# Project (1)

Spartan6 - DSP48A1

Abdullah Mohammed Ahmed Mahmoud
**Under the Supervision**
Eng. Kareem Wassem

August 2024

# RTL Code For the Design and Mux

## 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
```

## 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;
    parameter OPMODEREG = 1;
    parameter CARRYINSEL = "OPMODE5";
    parameter B_INPUT = "DIRECT";
    parameter RSTTYPE = "SYNC";

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

    wire CYI, CIN, carryout_postsum;
```

```
    wire [7:0] opmode_mux;
    wire [17:0] A0, A1, B0, B1, D_mux, B_mux, PRE_SUM, B0_mux;
    wire [35:0] prod;
    wire [47:0] C_mux, conc_signal, post_sum;
    reg [47:0] X_mux, Z_mux;

    assign CYI = (CARRYINSEL == "OPMODE5") ? opmode_mux[5] : (CARRYINSEL == "CARRYIN") ? CARRYIN : 0;
    assign B_mux = (B_INPUT == "DIRECT") ? B : (B_INPUT == "CASCADE") ? BCIN : 0;
    assign PRE_SUM = (opmode_mux[6]) ? D_mux - B0 : D_mux + B0;
    assign B0_mux = (opmode_mux[4]) ? PRE_SUM : B0;
    assign conc_signal = {D_mux[11:0], A1, B1};
    assign prod = A1 * B1;
    assign BCOUT = B1;
    assign {carryout_postsum, post_sum} = (opmode_mux[7]) ? (Z_mux - (X_mux + CIN)) : (Z_mux + X_mux + CIN);
    assign CARRYOUTF = CARRYOUT;
    assign PCOUT = P;

    reg_mux #(.reg_size(18), .RST_TYPE(RSTTYPE)) A0_REG (RSTA, clk, CEA, A0REG, A, A0);
    reg_mux #(.reg_size(18), .RST_TYPE(RSTTYPE)) A1_REG (RSTA, clk, CEA, A1REG, A0, A1);
    reg_mux #(.reg_size(18), .RST_TYPE(RSTTYPE)) B0_REG (RSTB, clk, CEB, A0REG, B_mux, B0);
    reg_mux #(.reg_size(18), .RST_TYPE(RSTTYPE)) B1_REG (RSTB, clk, CEB, A1REG, B0_mux, B1);
    reg_mux #(.reg_size(18), .RST_TYPE(RSTTYPE)) D_REG (RSTD, clk, CED, DREG, D, D_mux);
    reg_mux #(.reg_size(48), .RST_TYPE(RSTTYPE)) C_REG (RSTC, clk, CEC, CREG, C, C_mux);
    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
```

## 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 with default parameters
    firstDSP dut (
        .A(A), .B(B), .C(C), .D(D), .CARRYIN(CARRYIN), .clk(clk), .opmode(opmode), .BCIN(BCIN),
        .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),
        .BCOUT(BCOUT), .PCIN(PCIN), .PCOUT(PCOUT), .M(M), .P(P), .CARRYOUT(CARRYOUT), .CARRYOUTF(CARRYOUTF)
    );

    // Instantiate the second DUT with modified parameters
    firstDSP #(
        .CARRYINSEL("CARRYIN"), .B_INPUT("CASCADE"), .RSTTYPE("ASYNC")
    ) dut_2 (
        .A(A), .B(B), .C(C), .D(D), .CARRYIN(CARRYIN), .clk(clk), .opmode(opmode), .BCIN(BCIN),
        .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),
        .BCOUT(BCOUT_2), .PCIN(PCIN), .PCOUT(PCOUT_2), .M(M2), .P(P_2), .CARRYOUT(CARRYOUT_2), .CARRYOUTF(CARRYOUTF_2)
    );

    // Clock generation
    initial begin
        clk = 1;
        forever #2 clk = ~clk;
    end

    integer i;

    // Initial block for reset and testing
    initial begin
        // Apply reset
        RSTA = 1; RSTB = 1; RSTC = 1; RSTCARRYIN = 1;
        RSTD = 1; RSTM = 1; RSTOPMODE = 1; RSTP = 1;
        for (i = 0; i < 10; i = i + 1) begin
            @(negedge clk);
            A = $random; B = $random; C = $random; D = $random; BCIN = $random; PCIN = $random; opmode = $random;
            CARRYIN = $random; CEA = $random; CEB = $random; CEC = $random; CECARRYIN = $random;
            CED = $random; CEM = $random; CEP = $random; CEOPMODE = $random;
            #5;
            // Testing internal synchronous reset
            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
                $display("Error in SYNCH RESET functionality");
                $stop;
            end
            // Testing internal asynchronous reset
            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
```

```
                $display("Error in ASYNCH RESET functionality");
                $stop;
            end
        end
        $display("RESET TEST PASSED");

        // Deassert reset
        RSTA = 0; RSTB = 0; RSTC = 0; RSTCARRYIN = 0;
        RSTD = 0; RSTM = 0; RSTOPMODE = 0; RSTP = 0;
        CEA = 0; CEB = 0; CEC = 0; CECARRYIN = 0; CED = 0; CEM = 0; CEOPMODE = 0; CEP = 0;
        for (i = 0; i < 10; i = i + 1) begin
            @(negedge clk);
            A = $random; B = $random; C = $random; D = $random; BCIN = $random; PCIN = $random; opmode = $random;
            CARRYIN = $random;
            #5;
            // Testing clock enable with synchronous reset
            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
                $display("Error in CLOCK ENABLE functionality");
                $stop;
            end
            // Testing clock enable with asynchronous reset
            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
                $display("Error in CLOCK ENABLE functionality");
                $stop;
            end
        end
        $display("CLOCK ENABLE TEST PASSED");

        // Testing specific functional cases
        @(negedge clk);
        CEA = 1; CEB = 1; CEC = 1; CECARRYIN = 1; CED = 1; CEM = 1; CEOPMODE = 1; CEP = 1;
        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
```

## Simulation on Questa Sim

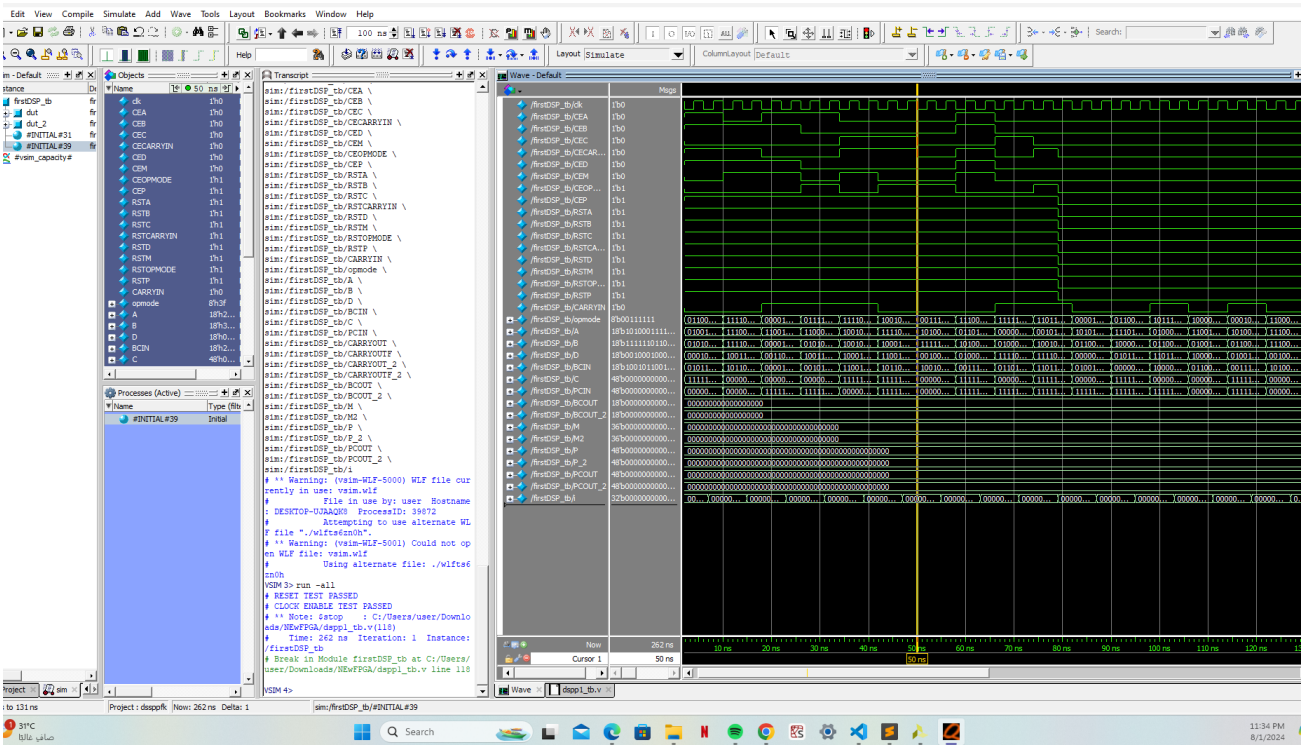


Figure 1: A snippet of the design's testbench simulation.

# 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
```

## Vivado Simulation and Synthesis for the DSP

I have added a constraint file with the following:

```
# Define a clock with a period of 10 ns (100 MHz) on pin W5
create_clock -period 10.0 [get_ports clk]

# Set the pin location and I/O standard for the clock
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
```

Then, I clicked on the run synthesis and also did a report timing summary for the design, here are the results:

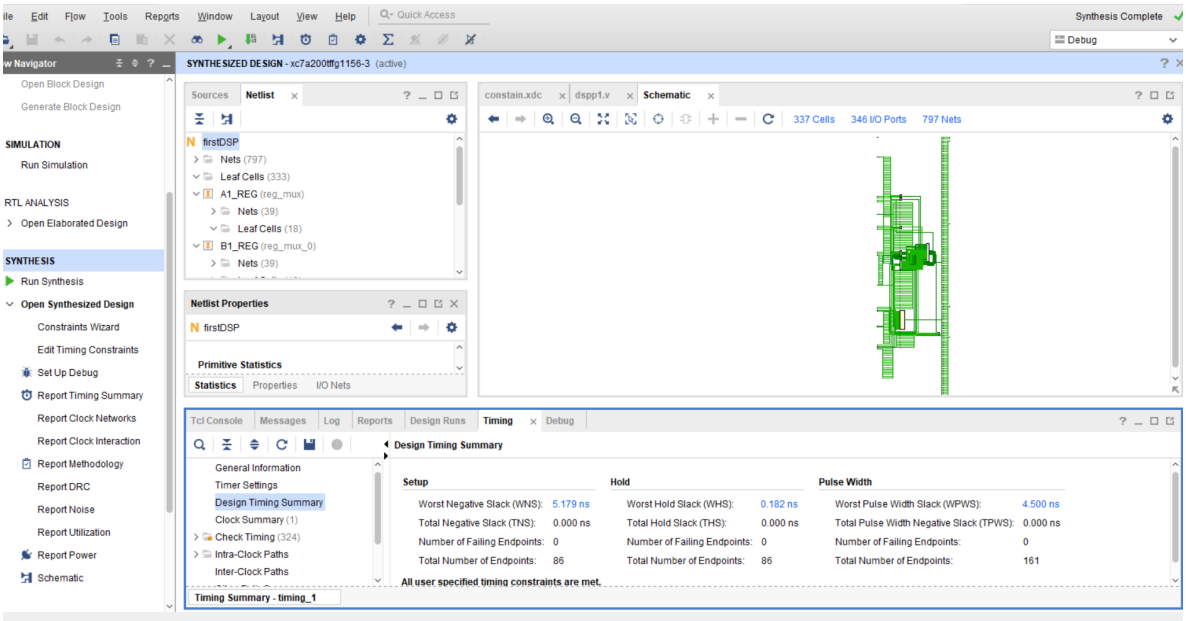



Figure 2: The Schematic of the design with the report timing summary.

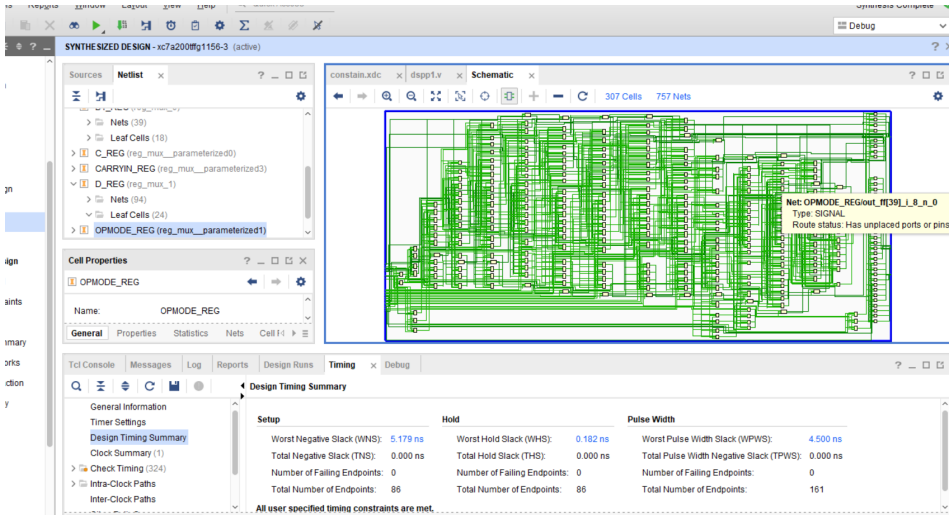



Figure 3: Inside the design of the opmode.

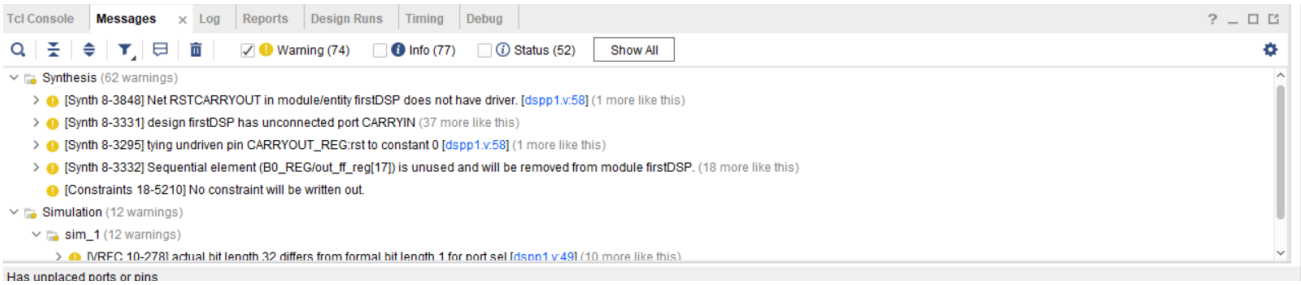
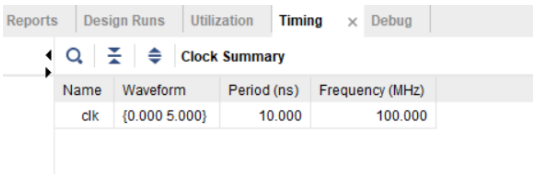
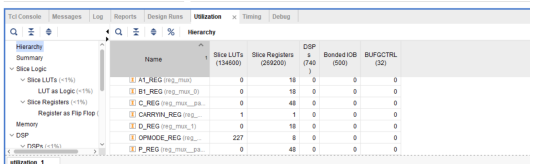



Figure 4: The messages bar shows no critical errors.




(a) Timing



(b) Utilization Report

Figure 5: The Utilization and Timing Before Implementation.

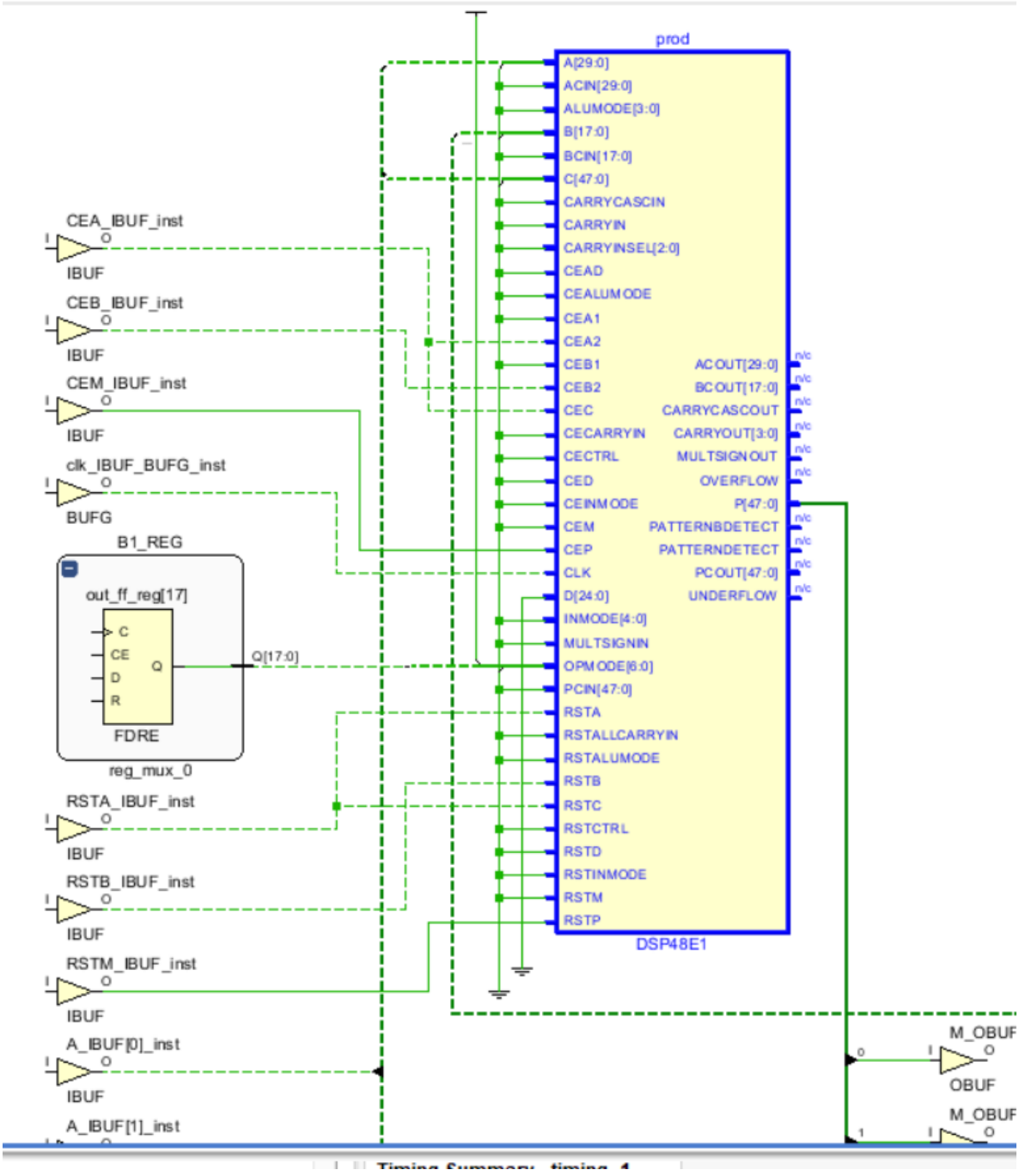



Figure 6: The DSP48A1

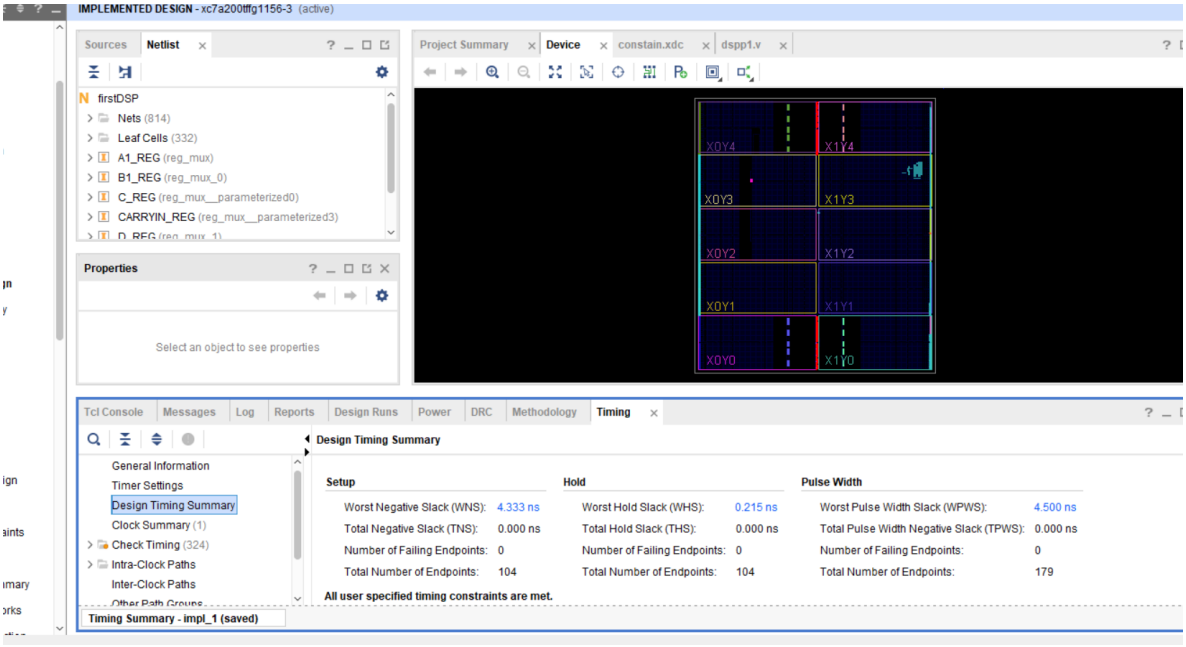



Figure 7: The Implementation Design after running it.

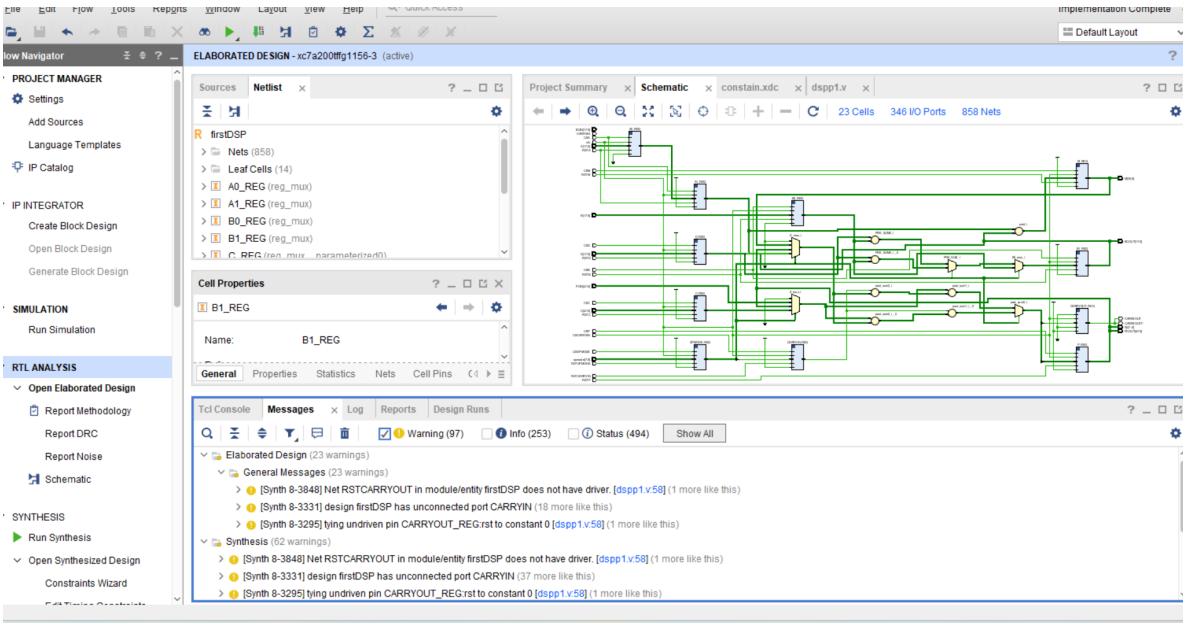



Figure 8: The Elaborated Design