# Spartan6 DSP48A1

Ameer Ashraf Louly – G3

# 1. RTL Code:

## 1.1 Mux Reg Module:

```
      module mux_reg(in, out, clk, ce, rst);

    parameter WIDTH = 18;
    parameter RSTTYPE = "SYNC";
    parameter REG = 0;
    input [WIDTH - 1 : 0] in;
    input   clk,
            ce,
            rst;
    output [WIDTH - 1 : 0] out;

    reg [WIDTH - 1 : 0] in_reg;

    generate
        if(REG == 1) begin
            if(RSTTYPE == "SYNC") begin
            always @(posedge clk) begin
                    if(rst)
                        in_reg <= 0;
                    else if(ce)
                        in_reg <= in;
                end
            end
            else if (RSTTYPE == "ASYNC") begin
                always @(posedge clk or posedge rst) begin
                    if(rst)
                        in_reg <= 0;
                    else if(ce)
                        in_reg <= in;
                end
            end
            assign out = in_reg;
        end
        else if (REG == 0)
           assign out = in;
    endgenerate
endmodule
```

## 1.2 DSP RTL:

```
module dsp48a1 #(
    parameter   A0REG = 0,
                A1REG = 1,
                B0REG = 0,
                B1REG = 1,
                CREG = 1,
                DREG = 1,
                MREG = 1,
                PREG = 1,
                CARRYINREG = 1,
                CARRYOUTREG = 1,
                OPMODEREG = 1,
                CARRYINSEL = "OPMODE5",
                B_INPUT = "DIRECT",
                RSTTYPE = "SYNC"
) (
    input [17 : 0]  A,
                    B,
                    D,
                    BCIN,
    input [7 : 0]   OPMODE,
    input [47 : 0]  C,
                    PCIN,
    input   CLK,
            CARRYIN,
            RSTA,
            RSTB,
            RSTM,
            RSTP,
            RSTC,
            RSTD,
            RSTCARRYIN,
            RSTOPMODE,
            CEA,
            CEB,
            CEM,
            CEP,
            CEC,
            CED,
            CECARRYIN,
            CEOPMODE,
    output  [17 : 0]    BCOUT,
    output  [47 : 0]    PCOUT,
                        P,
    output  [35 : 0]    M,
    output  CARRYOUT,
```

```
            CARRYOUTF
);
    wire [17 : 0]   A0_out,
                    D_out,
                    B0_in,
                    B0_out,
                    B1_in,
                    A1_out,
                    B1_out;
    reg [17 : 0]    pre_adder_subtractor_out;
    reg [47 : 0]    x_out,
                    z_out;
    wire [47 : 0]   C_out,
                    post_adder_subtractor_out,
                    P_in;
    wire [35 : 0]   multiplier_out,
                    M_out;
    wire [7 : 0]    opmode_out;
    wire    CYI_in,
            CYI_out,
            CYO_in;


    // Input Stage
    mux_reg #(  .REG(A0REG),
                .RSTTYPE(RSTTYPE),
                .WIDTH(18)) A0_REG (
                    .in(A),
                    .out(A0_out),
                    .clk(CLK),
                    .ce(CEA),
                    .rst(RSTA)
                );

    mux_reg #(  .REG(DREG),
                .RSTTYPE(RSTTYPE),
                .WIDTH(18)) D_REG (
                    .in(D),
                    .out(D_out),
                    .clk(CLK),
                    .ce(CED),
                    .rst(RSTD)
                );

    // assign B0_in = B;
    generate
        if(B_INPUT == "DIRECT") begin
            assign B0_in = B;
        end
```

```
        else if (B_INPUT == "CASCADE") begin
            assign B0_in = BCIN;
        end
        else begin
            assign B0_in = 0;
        end
    endgenerate

    mux_reg #(  .REG(B0REG),
                .RSTTYPE(RSTTYPE),
                .WIDTH(18)) B0_REG (
                    .in(B0_in),
                    .out(B0_out),
                    .clk(CLK),
                    .ce(CEB),
                    .rst(RSTB)
                );

    mux_reg #(  .REG(OPMODEREG),
                .RSTTYPE(RSTTYPE),
                .WIDTH(8)) OPMODE_REG (
                    .in(OPMODE),
                    .out(opmode_out),
                    .clk(CLK),
                    .ce(CEOPMODE),
                    .rst(RSTOPMODE)
                );

    mux_reg #(  .REG(CREG),
                .RSTTYPE(RSTTYPE),
                .WIDTH(48)) C_REG (
                    .in(C),
                    .out(C_out),
                    .clk(CLK),
                    .ce(CEC),
                    .rst(RSTC)
                );

    // Pre Adder Subtractor

    always @(*) begin
        if(opmode_out[6])
            pre_adder_subtractor_out = D_out - B0_out;
        else
            pre_adder_subtractor_out = D_out + B0_out;
    end

    assign B1_in = (opmode_out[4]) ? pre_adder_subtractor_out : B0_out;
```

```
mux_reg #(  .REG(A1REG),
            .RSTTYPE(RSTTYPE),
            .WIDTH(18)) A1_REG (
                .in(A0_out),
                .out(A1_out),
                .clk(CLK),
                .ce(CEA),
                .rst(RSTA)
            );

mux_reg #(  .REG(B1REG),
            .RSTTYPE(RSTTYPE),
            .WIDTH(18)) B1_REG (
                .in(B1_in),
                .out(B1_out),
                .clk(CLK),
                .ce(CEB),
                .rst(RSTB)
            );

assign BCOUT = B1_out; //! Might Need to Change Later
assign multiplier_out = B1_out * A1_out;

mux_reg #(  .REG(MREG),
            .RSTTYPE(RSTTYPE),
            .WIDTH(36)) M_REG (
                .in(multiplier_out),
                .out(M_out),
                .clk(CLK),
                .ce(CEM),
                .rst(RSTM)
            );

assign M = M_out;

//* Mux X
always @(*) begin
    case(opmode_out[1 : 0])
        2'b00: x_out = 0;
        2'b01: x_out = {12'b000000000000, M_out};
        2'b10: x_out = PCOUT;
        2'b11: x_out = {D_out[11:0], A1_out[17:0], B1_out[17:0]};
    endcase
end

//* Mux Z
always @(*) begin
```

```
        case(opmode_out[3 : 2])
            2'b00: z_out = 0;
            2'b01: z_out = PCIN;
            2'b10: z_out = PCOUT;
            2'b11: z_out = C_out;
        endcase
    end
    generate
        if(CARRYINSEL == "CARRYIN")
            assign CYI_in = CARRYIN;
        else if(CARRYINSEL == "OPMODE5")
            assign CYI_in = opmode_out[5];
    endgenerate
    mux_reg #(  .REG(CARRYINREG),
                .RSTTYPE(RSTTYPE),
                .WIDTH(1)) CYI_REG (
                    .in(CYI_in),
                    .out(CYI_out),
                    .clk(CLK),
                    .ce(CECARRYIN),
                    .rst(RSTCARRYIN)
                );
    assign {CYO_in, P_in} = (opmode_out[7]) ?
                            z_out - (x_out + CYI_out) :
                            x_out + z_out + CYI_out;
    mux_reg #(  .REG(PREG),
                .RSTTYPE(RSTTYPE),
                .WIDTH(48)) P_REG (
                    .in(P_in),
                    .out(P),
                    .clk(CLK),
                    .ce(CEP),
                    .rst(RSTP)
                );
    assign PCOUT = P;
    mux_reg #(  .REG(CARRYOUTREG),
                .RSTTYPE(RSTTYPE),
                .WIDTH(1)) CYO_REG (
                    .in(CYO_in),
                    .out(CARRYOUT),
                    .clk(CLK),
                    .ce(CECARRYIN),
                    .rst(RSTCARRYIN)
                );

    assign CARRYOUTF = CARRYOUT;
endmodule
```

## 2. Testbench Code:

```
module dsp48a1_tb();
    reg [17 : 0]    A,
                    B,
                    D,
                    BCIN;
    reg [7 : 0]     OPMODE;
    reg [47 : 0]    C,
                    PCIN;
    reg     CLK,
            CARRYIN,
            RSTA,
            RSTB,
            RSTM,
            RSTP,
            RSTC,
            RSTD,
            RSTCARRYIN,
            RSTOPMODE,
            CEA,
            CEB,
            CEM,
            CEP,
            CEC,
            CED,
            CECARRYIN,
            CEOPMODE;
    wire  [17 : 0]    BCOUT;
    wire  [47 : 0]  PCOUT,
                    P;
    wire  [35 : 0]    M;
    wire    CARRYOUT,
            CARRYOUTF;

    dsp48a1 DUT(.A(A),
                .B(B),
                .D(D),
                .BCIN(BCIN),
                .OPMODE(OPMODE),
                .C(C),
                .PCIN(PCIN),
                .CLK(CLK),
                .CARRYIN(CARRYIN),
                .RSTA(RSTA),
                .RSTB(RSTB),
                .RSTM(RSTM),
                .RSTP(RSTP),
```

```
            .RSTC(RSTC),
            .RSTD(RSTD),
            .RSTCARRYIN(RSTCARRYIN),
            .RSTOPMODE(RSTOPMODE),
            .CEA(CEA),
            .CEB(CEB),
            .CEM(CEM),
            .CEP(CEP),
            .CEC(CEC),
            .CED(CED),
            .CECARRYIN(CECARRYIN),
            .CEOPMODE(CEOPMODE),
            .BCOUT(BCOUT),
            .PCOUT(PCOUT),
            .P(P),
            .M(M),
            .CARRYOUT(CARRYOUT),
            .CARRYOUTF(CARRYOUTF));

    initial begin
        CLK = 0;
        forever begin
            #1 CLK = ~CLK;
        end
    end

    initial begin
        // Testing Reset
        RSTA = 1;
        RSTB = 1;
        RSTM = 1;
        RSTP = 1;
        RSTC = 1;
        RSTD = 1;
        RSTCARRYIN = 1;
        RSTOPMODE = 1;
        repeat(100) begin
            A = $random;
            B = $random;
            D = $random;
            BCIN = $random;
            OPMODE = $random;
            C = $random;
            PCIN = $random;
            CLK = $random;
            CARRYIN = $random;
            CEA = $random;
            CEB = $random;
```

```
        CEM = $random;
        CEP = $random;
        CEC = $random;
        CED = $random;
        CECARRYIN = $random;
        CEOPMODE = $random;

        @(negedge CLK);

        if (BCOUT     != 0 ||
            PCOUT     != 0 ||
            P         != 0 ||
            M         != 0 ||
            CARRYOUT  != 0 ||
            CARRYOUTF != 0) begin
            $display("ERROR - Reset");
            $exit;
        end
    end // End of Reset Test Loop

    RSTA = 0;
    RSTB = 0;
    RSTM = 0;
    RSTP = 0;
    RSTC = 0;
    RSTD = 0;
    RSTCARRYIN = 0;
    RSTOPMODE = 0;

    CEA        = 1;
    CEB        = 1;
    CEM        = 1;
    CEP        = 1;
    CEC        = 1;
    CED        = 1;
    CECARRYIN  = 1;
    CEOPMODE   = 1;

    OPMODE = 8'b11011101;
    A = 20;
    B = 10;
    C = 350;
    D = 25;
    repeat(100) begin
        BCIN     = $random;
        PCIN     = $random;
        CARRYIN  = $random;
        repeat(4) @(negedge CLK);
```

```
        if (BCOUT       != 'hf      &&
            M           != 'h12c    &&
            P           != 'h32     &&
            PCOUT       != 'h32     &&
            CARRYOUT    != 0        &&
            CARRYOUTF   != 0) begin
            $display("ERROR - DSP Path 1");
            $exit;
        end
    end // End of DSP Path 1 Loop

    OPMODE = 8'b00010000;
    A      = 20;
    B      = 10;
    C      = 350;
    D      = 25;

    repeat(100) begin
        BCIN     = $random;
        PCIN     = $random;
        CARRYIN  = $random;

        repeat(3) @(negedge CLK); // Wait for DREG, B1REG, MREG propagation

        if (
            BCOUT       != 'h23    &&
            M           != 'h2bc   &&
            P           != 0       &&
            PCOUT       != 0       &&
            CARRYOUT    != 0       &&
            CARRYOUTF   != 0
        )begin
            $display("ERROR - DSP Path 2");
            $exit;
        end
    end // End of DSP Path 2 Loop

    // DSP Path 3 – No Pre-addition, P Feedback through Mux X and Z
    OPMODE = 8'b00001010;
    A      = 20;
    B      = 10;
    C      = 350;
    D      = 25;

    repeat(100) begin
        BCIN     = $random;
        PCIN     = $random;
```

```
            CARRYIN  = $random;

            repeat(3) @(negedge CLK); // Wait for B1REG, MREG, PREG propagation

            if (
                BCOUT      != 'ha   &&
                M          != 'hc8  &&
                P          != PCOUT &&
                CARRYOUT   != CARRYOUTF
            ) begin
                $display("ERROR - DSP Path 3");
                $exit;
            end
        end // End of DSP Path 3 Loop

        // DSP Path 4 – Post-subtraction with D:A:B concatenation and PCIN routing
        OPMODE = 8'b10100111;
        A      = 5;
        B      = 6;
        C      = 350;
        D      = 25;
        PCIN   = 3000;

        repeat(100) begin
            BCIN     = $random;
            CARRYIN  = $random;

            repeat(3) @(negedge CLK); // Wait for B1REG, MREG, PREG propagation

            if (
                BCOUT      != 'h6                   &&
                M          != 'h1e                  &&
                P          != 'hfe6fffec0bb1        &&
                PCOUT      != 'hfe6fffec0bb1        &&
                CARRYOUT   != 1                     &&
                CARRYOUTF  != 1
            ) begin
                $display("ERROR - DSP Path 4");
                $exit;
            end
        end // End of DSP Path 4 Loop



        $display("Test Successful");
        $exit;
    end
endmodule
```

## 3. Do File:

```
vlib work

vlog DSP48A1.v  DSP48A1_tb.v

vsim -voptargs=+acc work.dsp48a1_tb

add wave *

run -all

quit -sim
```

# 4. Simulation Snippet:

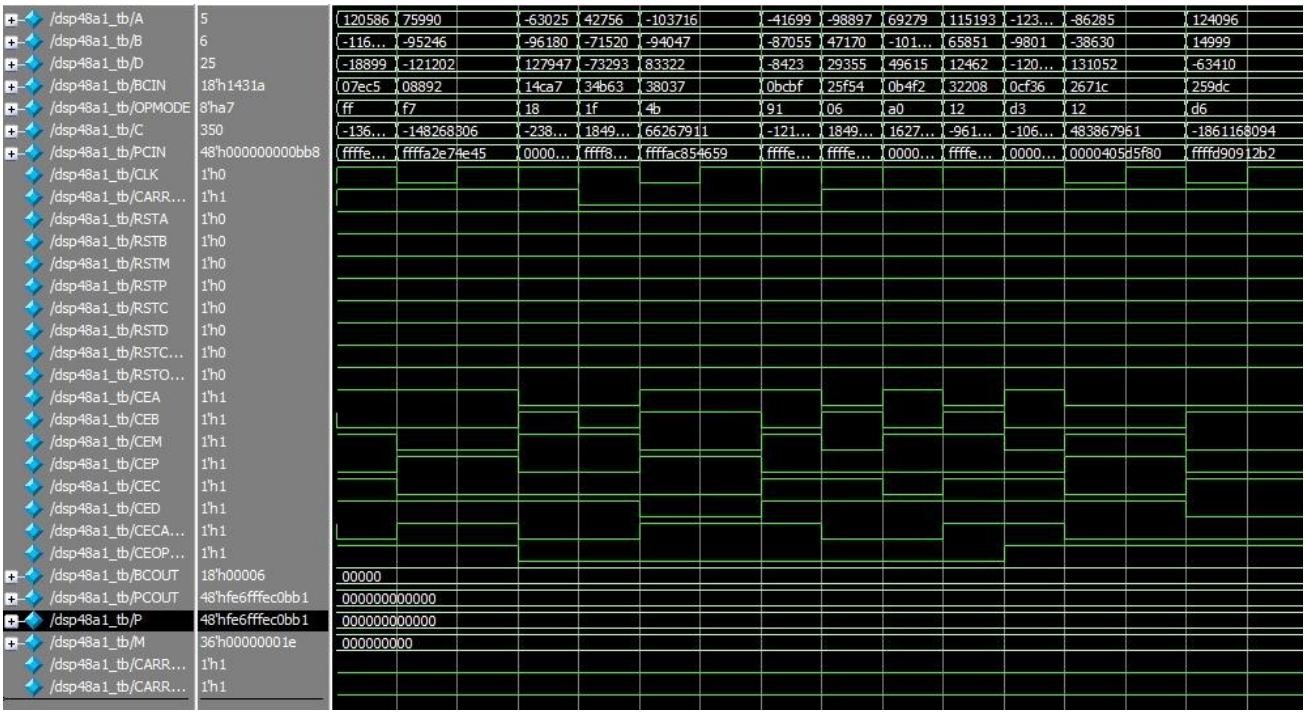


*Figure 1 Reset Test*

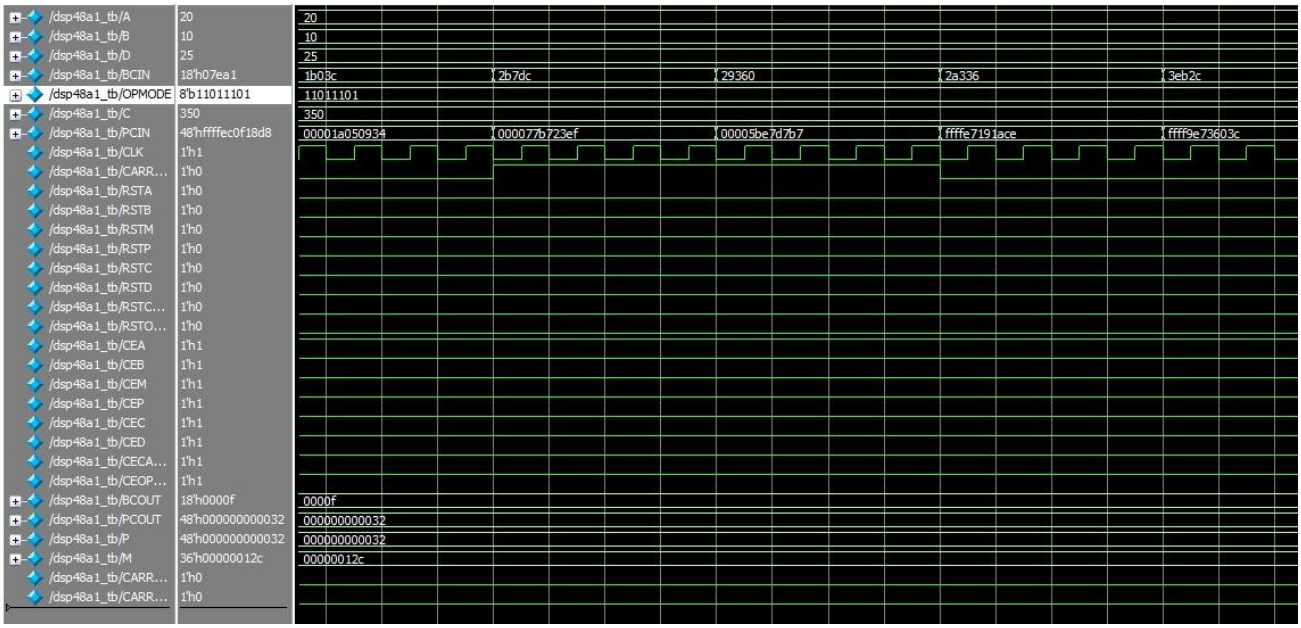


*Figure 2 DSP Path 1*

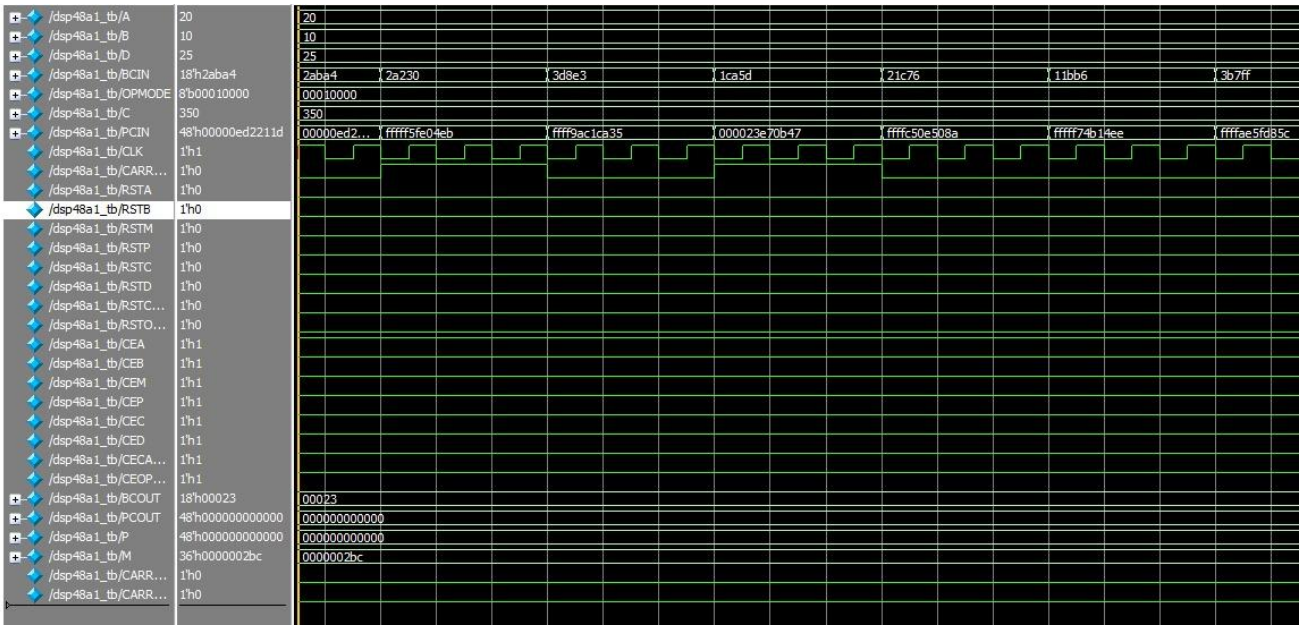


*Figure 3 DSP Path 2*

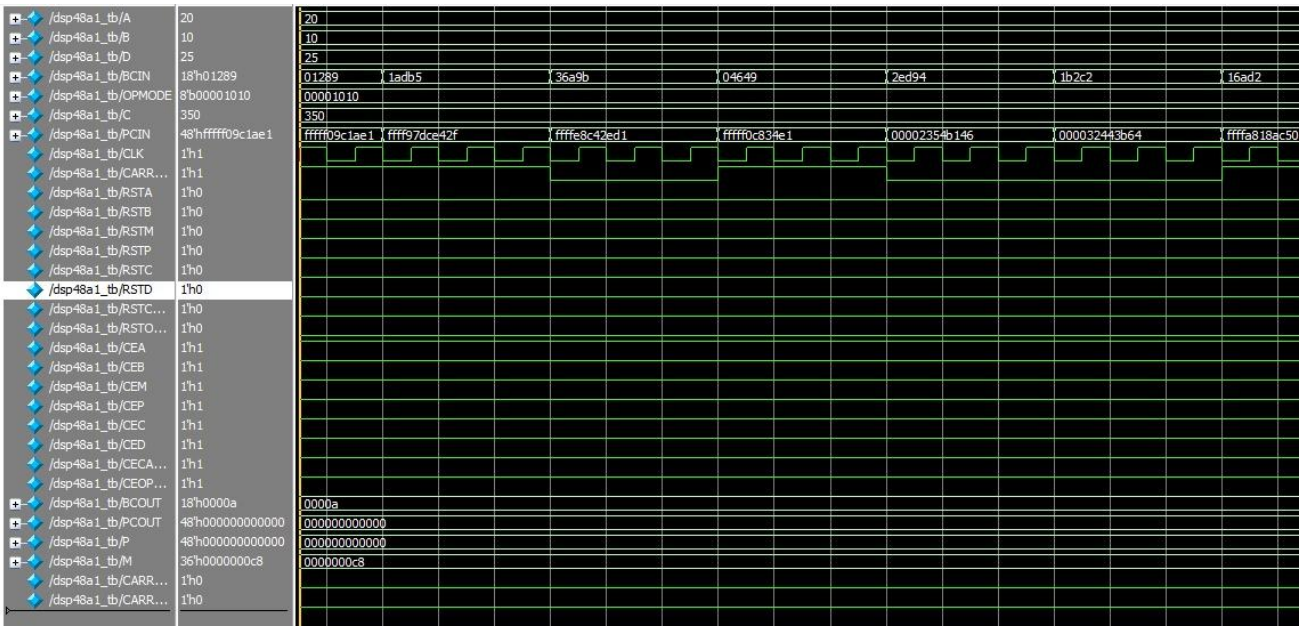


*Figure 4 DSP Path 3*

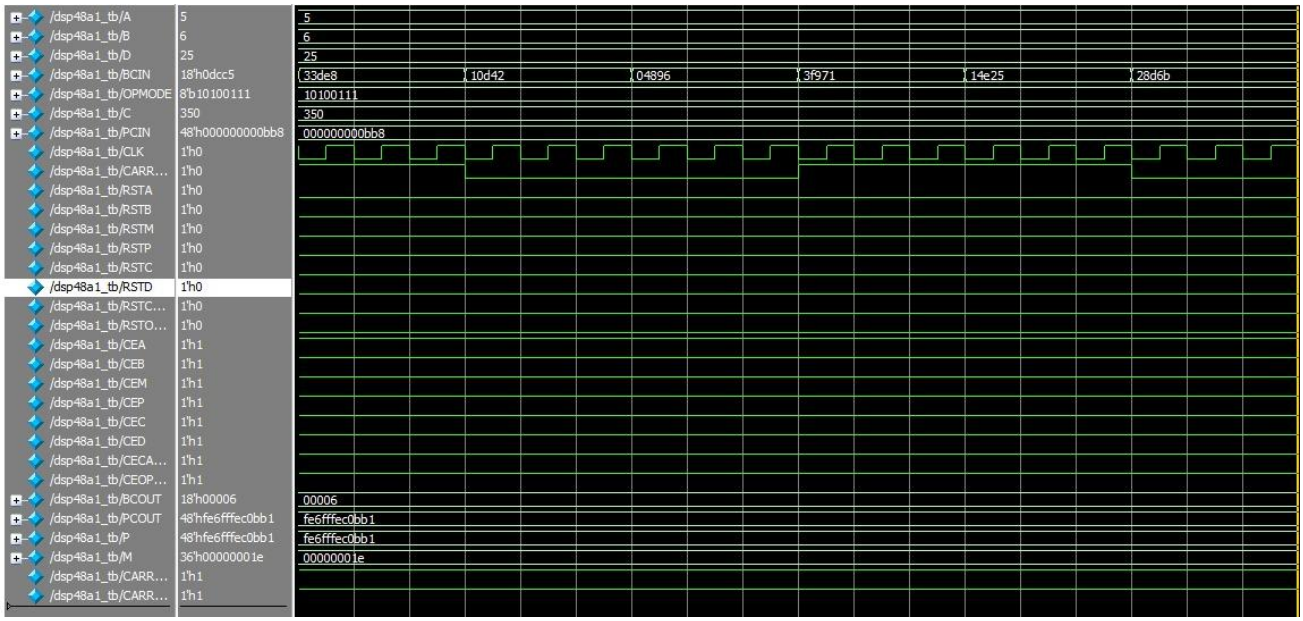


*Figure 5 DSP Path 4*

# 5. Linting:

| Severity | Status | Check | Alias | Message | Module | Category | State | Owner | STARC Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | parameter_name_duplicate | | Same parameter name is used in more than one mod… | dsp48a1 | Nomenclature… | open | unassig… | 1.1.4.3, 3.2.2.2 |
| | | multi_ports_in_single_line | | Multiple ports are declared in one line. Module mux_r… | mux_reg | Rtl Design Style | open | unassig… | 3.5.6.3 |

*Figure 6 Linter Showing no Critical Errors*

# 6. Constraint File:

```
## Clock signal
set_property -dict {PACKAGE_PIN W5 IOSTANDARD LVCMOS33} [get_ports CLK]
create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_ports CLK]
```

*Figure 7 Setting Clock Signal through Constraint File, Nothing else was edited*

# 7. Vivado Steps:

## 7.1 Elaboration:

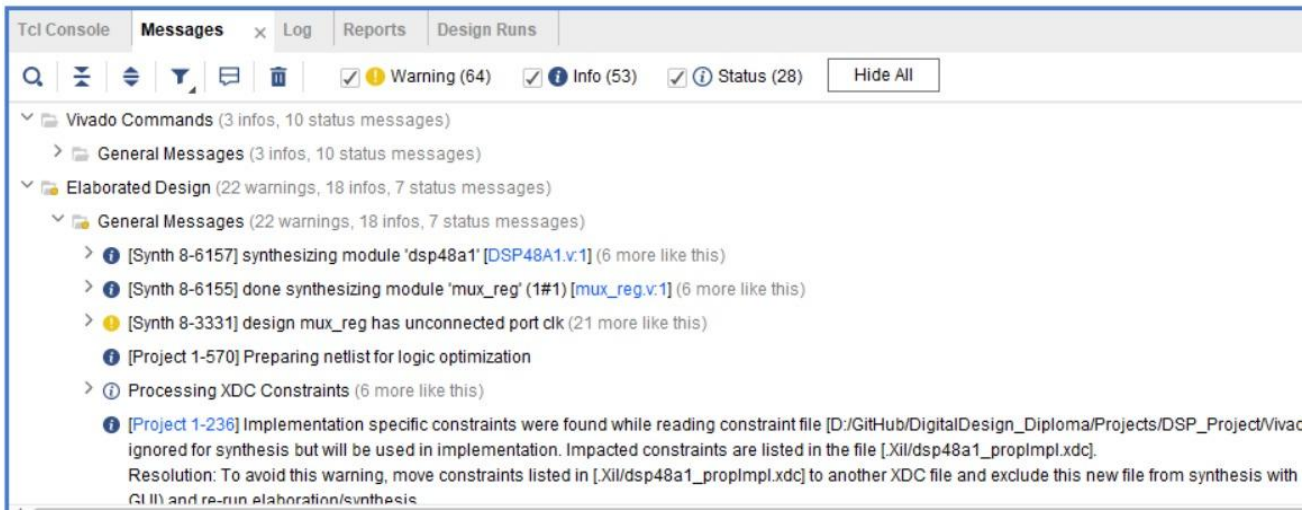


*Figure 8 No Critical Warning or Errors After Elaboration*

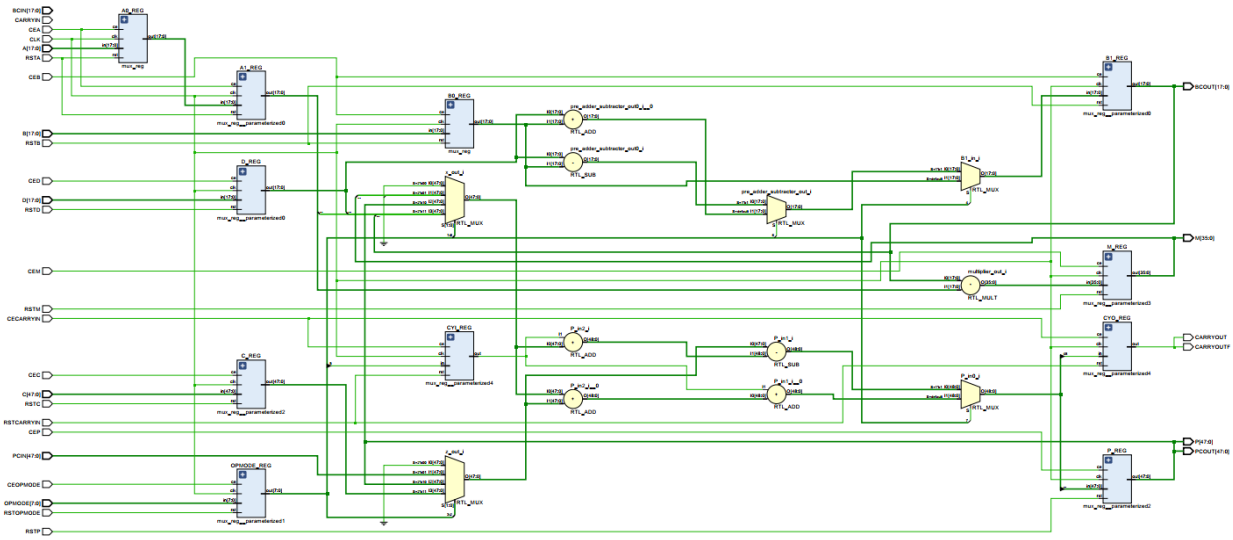

*Figure 9 Elaborated Schematic*

## 7.2 Synthesis


*Figure 10 No Critical Warnings or Errors After Synthesis*


*Figure 11 Synthesis Timing Report*

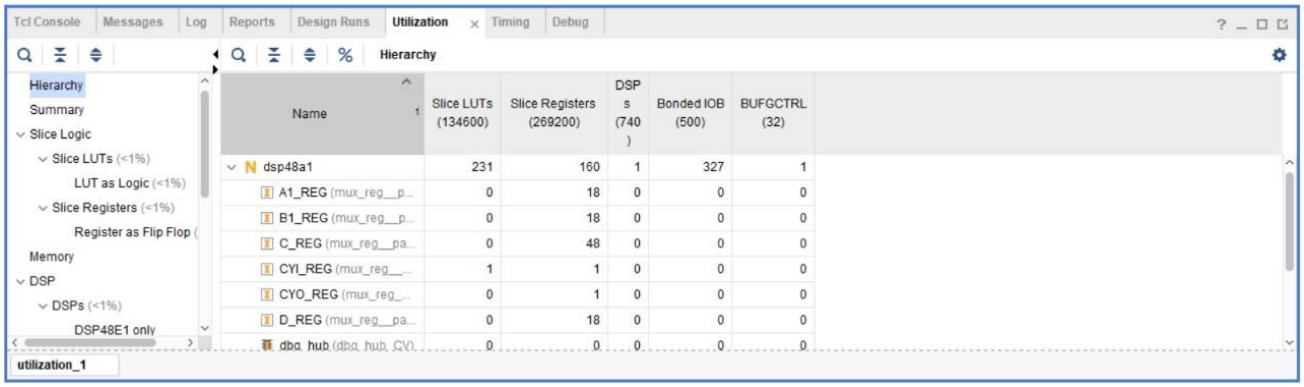


*Figure 12 Synthesis Utilization Report*

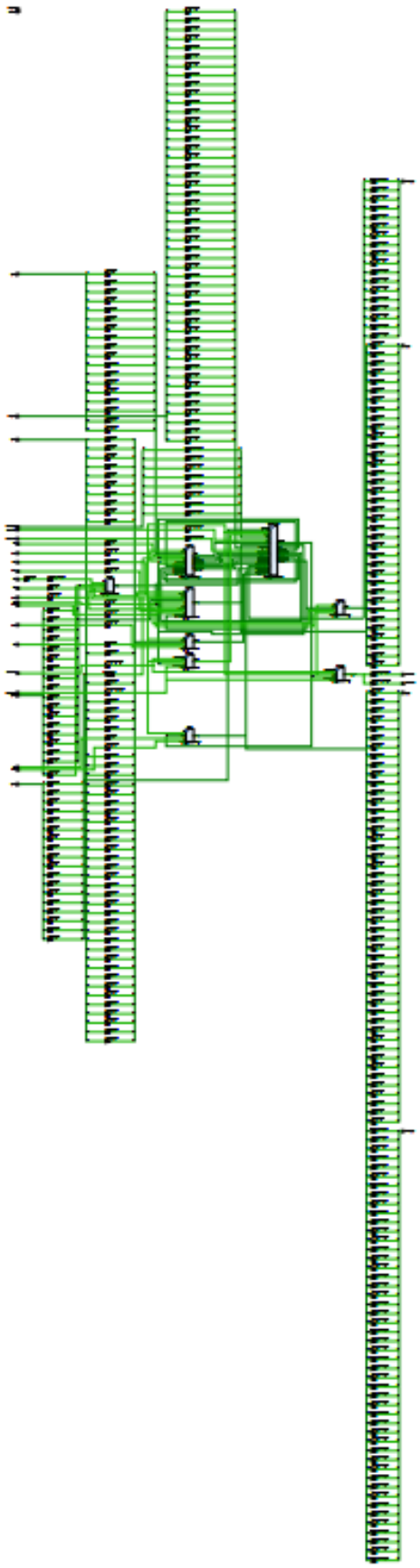

*Figure 13 Synthesized Schematic*

## 7.3 Implementation:

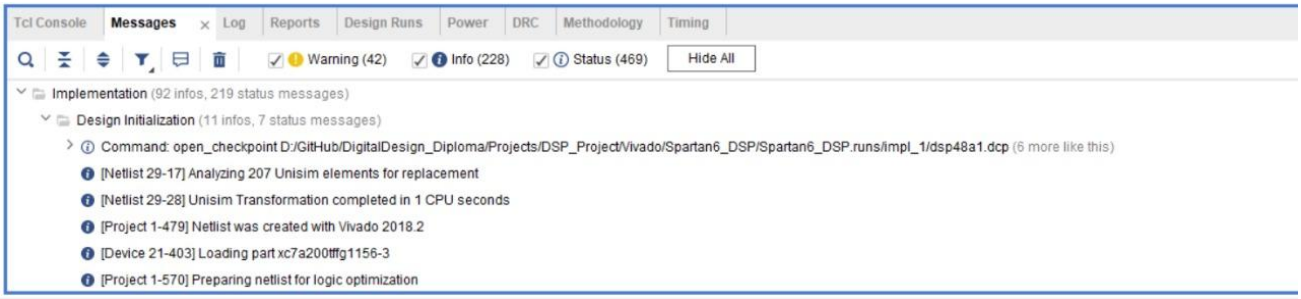


*Figure 14 No Cirtical Warnings or Errors After Implementation*

**Design Timing Summary**

| Setup | | Hold | | Pulse Width | |
|---|---|---|---|---|---|
| Worst Negative Slack (WNS): | 4.447 ns | Worst Hold Slack (WHS): | 0.204 ns | Worst Pulse Width Slack (WPWS): | 4.500 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: | 125 | Total Number of Endpoints: | 125 | Total Number of Endpoints: | 181 |

**All user specified timing constraints are met.**

*Figure 15 Implementation Timing Report*

| Name | Slice LUTs (133800) | Slice Registers (267600) | Slice (33450) | LUT as Logic (133800) | LUT Flip Flop Pairs (133800) | DSPs (740) | Bonded IOB (500) | BUFGCTRL (32) |
|---|---|---|---|---|---|---|---|---|
| dsp48a1 | 230 | 179 | 107 | 230 | 51 | 1 | 327 | 1 |
| A1_REG (mux_reg__p... | 0 | 18 | 6 | 0 | 0 | 0 | 0 | 0 |
| B1_REG (mux_reg__p... | 0 | 36 | 16 | 0 | 0 | 0 | 0 | 0 |
| C_REG (mux_reg__pa... | 0 | 48 | 18 | 0 | 0 | 0 | 0 | 0 |
| CYI_REG (mux_reg__... | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| CYO_REG (mux_reg_... | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |

*Figure 16 Implementation Utilization Report*

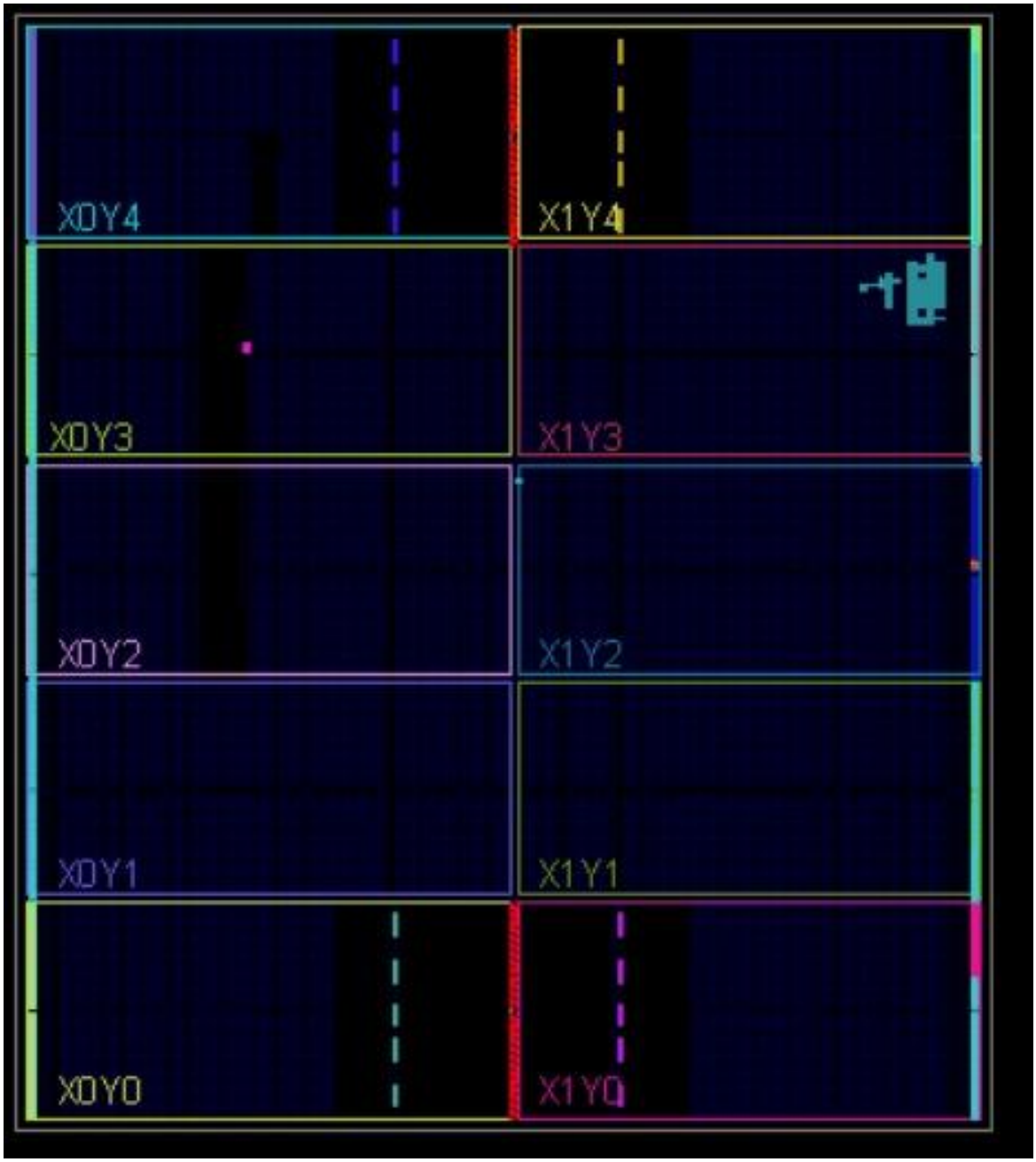


*Figure 17 Implemented Device Snippet*