Project (1): Spartan6 - DSP48A1

Ahmed Saad Sweffi

Under the Supervision of Eng. Kareem Wassem

1 RTL Code for the Design and Mux

1.1 Mux Code

```
module reg_mux(rst, clk, ce, sel, in, out_mux);
    parameter reg_size = 18;
    parameter RST_TYPE = "SYNC";
    input rst, clk, ce, sel;
    input [reg_size-1:0] in;
    output wire [reg_size-1:0] out_mux;
    reg [reg_size-1:0] out_ff;

    generate
        if (RST_TYPE == "SYNC") begin
            always @(posedge clk) begin
                if (rst)
                    out_ff <= {reg_size{1'b0}};
                else if (ce)
                    out_ff <= in;
            end
        end else begin
            always @(posedge clk or posedge rst) begin
                if (rst)
                    out_ff <= {reg_size{1'b0}};
                else if (ce)
                    out_ff <= in;
            end
        end
    endgenerate

    assign out_mux = (sel) ? out_ff : in;
endmodule
```

1.2 DSP Design Code

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

    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;
```

```
16      parameter OPMODEREG = 1;
17      parameter CARRYINSEL = "OPMODE5";
18      parameter B_INPUT = "DIRECT";
19      parameter RSTTYPE = "SYNC";
20
21      input clk, CEA, CEB, CEC, CECARRYIN, CED, CEM, CEOPMODE, CEP;
22      input RSTA, RSTB, RSTC, RSTCARRYIN, RSTD, RSTM, RSTOPMODE, RSTP,
            CARRYIN;
23      input [7:0] opmode;
24      input [17:0] A, B, D, BCIN;
25      input [47:0] C, PCIN;
26      output CARRYOUT, CARRYOUTF;
27      output [17:0] BCOUT;
28      output [35:0] M;
29      output [47:0] P, PCOUT;
30
31      wire CYI, CIN, carryout_postsum;
32      wire [7:0] opmode_mux;
33      wire [17:0] A0, A1, B0, B1, D_mux, B_mux, PRE_SUM, B0_mux;
34      wire [35:0] prod;
35      wire [47:0] C_mux, conc_signal, post_sum;
36      reg [47:0] X_mux, Z_mux;
37
38      assign CYI = (CARRYINSEL == "OPMODE5") ? opmode_mux[5] : (
           CARRYINSEL == "CARRYIN") ? CARRYIN : 0;
39      assign B_mux = (B_INPUT == "DIRECT") ? B : (B_INPUT == "CASCADE
           ") ? BCIN : 0;
40      assign PRE_SUM = (opmode_mux[6]) ? D_mux- B0 : D_mux + B0;
41      assign B0_mux = (opmode_mux[4]) ? PRE_SUM : B0;
42      assign conc_signal = {D_mux[11:0], A1, B1};
43      assign prod = A1 * B1;
44      assign BCOUT = B1;
45      assign {carryout_postsum, post_sum} = (opmode_mux[7]) ? (Z_mux-
           (X_mux + CIN)) : (Z_mux + X_mux + CIN);
46      assign CARRYOUTF = CARRYOUT;
47      assign PCOUT = P;
48
49      reg_mux #(.reg_size(18), .RST_TYPE(RSTTYPE)) A0_REG (RSTA, clk,
           CEA, A0REG, A, A0);
50      reg_mux #(.reg_size(18), .RST_TYPE(RSTTYPE)) A1_REG (RSTA, clk,
           CEA, A1REG, A0, A1);
51      reg_mux #(.reg_size(18), .RST_TYPE(RSTTYPE)) B0_REG (RSTB, clk,
           CEB, A0REG, B_mux,B0);
52      reg_mux #(.reg_size(18), .RST_TYPE(RSTTYPE)) B1_REG (RSTB, clk,
           CEB, A1REG, B0_mux,B1);
53      reg_mux #(.reg_size(18), .RST_TYPE(RSTTYPE)) D_REG (RSTD, clk,
           CED, DREG, D, D_mux);
54      reg_mux #(.reg_size(48), .RST_TYPE(RSTTYPE)) C_REG (RSTC, clk,
           CEC, CREG, C, C_mux);
55      reg_mux #(.reg_size(8), .RST_TYPE(RSTTYPE)) OPMODE_REG (
           RSTOPMODE, clk, CEOPMODE,OPMODEREG, opmode, opmode_mux);
```

```
    reg_mux #(.reg_size(36), .RST_TYPE(RSTTYPE)) M_REG (RSTM, clk, CEM, MREG, prod, M);
    reg_mux #(.reg_size(1), .RST_TYPE(RSTTYPE)) CARRYIN_REG ( RSTCARRYIN, clk, CECARRYIN,CARRYINREG, CYI, CIN);
    reg_mux #(.reg_size(1), .RST_TYPE(RSTTYPE)) CARRYOUT_REG ( RSTCARRYOUT, clk,CECARRYOUT, CARRYOUTREG, carryout_postsum, CARRYOUT);
    reg_mux #(.reg_size(48), .RST_TYPE(RSTTYPE)) P_REG (RSTP, clk, CEP, PREG, post_sum, P);

    always @(*) begin
        case ({opmode_mux[1], opmode_mux[0]})
            2'b00: X_mux = 0;
            2'b01: X_mux = M;
            2'b10: X_mux = PCOUT;
            2'b11: X_mux = conc_signal;
        endcase
    end

    always @(*) begin
        case ({opmode_mux[3], opmode_mux[2]})
            2'b00: Z_mux = 0;
            2'b01: Z_mux = PCIN;
            2'b10: Z_mux = P;
            2'b11: Z_mux = C_mux;
        endcase
  end
endmodule
```

2 Simulation on Questa Sim

2.1 TestBench

```
module firstDSP_tb();
    reg clk, CEA, CEB, CEC, CECARRYIN, CED, CEM, CEOPMODE, CEP;
    reg RSTA, RSTB, RSTC, RSTCARRYIN, RSTD, RSTM, RSTOPMODE, RSTP, CARRYIN;
    reg [7:0] opmode;
    reg [17:0] A, B, D, BCIN;
    reg [47:0] C, PCIN;
    wire CARRYOUT, CARRYOUTF, CARRYOUT_2, CARRYOUTF_2;
    wire [17:0] BCOUT, BCOUT_2;
    wire [35:0] M, M2;
    wire [47:0] P, P_2, PCOUT, PCOUT_2;

  // Instantiate the first DUT
    firstDSP dut (
    .A(A), .B(B), .C(C), .D(D), .CARRYIN(CARRYIN), .clk(clk), .opmode(opmode), .BCIN(BCIN),
```

```
15          .CEA(CEA), .CEB(CEB), .CEC(CEC), .CECARRYIN(CECARRYIN), .CED
               (CED), .CEM(CEM), .CEOPMODE(CEOPMODE), .CEP(CEP),
16          .RSTA(RSTA), .RSTB(RSTB), .RSTC(RSTC), .RSTCARRYIN(
               RSTCARRYIN), .RSTD(RSTD), .RSTM(RSTM), .RSTOPMODE(
               RSTOPMODE), .RSTP(RSTP),
17          .BCOUT(BCOUT), .PCIN(PCIN), .PCOUT(PCOUT), .M(M), .P(P), .
               CARRYOUT(CARRYOUT), .CARRYOUTF(CARRYOUTF)
18      );
19
20  // Instantiate with modified parameters
21      firstDSP #(
22          .CARRYINSEL("CARRYIN"), .B_INPUT("CASCADE"), .RSTTYPE("ASYNC
               ")
23      ) dut_2 (
24          .A(A), .B(B), .C(C), .D(D), .CARRYIN(CARRYIN), .clk(clk), .
               opmode(opmode), .BCIN(BCIN),
25          .CEA(CEA), .CEB(CEB), .CEC(CEC), .CECARRYIN(CECARRYIN), .CED
               (CED), .CEM(CEM), .CEOPMODE(CEOPMODE), .CEP(CEP),
26          .RSTA(RSTA), .RSTB(RSTB), .RSTC(RSTC), .RSTCARRYIN(
               RSTCARRYIN), .RSTD(RSTD), .RSTM(RSTM), .RSTOPMODE(
               RSTOPMODE), .RSTP(RSTP),
27          .BCOUT(BCOUT_2), .PCIN(PCIN), .PCOUT(PCOUT_2), .M(M2), .P(
               P_2), .CARRYOUT(
28      CARRYOUT_2), .CARRYOUTF(CARRYOUTF_2)
29      );
30
31   // Clock
32      initial begin
33          clk = 1;
34          forever #2 clk = ~clk;
35      end
36
37      integer i;
38      initial begin
39      // reset
40          RSTA = 1; RSTB = 1; RSTC = 1; RSTCARRYIN = 1;
41          RSTD = 1; RSTM = 1; RSTOPMODE = 1; RSTP = 1;
42          for (i = 0; i < 10; i = i + 1) begin
43              @(negedge clk);
44              A = $random; B = $random; C = $random; D = $random; BCIN
                   = $random; PCIN =
45      $random; opmode = $random;
46          CARRYIN = $random; CEA = $random; CEB = $random; CEC =
               $random; CECARRYIN =
47      $random;
48          CED = $random; CEM = $random; CEP = $random; CEOPMODE =
               $random;
49          #5;
50          // Testing internal sync reset
51          if (dut.B1 != 0 || dut.A1 != 0 || dut.D_mux != 0 || dut.
               C_mux != 0 ||dut.opmode_mux != 0 || dut.M != 0 || dut.CIN
```

```
            != 0 || dut.CARRYOUT != 0|| dut.P != 0) begin
52              $display("Error in SYNCH RESET functionality");
53              $stop;
54          end
55      // Testing internal async reset
56          if (dut_2.B1 != 0 || dut_2.A1 != 0 || dut_2.D_mux != 0 ||
               dut_2.C_mux != 0||dut_2.opmode_mux != 0 || dut_2.M != 0
               || dut_2.CIN != 0 || dut_2.CARRYOUT != 0 || dut_2.P != 0)
                begin
57          $display("Error in ASYNCH RESET functionality");
58          $stop;
59          end
60      end
61      $display("RESET TEST PASSED");
62
63      // Deassert reset
64      RSTA = 0; RSTB = 0; RSTC = 0; RSTCARRYIN = 0;
65      RSTD = 0; RSTM = 0; RSTOPMODE = 0; RSTP = 0;
66      CEA = 0; CEB = 0; CEC = 0; CECARRYIN = 0; CED = 0; CEM = 0;
           CEOPMODE = 0; CEP =0;
67      for (i = 0; i < 10; i = i + 1) begin
68          @(negedge clk);
69          A = $random; B = $random; C = $random; D = $random; BCIN =
               $random; PCIN =
70      $random; opmode = $random;
71          CARRYIN = $random;
72          #5;
73          // Testing clock enable with sync reset
74          if (dut.B1 != 0 || dut.A1 != 0 || dut.D_mux != 0 || dut.
               C_mux != 0 ||dut.opmode_mux != 0 || dut.M != 0 || dut.CIN
                != 0 || dut.CARRYOUT != 0|| dut.P != 0) begin
75                  $display("Error in CLOCK ENABLE functionality");
76                  $stop;
77          end
78          // Testing clock enable with async reset
79              if (dut_2.B1 != 0 || dut_2.A1 != 0 || dut_2.D_mux != 0
                   || dut_2.C_mux != 0||dut_2.opmode_mux != 0 || dut_2.M
                    != 0 || dut_2.CIN != 0 || dut_2.CARRYOUT != 0 ||
                   dut_2.P != 0) begin
80                  $display("Error in CLOCK ENABLE functionality");
81                  $stop;
82              end
83          end
84          $display("CLOCK ENABLE TEST PASSED");
85
86          // Testing specific functional cases
87          @(negedge clk);
88          CEA = 1; CEB = 1; CEC = 1; CECARRYIN = 1; CED = 1; CEM = 1;
               CEOPMODE = 1; CEP =1;
89          A = 1; B = 2; C = 3; D = 4; opmode = 8'b00111101; BCIN = 5;
               CARRYIN = 0;
```

```
        repeat (5) @(negedge clk);

        opmode = 8'b00000011; BCIN = 10; CARRYIN = 1;
        repeat (5) @(negedge clk);

        opmode = 8'b10011010; BCIN = 5; CARRYIN = 0;
        repeat (5) @(negedge clk);

        opmode = 8'b10100101; BCIN = 10; CARRYIN = 0; PCIN = 100;
        repeat (10) @(negedge clk);

        $stop;
    end
endmodule
```

2.2 Do File

```
vsim -voptargs=+acc work.firstDSP_tb
add wave-position insertpoint \
sim:/firstDSP_tb/clk \
sim:/firstDSP_tb/CEA \
sim:/firstDSP_tb/CEB \
sim:/firstDSP_tb/CEC \
sim:/firstDSP_tb/CECARRYIN \
sim:/firstDSP_tb/CED \
sim:/firstDSP_tb/CEM \
sim:/firstDSP_tb/CEOPMODE \
sim:/firstDSP_tb/CEP \
sim:/firstDSP_tb/RSTA \
sim:/firstDSP_tb/RSTB \
sim:/firstDSP_tb/RSTC \
sim:/firstDSP_tb/RSTCARRYIN \
sim:/firstDSP_tb/RSTD \
sim:/firstDSP_tb/RSTM \
sim:/firstDSP_tb/RSTOPMODE \
sim:/firstDSP_tb/RSTP \
sim:/firstDSP_tb/CARRYIN \
sim:/firstDSP_tb/opmode \
sim:/firstDSP_tb/A \
sim:/firstDSP_tb/B \
sim:/firstDSP_tb/D \
sim:/firstDSP_tb/BCIN \
sim:/firstDSP_tb/C \
sim:/firstDSP_tb/PCIN \
sim:/firstDSP_tb/CARRYOUT \
sim:/firstDSP_tb/CARRYOUTF \
sim:/firstDSP_tb/CARRYOUT_2 \
sim:/firstDSP_tb/CARRYOUTF_2 \
sim:/firstDSP_tb/BCOUT \
sim:/firstDSP_tb/BCOUT_2 \
```

```
sim:/firstDSP_tb/M \
sim:/firstDSP_tb/M2 \
sim:/firstDSP_tb/P \
sim:/firstDSP_tb/P_2 \
sim:/firstDSP_tb/PCOUT \
sim:/firstDSP_tb/PCOUT_2 \
sim:/firstDSP_tb/i
run -all
```

3 Design Figures

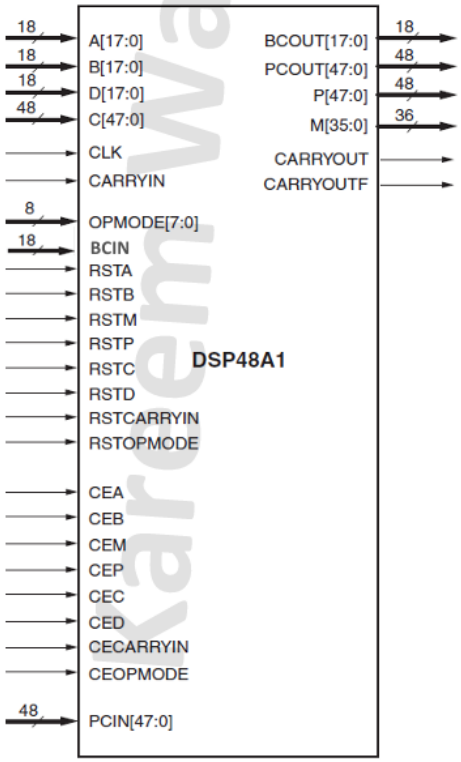
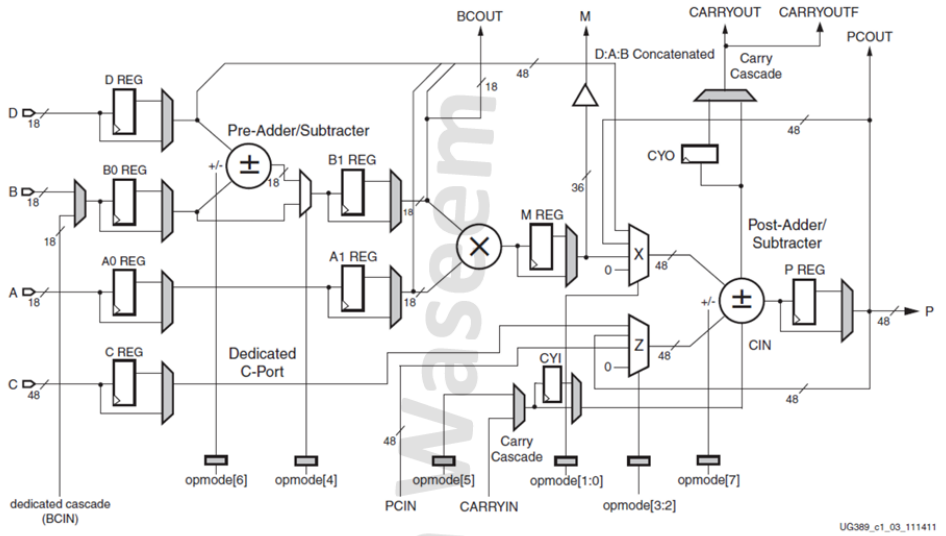


Figure 1: The DSP4A1.



Note: opmode input has a register and mux pair in the design entry the same way as the input A, D, or C

Figure 2: The Elaborated Design.

```
# RESET TEST PASSED
# CLOCK ENABLE TEST PASSED
# ** Note: $stop    : C:/Users/ahmed saad/Downloads/project 1/DSP_tb.v(101)
#    Time: 262 ns  Iteration: 1  Instance: /firstDSP_tb
# Break in Module firstDSP_tb at C:/Users/ahmed saad/Downloads/project 1/DSP_tb.v line 101

VSIM 6>
```

Figure 3: The messages bar shows no critical errors.

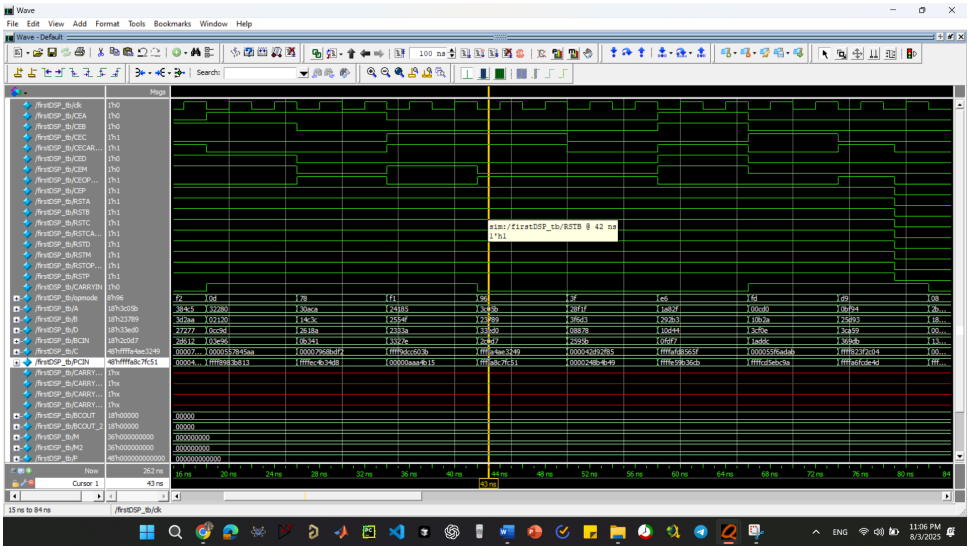


Figure 4: Wave form.

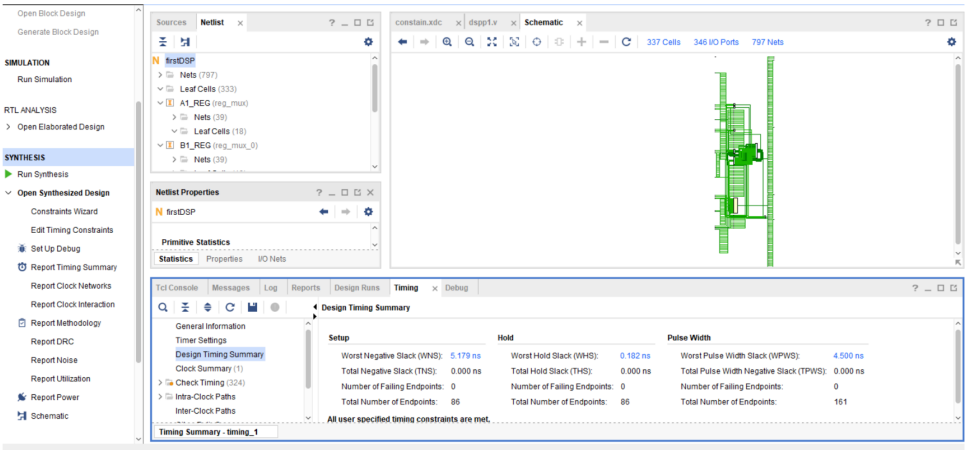


Figure 5: Report Timing Summary.

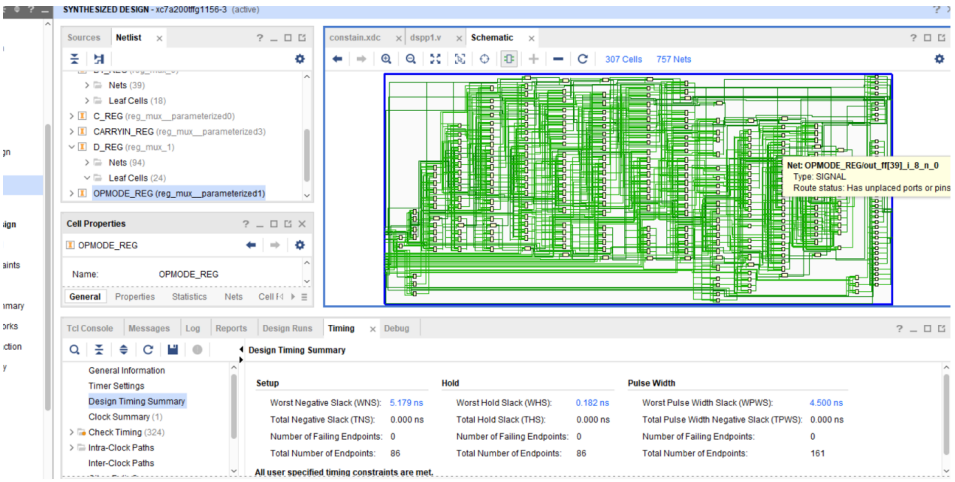


Figure 6: Inside the design of the Opmode.

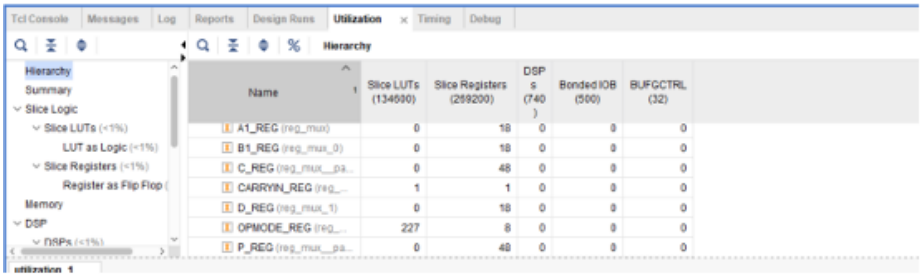


Figure 7: Utilization Report.

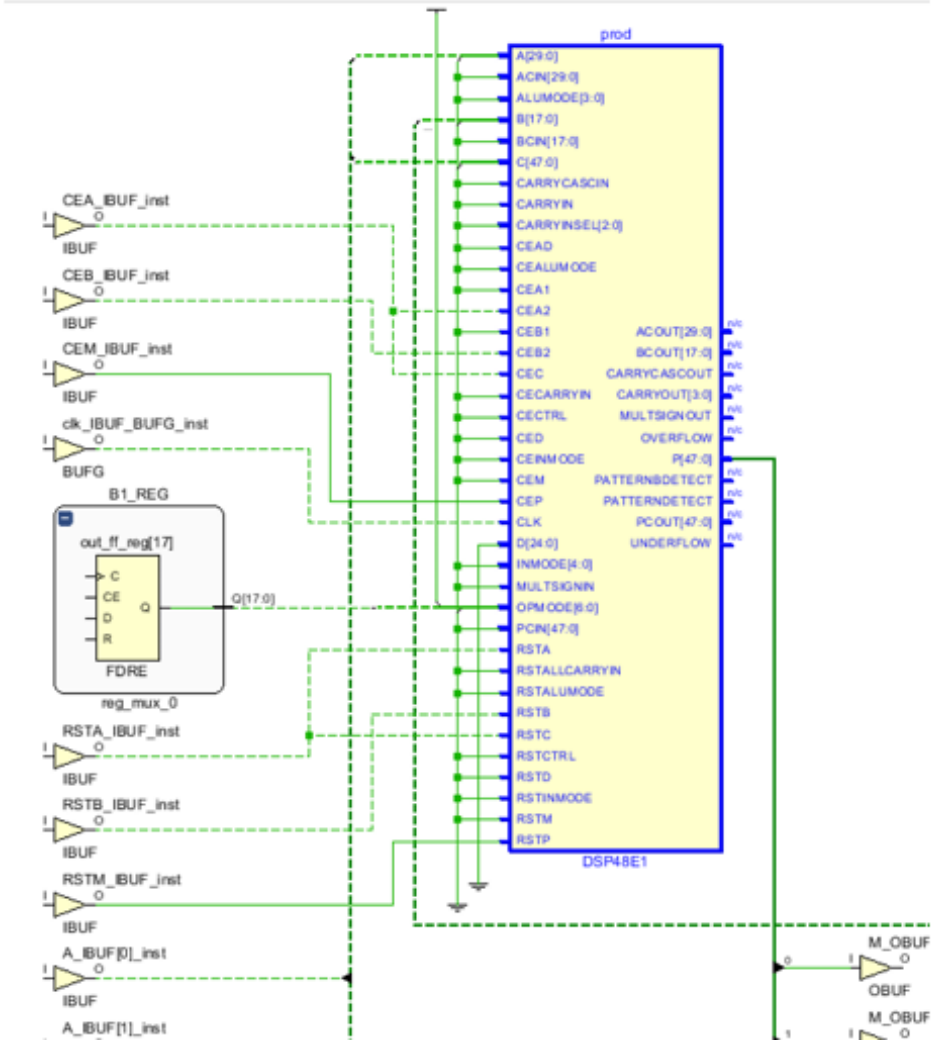


Figure 8: The DSP4A1.

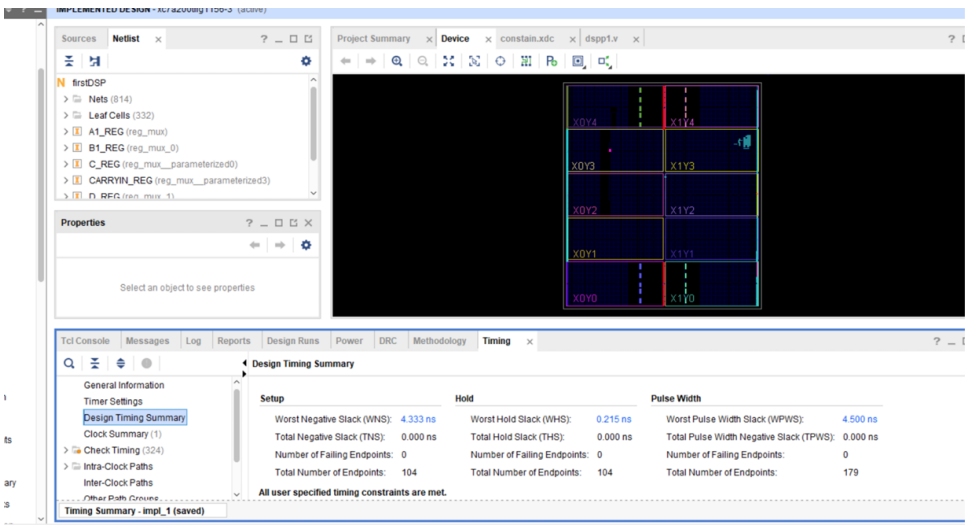


Figure 9: The Implementation Design after running it.

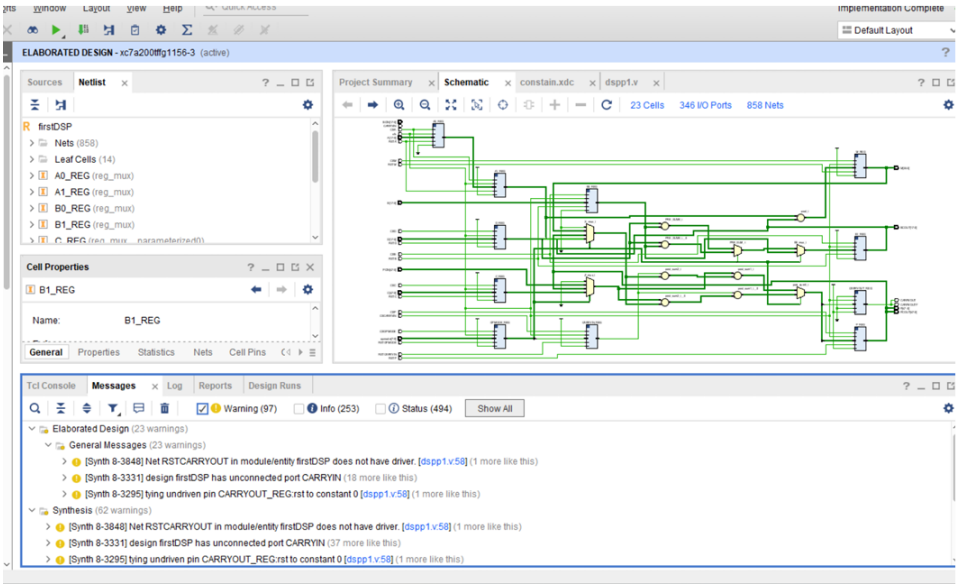


Figure 10: The Elaborated Design and messages.

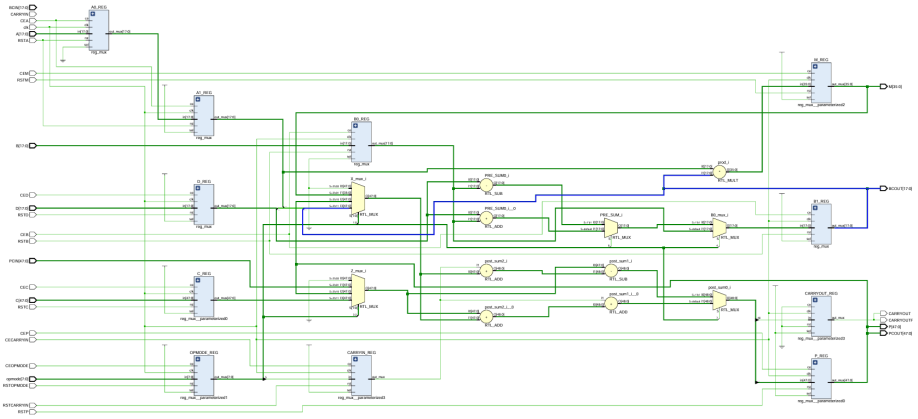


Figure 11: The Elaborated Design.