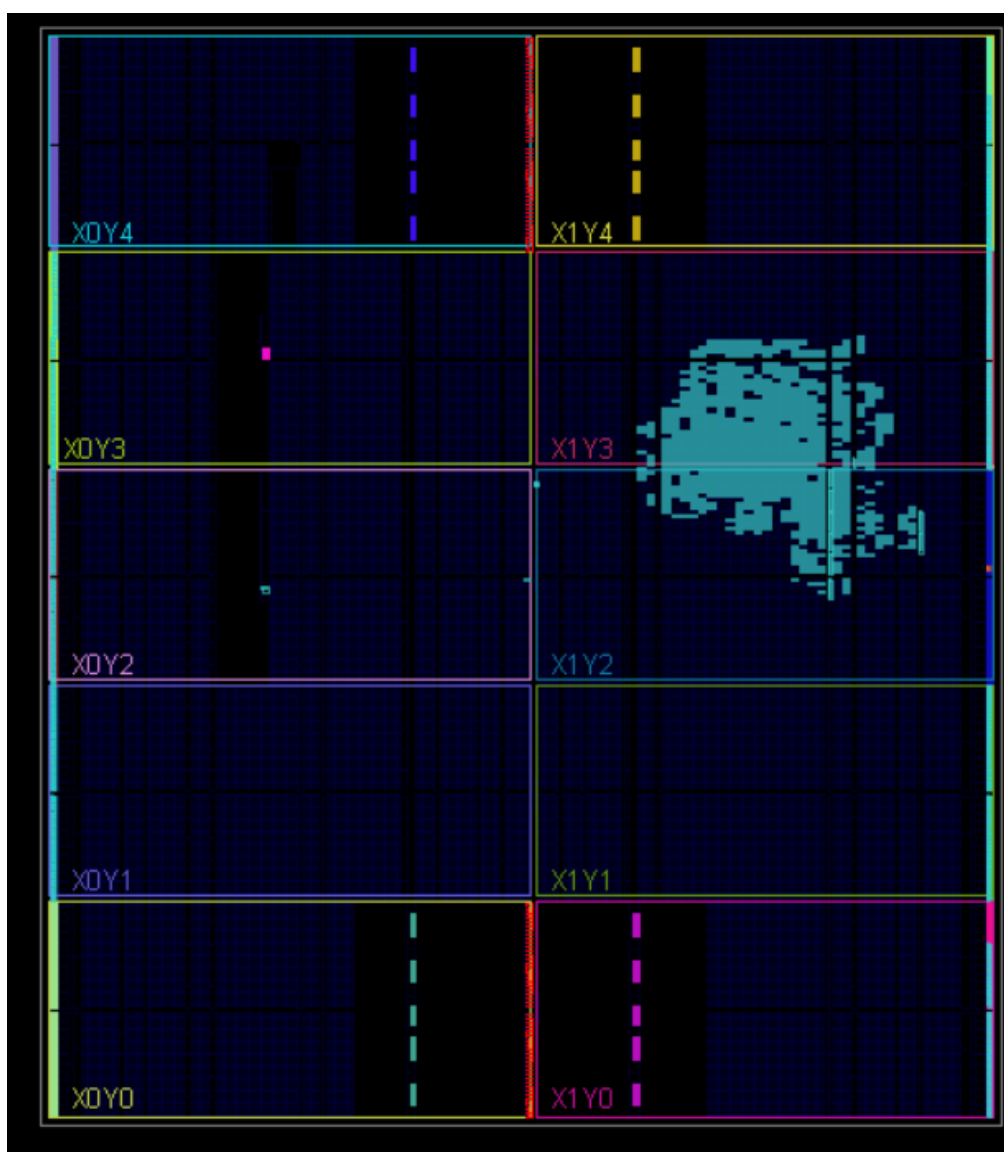


Figure 19: device

9-Linting:

Linting check showed just one warning as shown below:

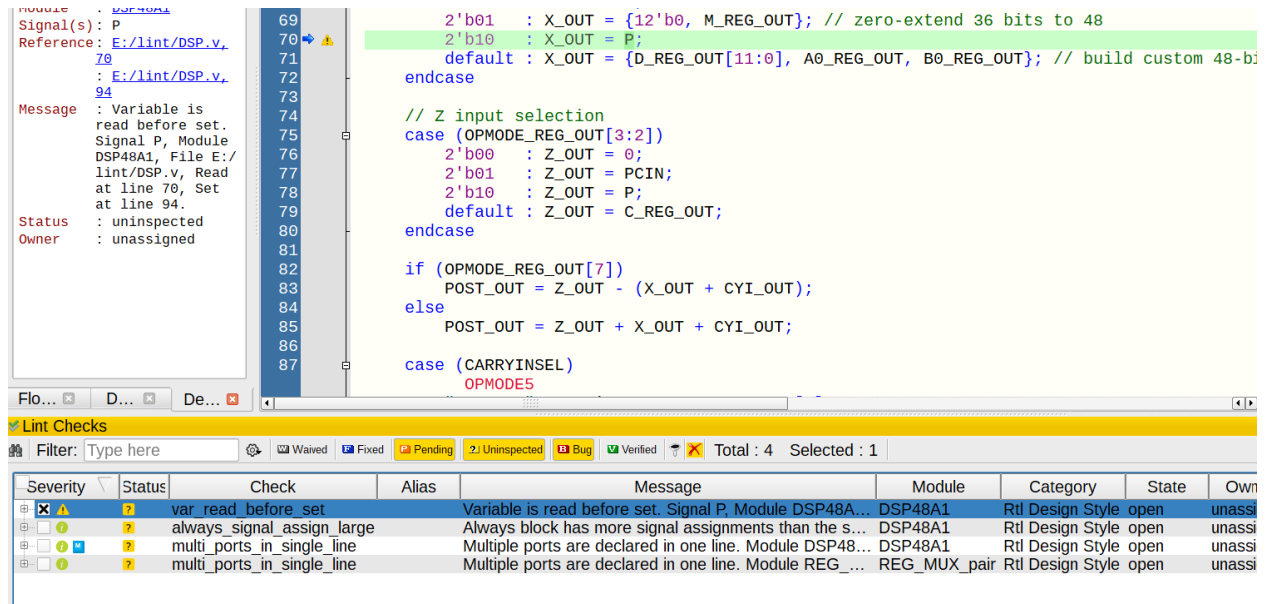


Figure 20: Warning in lint tool

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.

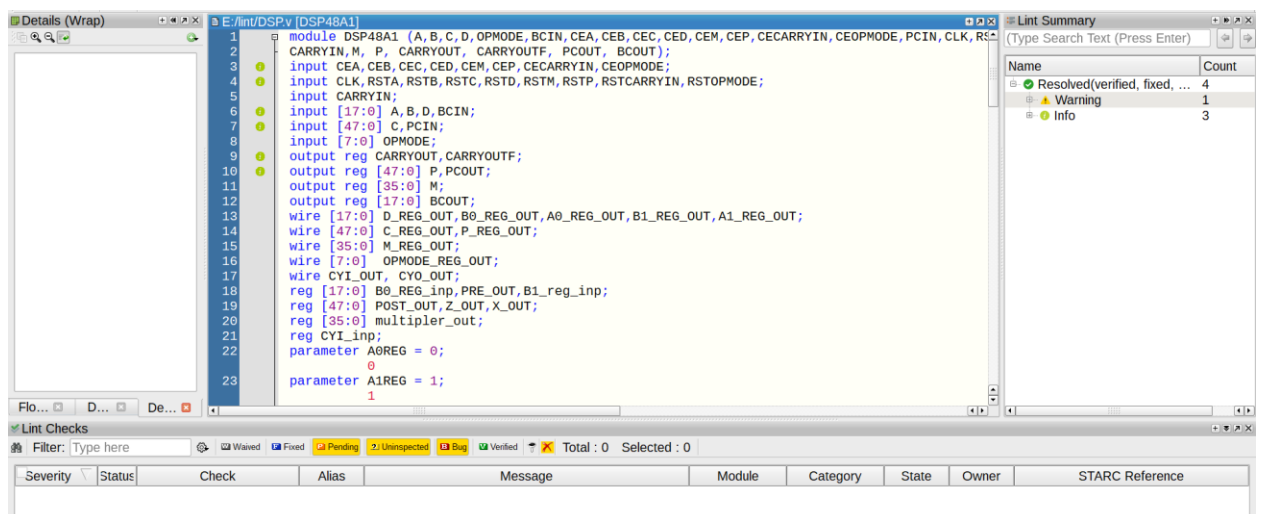


Figure 21: Lint snippet showing no Errors