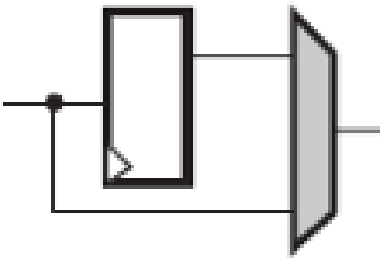
Contents

1 DSP's RTL

1.1 Register-Mux Stage



```
module reg_mux_stage (data,clk,rst,mux_out,clk_en,sel);

    parameter D_WIDTH = 18;
    parameter RSTTYPE = "SYNC";

    input [D_WIDTH-1 : 0] data;
    input rst,clk,clk_en,sel;
    output [D_WIDTH-1 : 0] mux_out;

    reg [D_WIDTH-1 : 0] data_reg;

    assign mux_out = (sel) ? data_reg : data;

    generate
        if (RSTTYPE == "SYNC") begin
            always @(posedge clk) begin
                if (rst)
                    data_reg <= 0;
                else if (clk_en)
                    data_reg <= data;
            end
        end else if (RSTTYPE == "ASYNC") begin
            always @(posedge clk or posedge rst) begin
                if (rst)
                    data_reg <= 0;
                else if (clk_en)
                    data_reg <= data;
            end
        end
    endgenerate

endmodule : reg_mux_stage
```

1.2 DSP

```
module DSP48A1 (A,B,C,D,CARRYIN,M,P,CARRYOUT,CARRYOUTF,CLK,OPMODE,CEA,CEB,CEC,CECARRYIN,CED,CEM,CEP,CEOPMODE
                ,RSTA,RSTB,RSTC,RSTCARRYIN,RSTD,RSTM,RSTP,RSTOPMODE,BCIN,BCOUT,PCIN,PCOUT);

    parameter A0REG=0, A1REG=1, B0REG=0, B1REG=1, CREG=1, DREG=1, MREG=1, PREG=1, CARRYINREG=1, CARRYOUTREG=1, OPMODEREG=1;
    parameter CARRYINSEL = "OPMODE5", B_INPUT = "DIRECT", RSTTYPE = "SYNC";

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

    wire [17:0] A0_mux_out,B0_mux1_out,D_mux_out,B0_mux0_out,ALU0_out,mux_opmode4_out,B1_mux_out,A1_mux_out;
    wire [47:0] C_mux_out,ALU1_out;
    wire [35:0] multiply_out,M_mux_out;
    wire [7:0] OPMODE_mux_out;
    wire mux_carryin_out,Cin_mux_out,CYO,CYO_mux_out;
    reg [47:0] mux_x_out,mux_z_out;

    reg_mux_stage #(.RSTTYPE(RSTTYPE),.D_WIDTH(18)) A0_SIG (A,CLK,RSTA,A0_mux_out,CEA,A0REG);
    reg_mux_stage #(.RSTTYPE(RSTTYPE),.D_WIDTH(18)) B0_SIG (B0_mux0_out,CLK,RSTB,B0_mux1_out,CEB,B0REG);
    reg_mux_stage #(.RSTTYPE(RSTTYPE),.D_WIDTH(48)) C_SIG (C,CLK,RSTC,C_mux_out,CEC,CREG);
    reg_mux_stage #(.RSTTYPE(RSTTYPE),.D_WIDTH(18)) D_SIG (D,CLK,RSTD,D_mux_out,CED,DREG);
    reg_mux_stage #(.RSTTYPE(RSTTYPE),.D_WIDTH(8)) OPMODE_SIG (OPMODE,CLK,RSTOPMODE,OPMODE_mux_out,CEOPMODE,OPMODEREG);
    reg_mux_stage #(.RSTTYPE(RSTTYPE),.D_WIDTH(18)) A1_SIG (A0_mux_out,CLK,RSTA,A1_mux_out,CEA,A1REG);
    reg_mux_stage #(.RSTTYPE(RSTTYPE),.D_WIDTH(18)) B1_SIG (mux_opmode4_out,CLK,RSTB,B1_mux_out,CEB,B1REG);
    reg_mux_stage #(.RSTTYPE(RSTTYPE),.D_WIDTH(36)) M_SIG (multiply_out,CLK,RSTM,M_mux_out,CEM,MREG);
    reg_mux_stage #(.RSTTYPE(RSTTYPE),.D_WIDTH(1)) CYI_SIG (mux_carryin_out,CLK,RSTCARRYIN,Cin_mux_out,CECARRYIN,CARRYINREG);
    reg_mux_stage #(.RSTTYPE(RSTTYPE),.D_WIDTH(1)) CYO_SIG (CYO,CLK,RSTCARRYIN,CYO_mux_out,CECARRYIN,CARRYOUTREG);
    reg_mux_stage #(.RSTTYPE(RSTTYPE),.D_WIDTH(48)) P_SIG (ALU1_out,CLK,RSTP,PCOUT,CEP,PREG);

    assign B0_mux0_out = (B_INPUT == "DIRECT") ? B :
                         (B_INPUT == "CASCADE") ? BCIN : 0;

    assign ALU0_out = (OPMODE_mux_out[6] == 0) ? (D_mux_out + B0_mux1_out) : (D_mux_out - B0_mux1_out);
    assign mux_opmode4_out = (OPMODE_mux_out[4] == 0) ? B0_mux1_out : ALU0_out;
    assign BCOUT = B1_mux_out;
    assign multiply_out = A1_mux_out * B1_mux_out;
    assign M = M_mux_out;
    assign mux_carryin_out = (CARRYINSEL == "CARRYIN") ? CARRYIN :
                             (CARRYINSEL == "OPMODE5") ? OPMODE_mux_out[5] : 0;

    always @(*) begin
        case (OPMODE_mux_out[1:0])
            2'b00 : mux_x_out = 0;
            2'b01 : mux_x_out = {{12{1'b0}},M_mux_out};
            2'b10 : mux_x_out = PCOUT;
            2'B11 : mux_x_out = {D_mux_out[11:0],A1_mux_out,B1_mux_out};
            default : mux_x_out = 0;
        endcase

        case (OPMODE_mux_out[3:2])
            2'b00 : mux_z_out = 0;
            2'b01 : mux_z_out = PCIN;
            2'b10 : mux_z_out = PCOUT;
            2'b11 : mux_z_out = C_mux_out;
            default : mux_z_out = 0;
        endcase
    end

    assign {CYO,ALU1_out} = (OPMODE_mux_out[7]) ? (mux_z_out - (mux_x_out+Cin_mux_out)) : (mux_x_out + mux_z_out + Cin_mux_out);
    assign CARRYOUT = CYO_mux_out;
    assign CARRYOUTF = CYO_mux_out;
    assign P = PCOUT;


endmodule : DSP48A1
```

2 DSP's Testbench

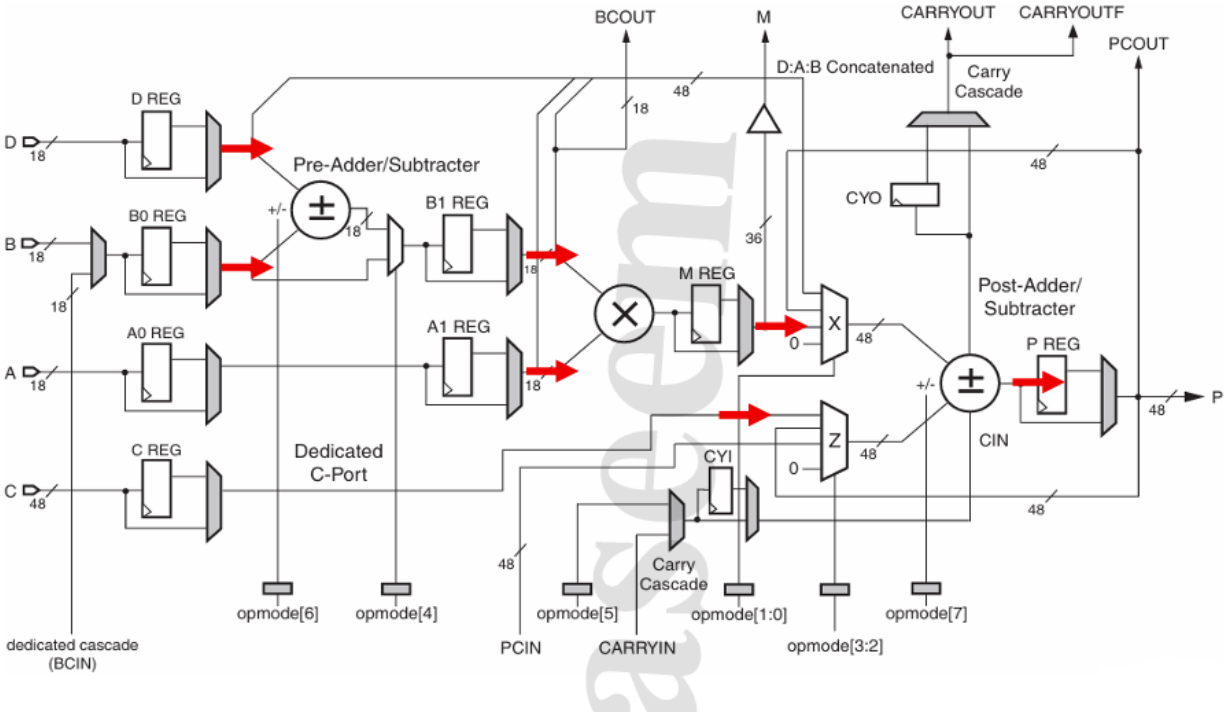
2.1 Tb signals, Instantiation and Clock generation

```
module DSP48A1_tb ();

    reg [17:0] A,B,D,BCIN;
    reg [47:0] C,PCIN;
    reg CARRYIN,CLK,CEA,CEB,CEC,CECARRYIN,CED,CEM,CEP,CEOPMODE,RSTA,RSTB,RSTC,RSTCARRYIN,RSTD,RSTM,RSTP,RSTOPMODE;
    reg [7:0] OPMODE;

    wire [35:0] M;
    wire [47:0] P,PCOUT;
    wire [17:0] BCOUT;
    wire CARRYOUT,CARRYOUTF;

    reg [35:0] M_exp;
    reg [47:0] P_exp,PCOUT_exp;
    reg [17:0] BCOUT_exp;
    reg CARRYOUT_exp,CARRYOUTF_exp;

    DSP48A1 dut (A,B,C,D,CARRYIN,M,P,CARRYOUT,CARRYOUTF,CLK,OPMODE,CEA,CEB,CEC,CECARRYIN,CED,CEM,CEP,CEOPMODE
                ,RSTA,RSTB,RSTC,RSTCARRYIN,RSTD,RSTM,RSTP,RSTOPMODE,BCIN,BCOUT,PCIN,PCOUT);

    initial begin
        CLK = 0;
        forever #5 CLK = ~CLK;
    end
```

2.2 Reset check

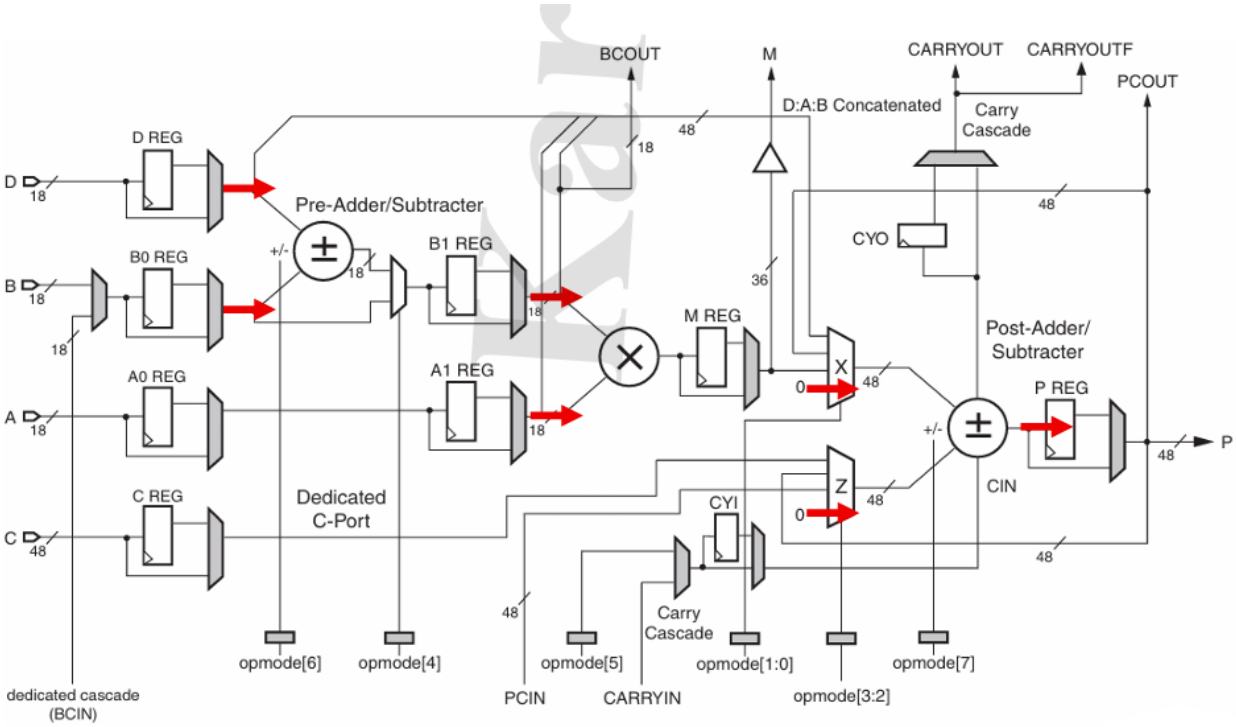
```
    initial begin
        // reset check
        RSTA = 1; RSTB = 1; RSTC = 1; RSTD = 1; RSTM = 1; RSTP = 1; RSTOPMODE = 1; RSTCARRYIN = 1;

        A = $random(); B = $random(); C = $random(); D = $random();
        BCIN = $random(); CARRYIN = $random(); PCIN = $random(); OPMODE = $random();

        CEA = $random(); CEB = $random(); CEC = $random(); CED = $random(); CECARRYIN = $random();CEM = $random();
        CEP = $random(); CEOPMODE = $random();

        M_exp = 0; P_exp = 0; PCOUT_exp = 0; BCOUT_exp = 0; CARRYOUT_exp = 0; CARRYOUTF_exp = 0;

        @(negedge CLK);

        if ((M == M_exp) && (P == P_exp) && (PCOUT == PCOUT_exp) && (BCOUT == BCOUT_exp) &&
            (CARRYOUT == CARRYOUT_exp) && (CARRYOUTF == CARRYOUTF_exp))
            $display("reset test passed");
        else begin
            $display("error in reset");
            $stop();
        end

        RSTA = 0; RSTB = 0; RSTC = 0; RSTD = 0; RSTM = 0; RSTP = 0; RSTOPMODE = 0; RSTCARRYIN = 0;
        CEA = 1; CEB = 1; CEC = 1; CED = 1; CECARRYIN = 1; CEM = 1; CEP = 1; CEOPMODE = 1;
```

2.3 Path 1 check



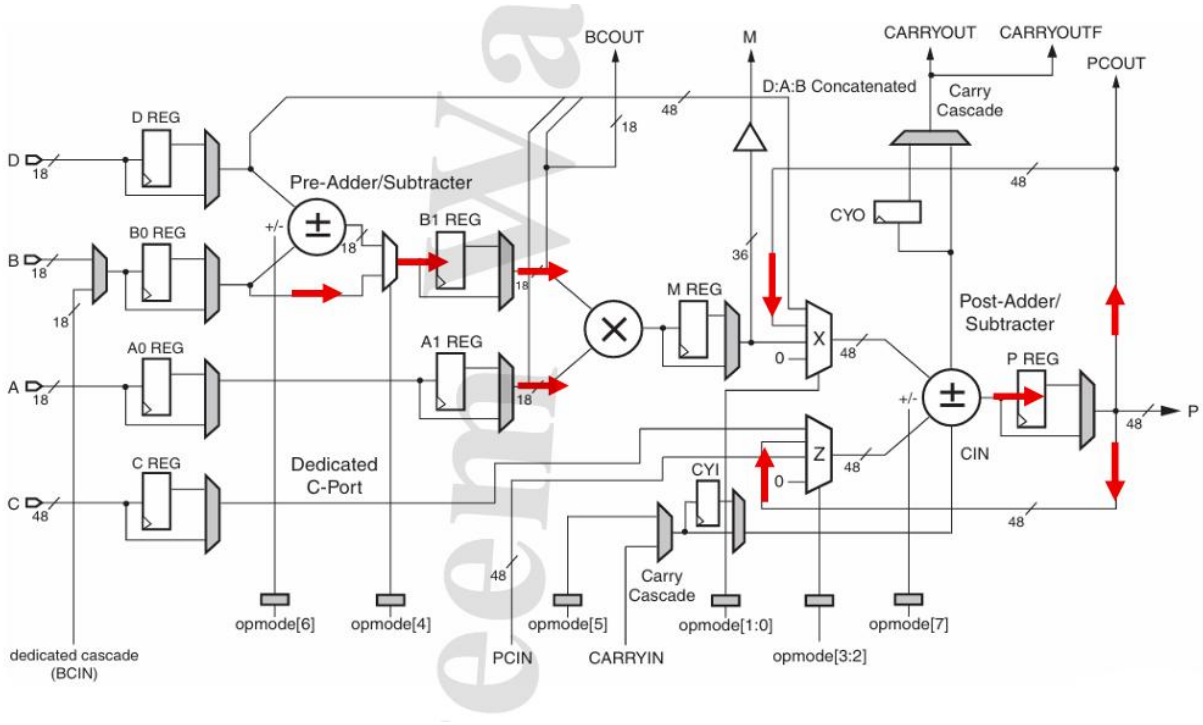

```
50
51          // path 1 test
52          A = 20; B = 10; C = 350; D = 25;
53          BCIN = $random(); CARRYIN = $random(); PCIN = $random(); OPMODE = 8'b1101_1101;
54
55          M_exp = 'h12c; P_exp = 'h32; PCOUT_exp = 'h32; BCOUT_exp = 'hf; CARRYOUT_exp = 0; CARRYOUTF_exp = 0;
56
57          repeat (4) @(negedge CLK);
58
59          if ((M == M_exp) && (P == P_exp) && (PCOUT == PCOUT_exp) && (BCOUT == BCOUT_exp) &&
60              (CARRYOUT == CARRYOUT_exp) && (CARRYOUTF == CARRYOUTF_exp))
61              $display("path 1 test passed");
62          else begin
63              $display("error in path 1");
64              $stop();
65          end
66
```

2.4 Path 2 check



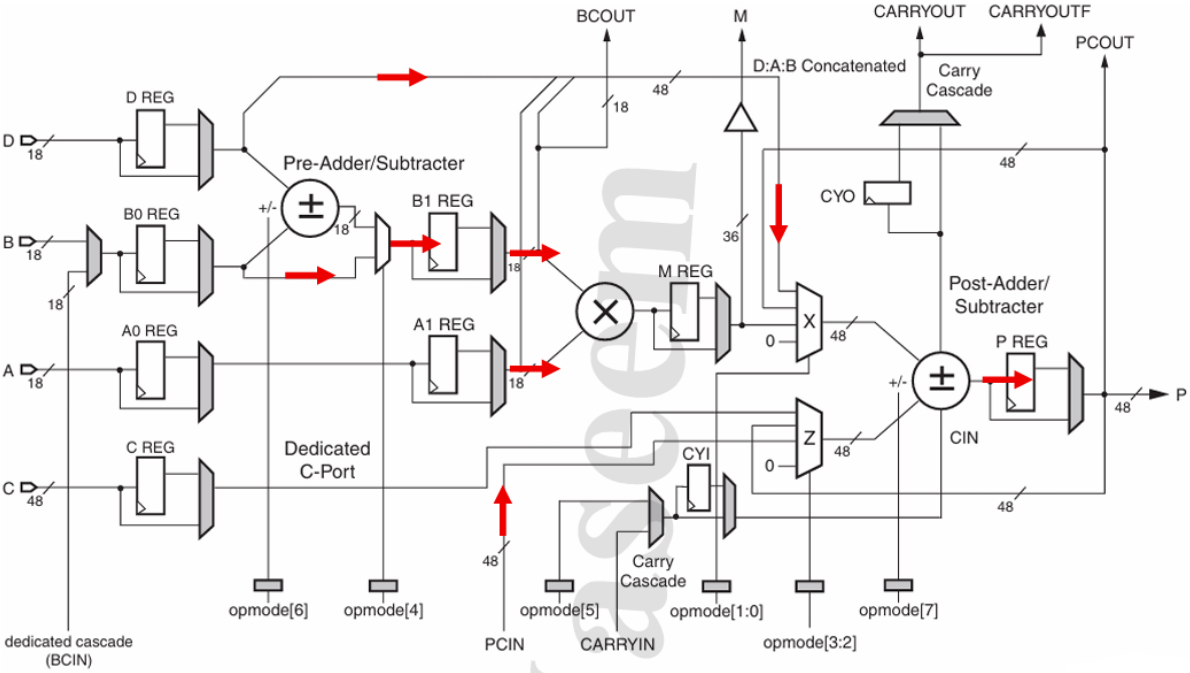

```
66
67          // path 2 test
68          A = 20; B = 10; C = 350; D = 25;
69          BCIN = $random(); CARRYIN = $random(); PCIN = $random(); OPMODE = 8'b0001_0000;
70
71          M_exp = 'h2bc; P_exp = 0; PCOUT_exp = 0; BCOUT_exp = 'h23; CARRYOUT_exp = 0; CARRYOUTF_exp = 0;
72
73          repeat (3) @(negedge CLK);
74
75          if ((M == M_exp) && (P == P_exp) && (PCOUT == PCOUT_exp) && (BCOUT == BCOUT_exp) &&
76              (CARRYOUT == CARRYOUT_exp) && (CARRYOUTF == CARRYOUTF_exp))
77              $display("path 2 test passed");
78          else begin
79              $display("error in path 2");
80              $stop();
81          end
82
```

2.5 Path 3 check




```
84          // path 3 test
85          A = 20; B = 10; C = 350; D = 25;
86          BCIN = $random(); CARRYIN = $random(); PCIN = $random(); OPMODE = 8'b0000_1010;
87
88          M_exp = 'hc8; P_exp = 0; PCOUT_exp = 0; BCOUT_exp = 'ha; CARRYOUT_exp = 0; CARRYOUTF_exp = 0;
89
90          repeat (3) @(negedge CLK);
91
92          if ((M == M_exp) && (P == P_exp) && (PCOUT == PCOUT_exp) && (BCOUT == BCOUT_exp) &&
93              (CARRYOUT == CARRYOUT_exp) && (CARRYOUTF == CARRYOUTF_exp))
94              $display("path 3 test passed");
95          else begin
96              $display("error in path 3");
97              $stop();
98          end
99
```

2.6 Path 4 check

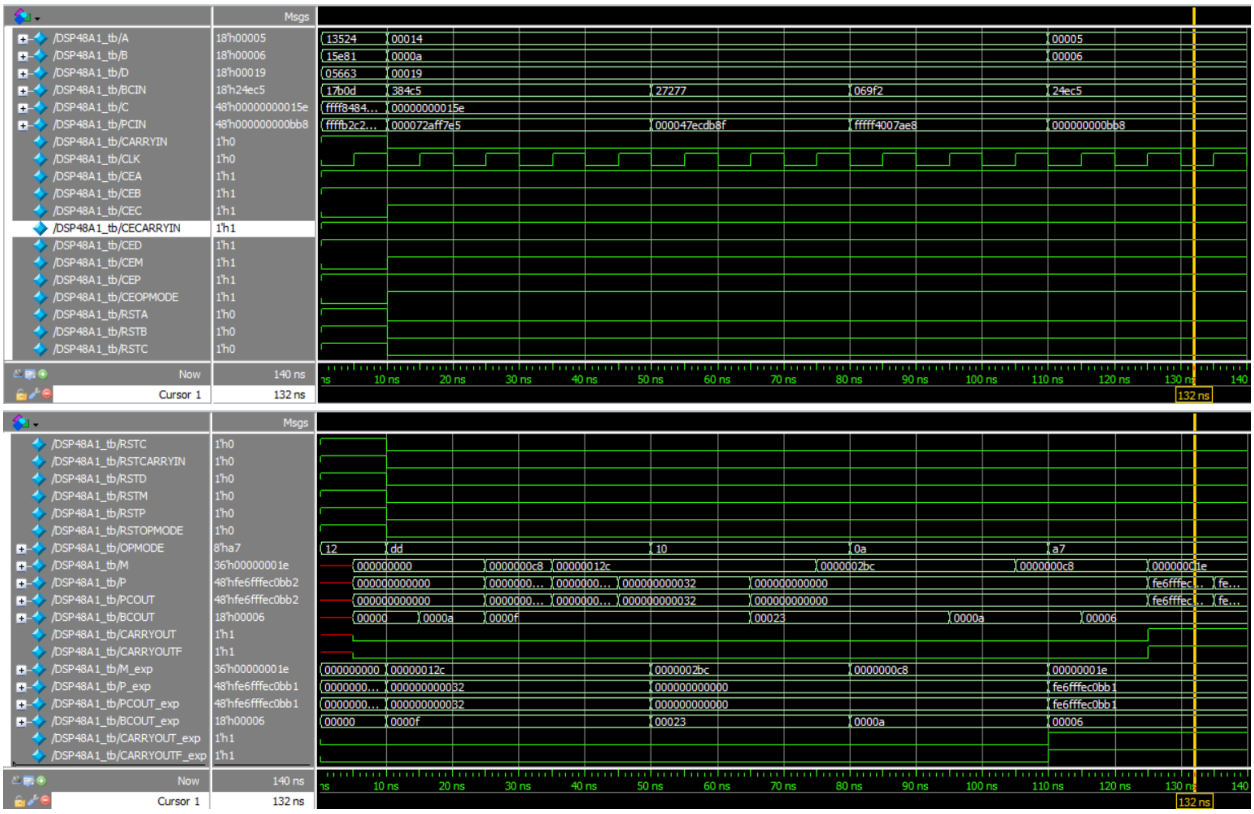



```
100
101         // path 4 test
102         A = 5; B = 6; C = 350; D = 25;
103         BCIN = $random(); CARRYIN = $random(); PCIN = 3000; OPMODE = 8'b10100111;
104
105         M_exp = 'h1e; P_exp = 'hfe6fffec0bb1; PCOUT_exp = 'hfe6fffec0bb1;
106         BCOUT_exp = 'h6; CARRYOUT_exp = 1; CARRYOUTF_exp = 1;
107
108         repeat (3) @(negedge CLK);
109
110         if ((M == M_exp) && (P == P_exp) && (PCOUT == PCOUT_exp) && (BCOUT == BCOUT_exp) &&
111             (CARRYOUT == CARRYOUT_exp) && (CARRYOUTF == CARRYOUTF_exp))
112             $display("path 4 test passed");
113         else begin
114             $display("error in path 4");
115             $stop();
116         end
117
118
119         $display("TEST PASSED :)");
120         $stop();
121     end
122
123 endmodule : DSP48A1_tb
```

3 Do file

```
1  vlib work
2  vlog  reg_mux_stage.v DSP48A1.v DSP48A1_tb.v
3  vsim -voptargs=+acc work.DSP48A1_tb
4  add wave *
5  run -all
6  #quit -sim
```

4 QuestaSim Waveform Snippets



5 Constrains file

```
## 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]


```

```

 ## Configuration options, can be used for all designs
 set_property CONFIG_VOLTAGE 1.8 [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]
```

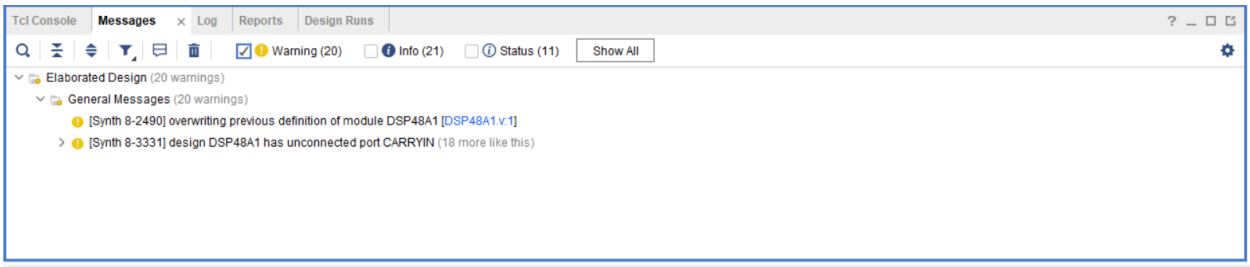
The remain is commented so it’s unused.

I made the CONFIG_VOLTAGE 1.8 instead of 3.3 due to a warning in vivado.

6 Vivado

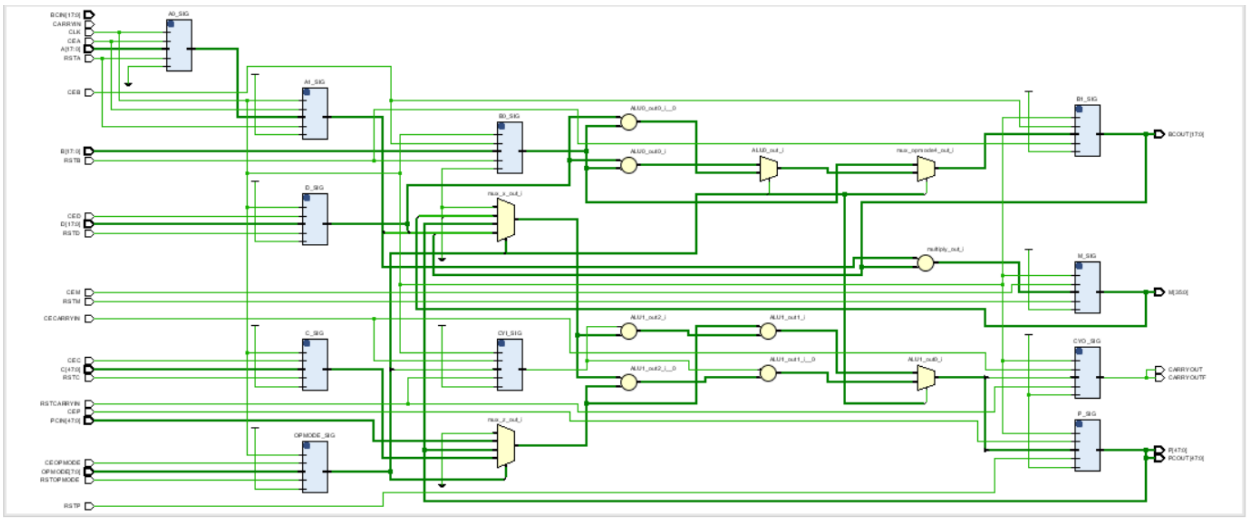
6.1 Elaboration

6.1.1 Massage tab



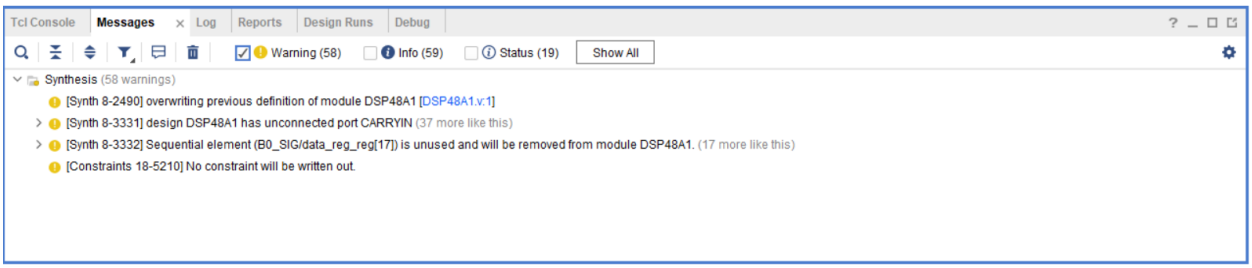
These warnings have shown as there some unused signals in the design

6.1.2 Schematic



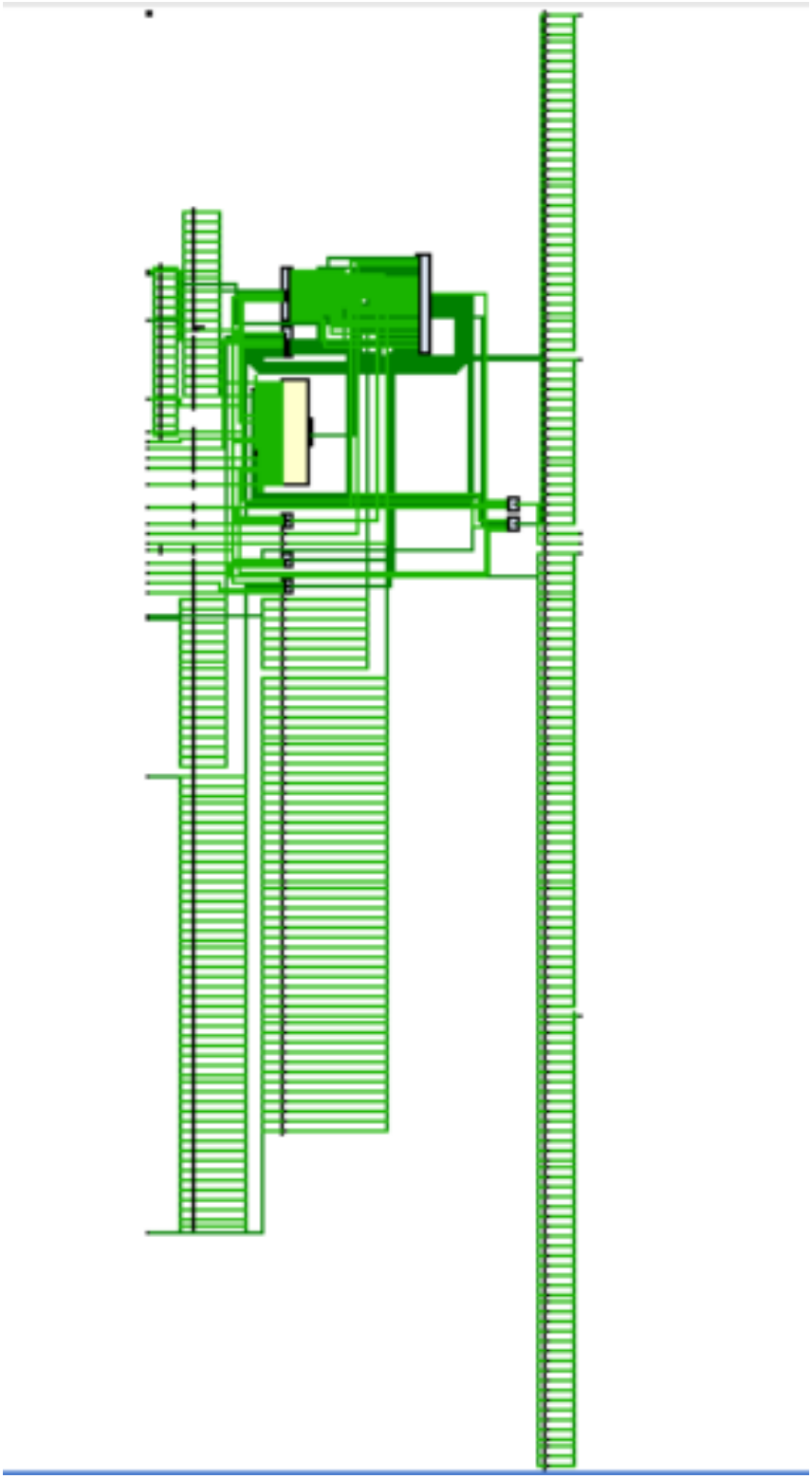
6.2 Synthesis

6.2.1 Massage tab

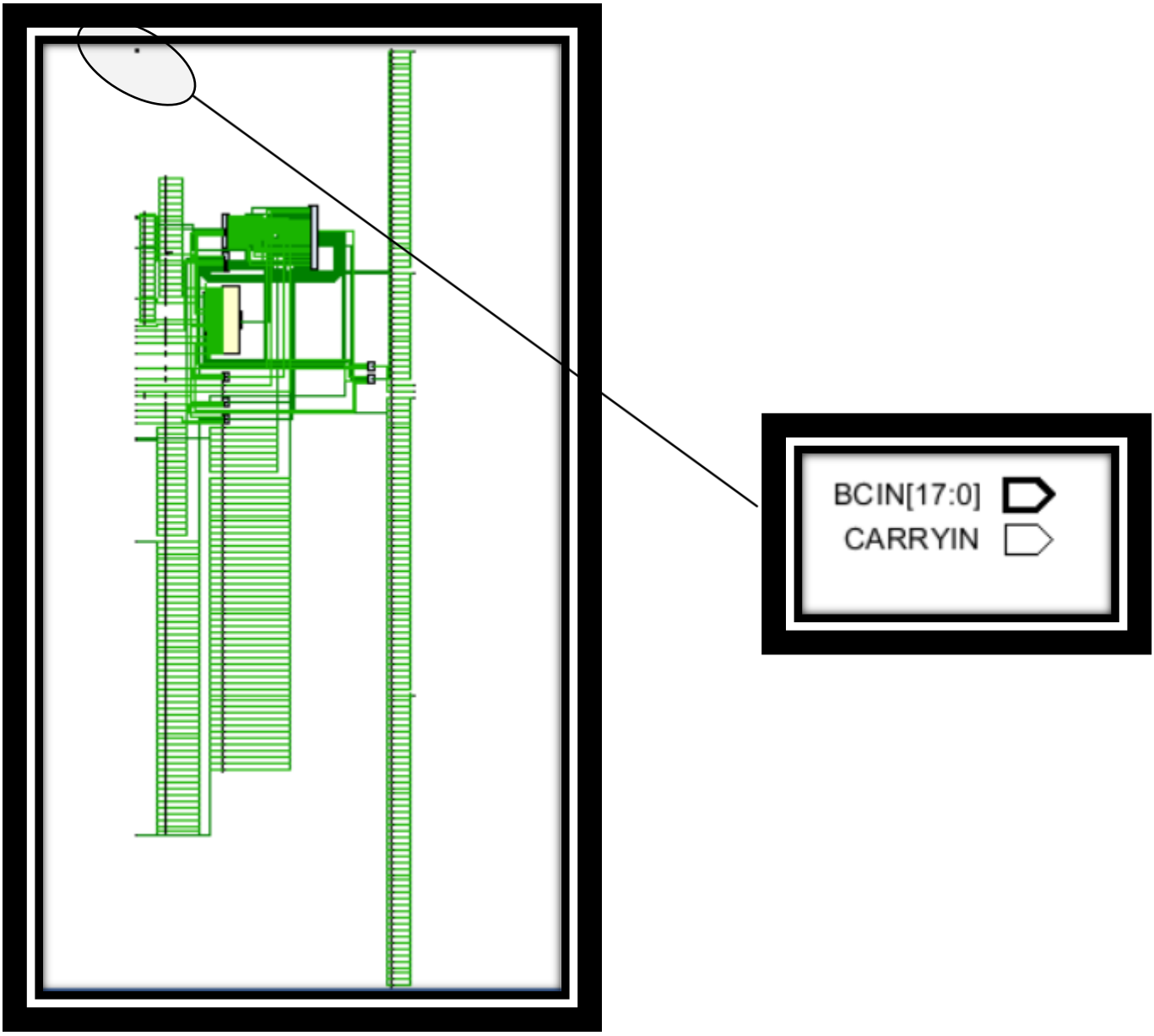


These warnings have shown as we didn’t use some input ports.

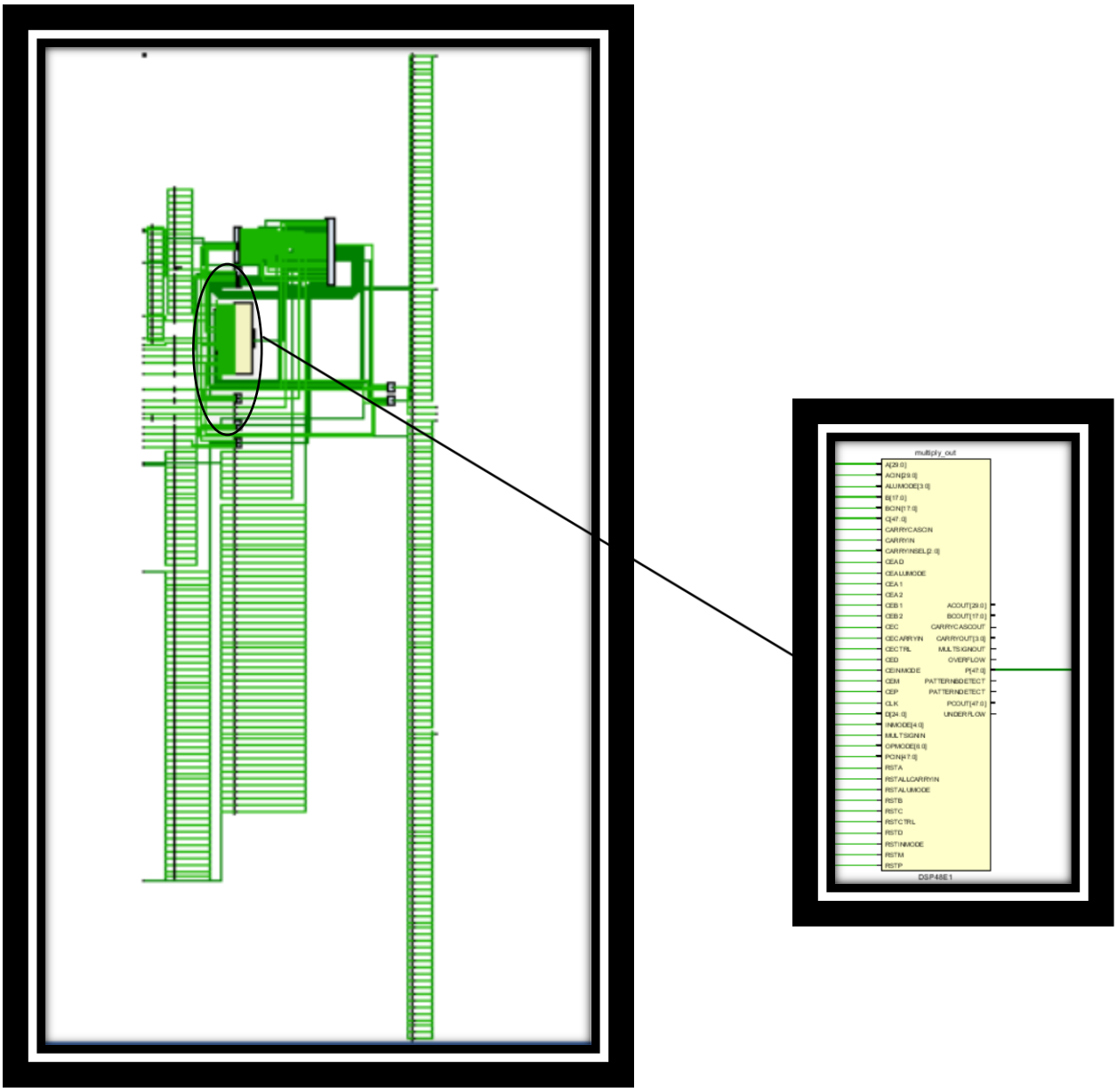
6.2.2 Schematic



Snippets

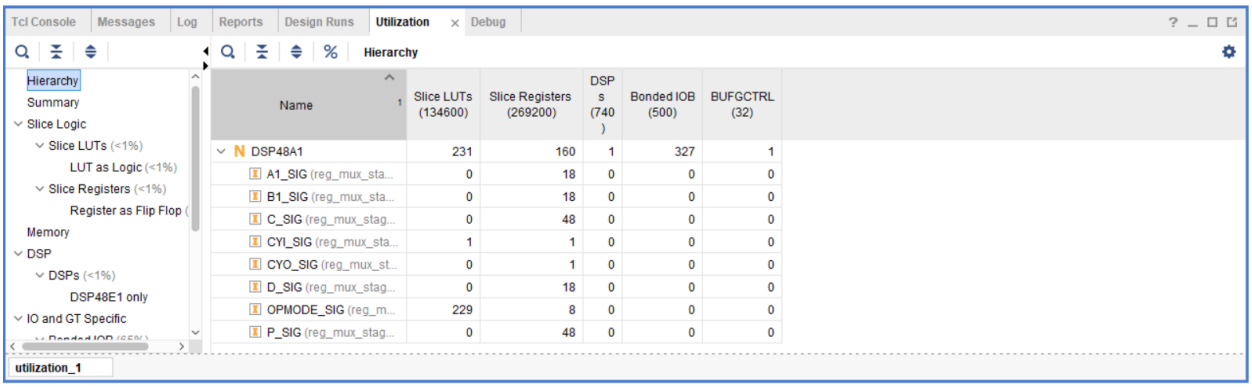


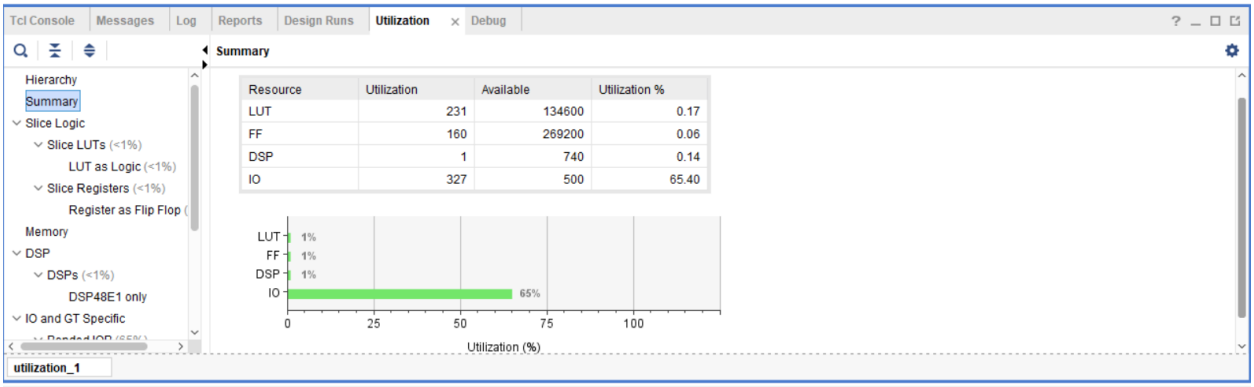
The unconnected inputs as the default design don’t use them



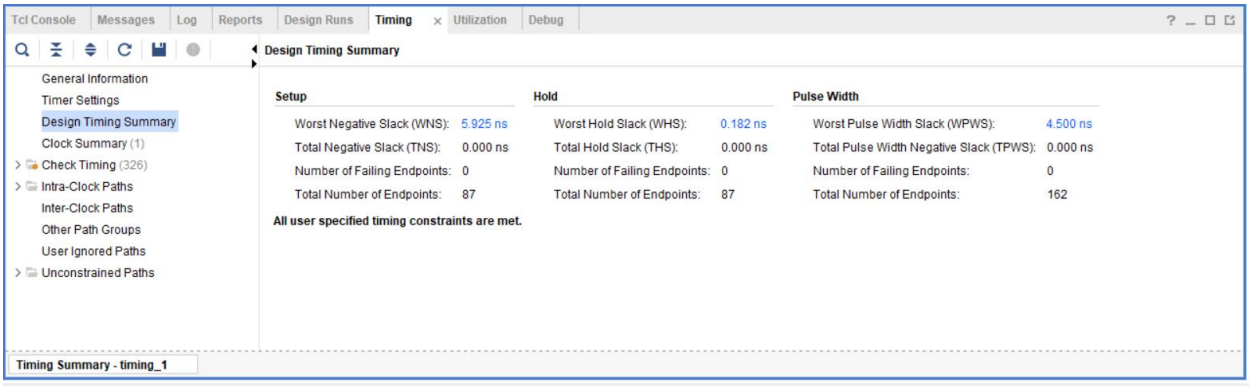
The remains blocks are the registers for the signals in the design.

6.2.3 Utilization Report



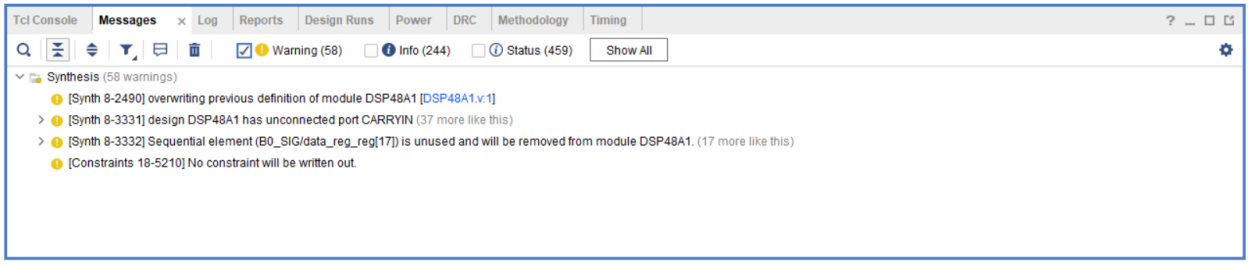


6.2.4 Timing Report



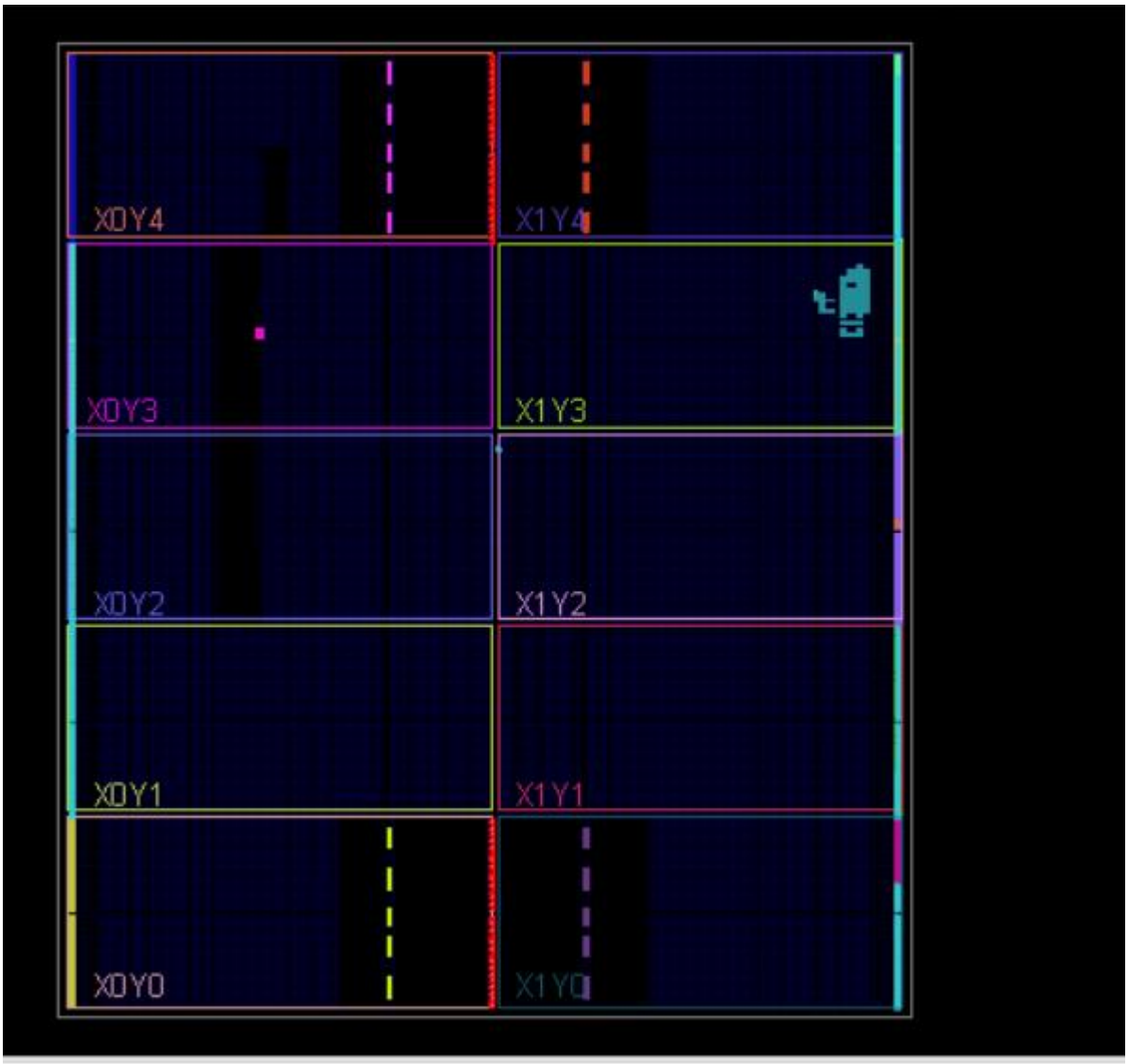
6.3 Implementation

6.3.1 Massage tab

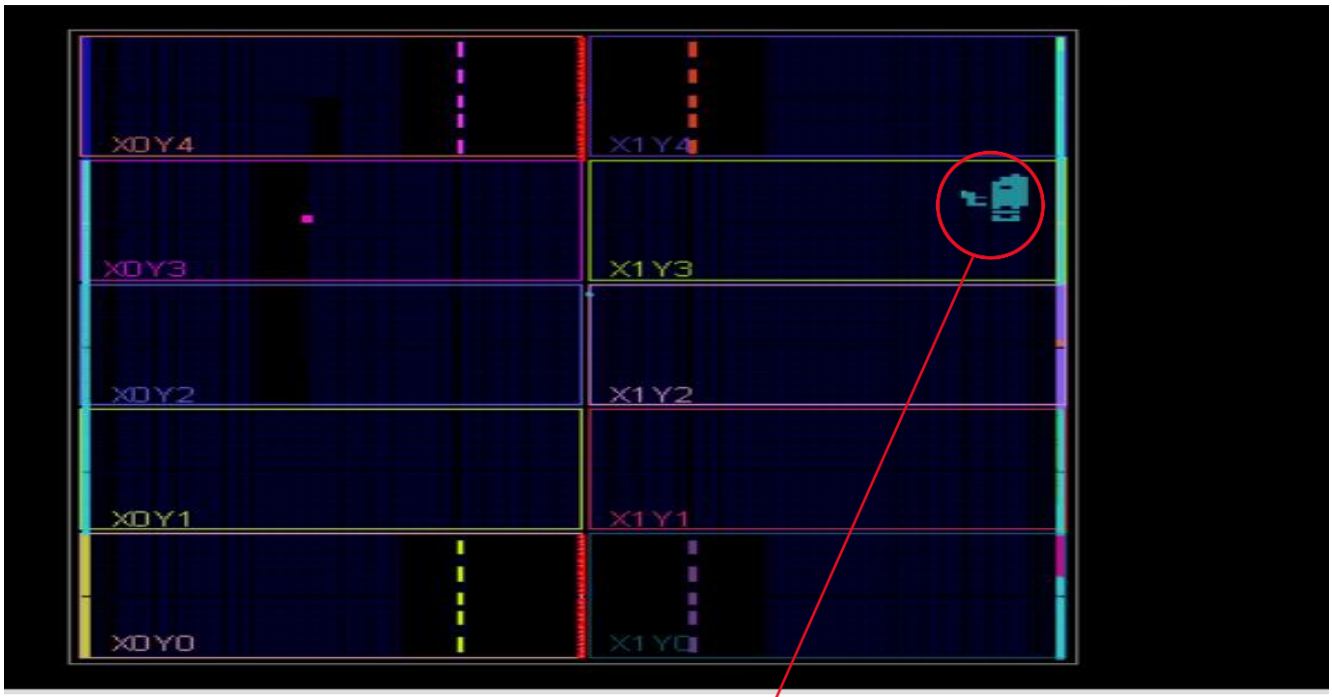


Nothing added.

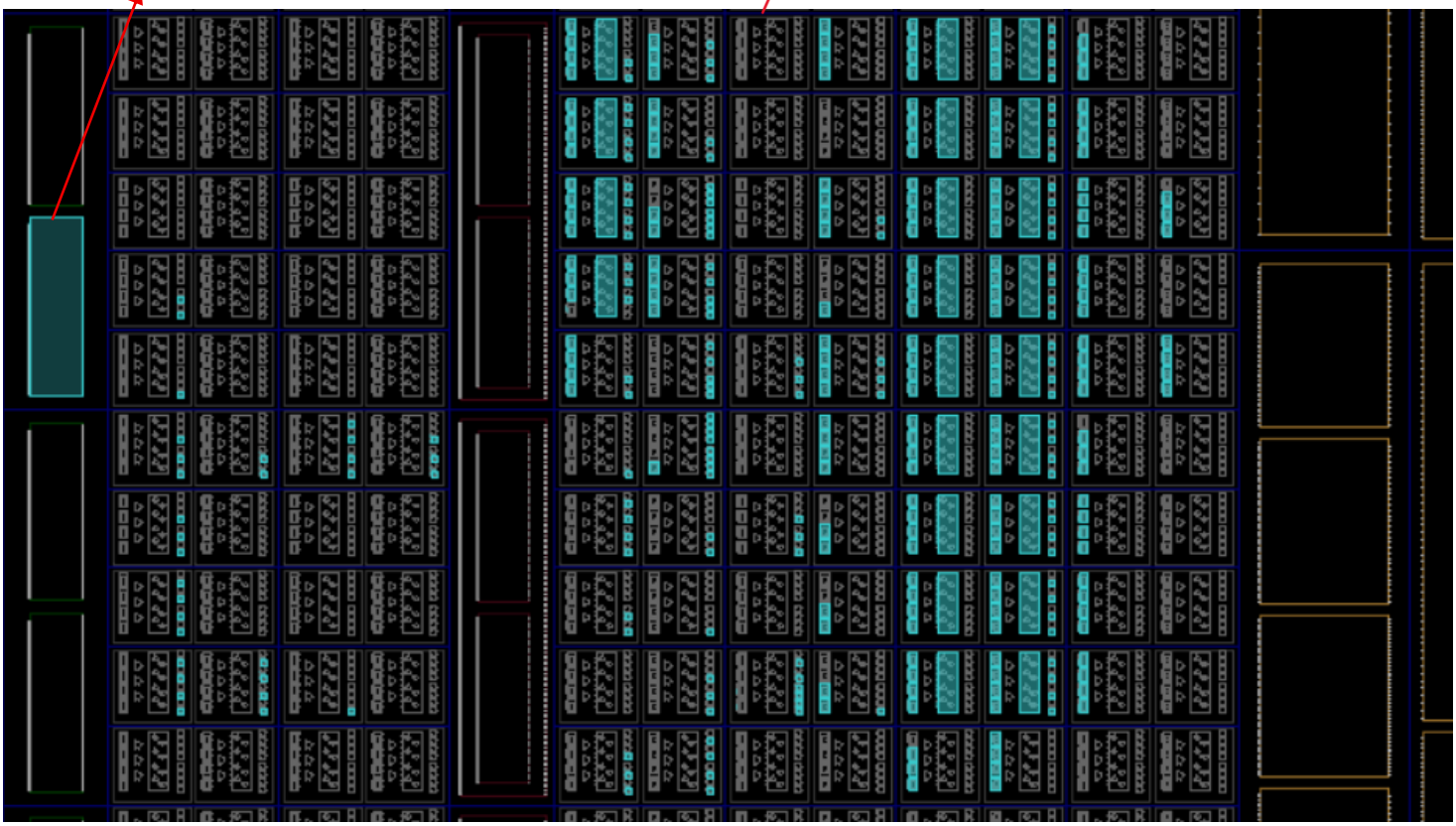
6.3.2 Device



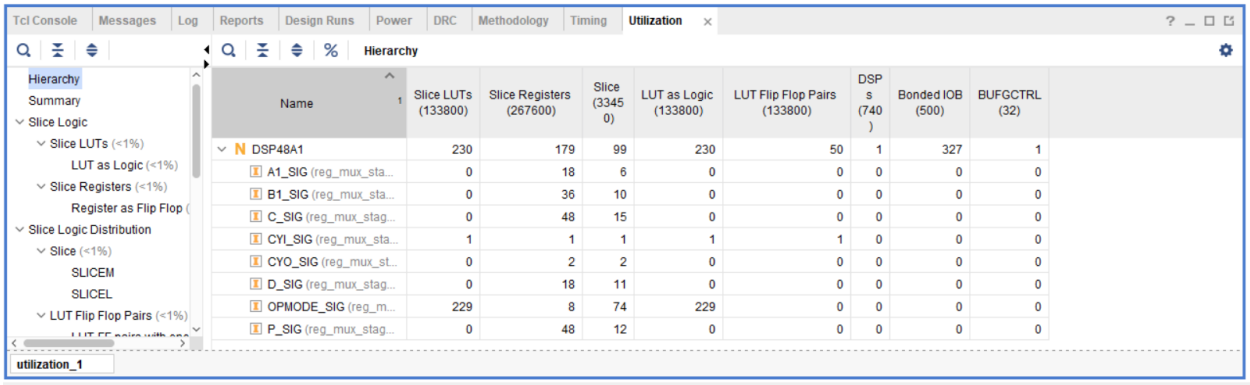
Snippets

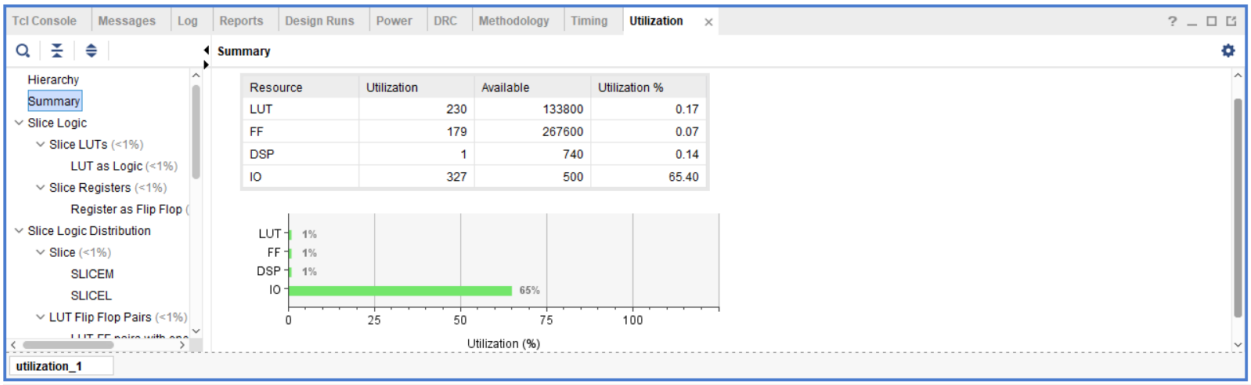


This is the DSP block used inside FPGA

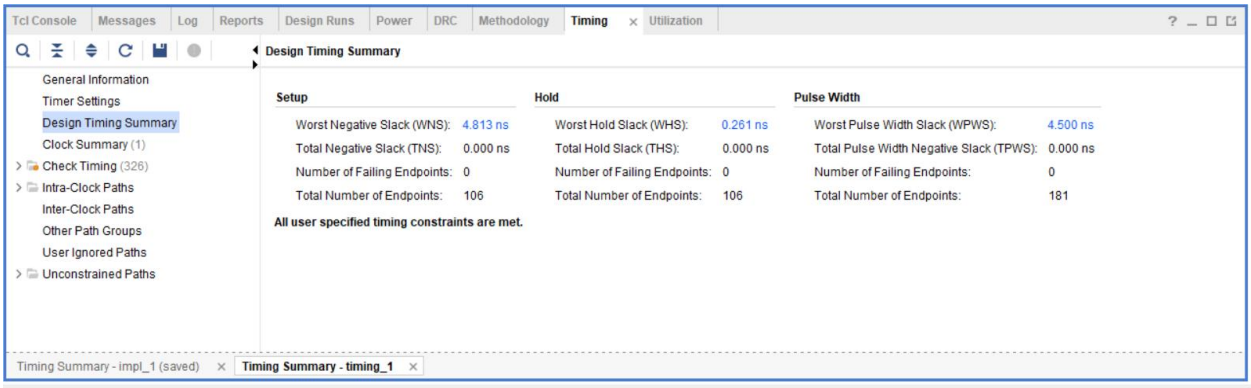


6.3.3 Utilization Report

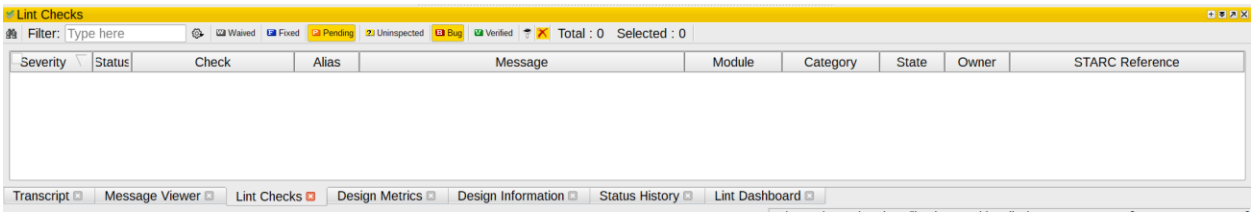


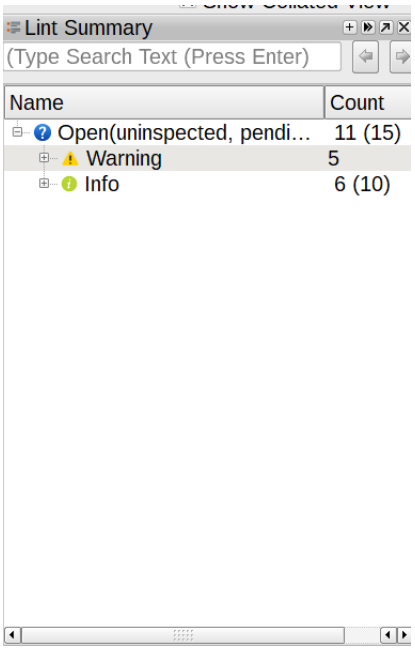


6.3.4 Timing Report



7 Linting





These warnings have shown as there is a fixed expressions in the muxes in my design but I can ignore these warnings as I intend to do this.