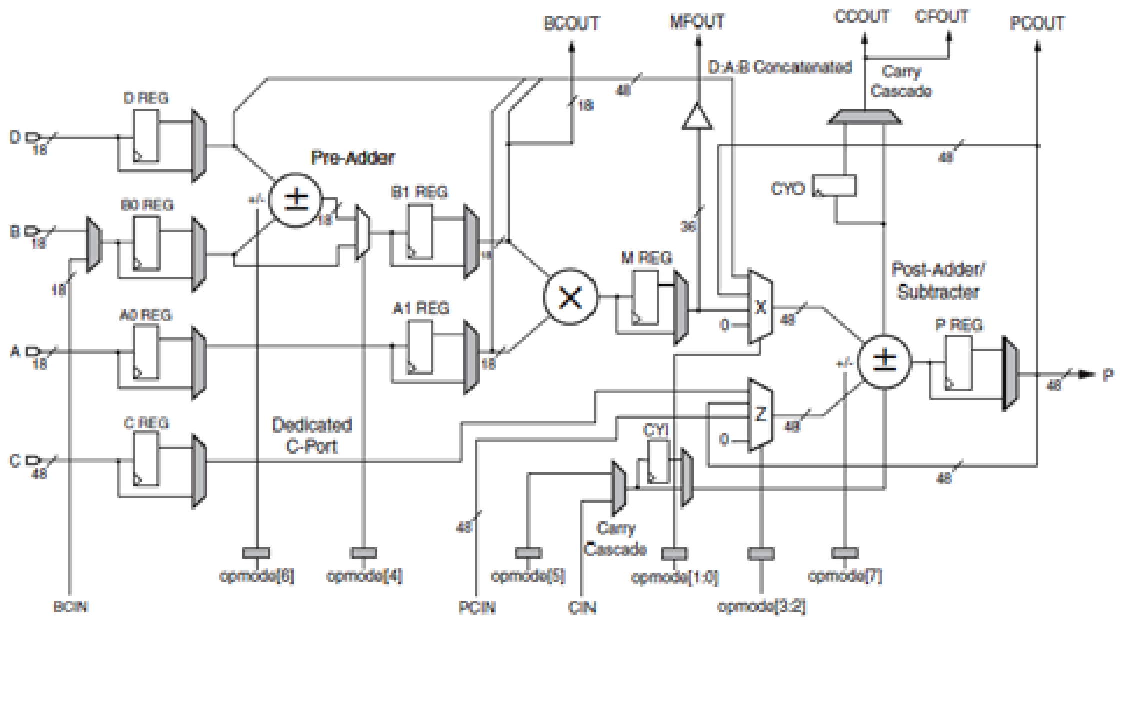
Spartan6 - DSP48A1



1. RTL code

```
module REG_MUX #(
    parameter DATA_WIDTH = 18,   // Width of the data bus
    parameter RSTTYPE = "SYNC",  // Reset type: "SYNC" for synchronous, "ASYNC"
for asynchronous
    parameter REG_OUT = 1            // Selector determines whether to use
registered or direct data
)(
    input CLK, // Clock signal
    input RST, // Reset signal
    input CE,  // Clock enable signal

    input  [DATA_WIDTH - 1:0] D, // Data input
    output [DATA_WIDTH - 1:0] Q  // Data output
);

    reg [DATA_WIDTH - 1:0] D_reg; // Register to store data

    // Output assignment: If REG selector is 1, output registered data; otherwise,
pass input directly
    assign Q = (REG_OUT == 1) ? D_reg : D;

    generate
        if (RSTTYPE == "SYNC") begin
            always @(negedge CLK) begin
                if (RST) begin
                    D_reg <= 0;
                end else if (CE) begin
                    D_reg <= D;
                end
            end
        end else if (RSTTYPE == "ASYNC") begin
            always @(negedge CLK or posedge RST) begin
                if (RST) begin
                    D_reg <= 0;
                end else if (CE) begin
                    D_reg <= D;
                end
            end
        end
    endgenerate

endmodule
```

```
module DSP48A1 #(
    parameter OPMODE_WIDTH = 8,
    parameter D_DATA_WIDTH = 18,
    parameter B_DATA_WIDTH = 18,
    parameter A_DATA_WIDTH = 18,
    parameter C_DATA_WIDTH = 48,
    parameter P_DATA_WIDTH = 48,
    parameter M_DATA_WIDTH = 36,

    parameter A0REG = 0,   // Register A0 or bypass
    parameter A1REG = 1,   // Register A1 or bypass
    parameter B0REG = 0,   // Register B0 or bypass
    parameter B1REG = 1,   // Register B1 or bypass

    parameter CREG = 1,           // Register C or bypass
    parameter DREG = 1,           // Register D or bypass
    parameter MREG = 1,           // Register M or bypass
    parameter PREG = 1,           // Register P or bypass
    parameter CARRYINREG  = 1,    // Register Carry-in or bypass
    parameter CARRYOUTREG = 1,    // Register Carry-out or bypass
    parameter OPMODEREG   = 1,    // Register OPMODE or bypass

    parameter CARRYINSEL = "OPMODE5",    // Carry-in selection for post
addition
    parameter B_INPUT    = "DIRECT",     // Select B input source
    parameter RSTTYPE    = "SYNC"        // Reset type
) (
    input wire [A_DATA_WIDTH - 1:0] A,      // Input A
    input wire [B_DATA_WIDTH - 1:0] B,      // Input B
    input wire [D_DATA_WIDTH - 1:0] D,      // Input D
    input wire [C_DATA_WIDTH - 1:0] C,      // Input C

    input wire CLK,                          // Clock signal
    input wire CARRYIN,                      // Input Carry-in
    input wire [OPMODE_WIDTH - 1:0] OPMODE,  // Input OPMODE
    input wire [B_DATA_WIDTH - 1:0] BCIN,    // Input BCIN

    input wire RSTA,          // Reset for A
    input wire RSTB,          // Reset for B
    input wire RSTC,          // Reset for C
    input wire RSTD,          // Reset for D
    input wire RSTM,          // Reset for M
    input wire RSTP,          // Reset for P
    input wire RSTCARRYIN,    // Reset for Carry-in & Carry-out
    input wire RSTOPMODE,     // Reset for OPMODE

    input wire CEA,           // Clock enable for A
    input wire CEB,           // Clock enable for B
    input wire CEC,           // Clock enable for C
    input wire CED,           // Clock enable for D
    input wire CEP,           // Clock enable for P
    input wire CEM,           // Clock enable for M
    input wire CEOPMODE,      // Clock enable for OPMODE
    input wire CECARRYIN,     // Clock enable for Carry-in

    input wire [P_DATA_WIDTH - 1:0] PCIN,   // Input PCIN

    output wire [B_DATA_WIDTH - 1:0] BCOUT, // Output BCOUT
    output wire [P_DATA_WIDTH - 1:0] PCOUT, // Output PCOUT
    output wire [P_DATA_WIDTH - 1:0] P,     // Output P
    output wire [M_DATA_WIDTH - 1:0] M,     // Output M

    output wire CARRYOUT,     // Output Carry-out
    output wire CARRYOUTF     // Output Carry-out final
);

    wire [OPMODE_WIDTH - 1:0] OPMODE_r;
    wire [D_DATA_WIDTH - 1:0] D_r;
    wire [B_DATA_WIDTH - 1:0] B_r;
    wire [A_DATA_WIDTH - 1:0] A_r;
    wire [C_DATA_WIDTH - 1:0] C_r;
    wire [P_DATA_WIDTH - 1:0] P_r;
    wire [M_DATA_WIDTH - 1:0] M_r;

    wire [B_DATA_WIDTH - 1:0] B0REG_in;
    wire [B_DATA_WIDTH - 1:0] B1REG_in;

    wire [B_DATA_WIDTH - 1:0] PRE_ADD_SUB_OUT;
    wire [M_DATA_WIDTH - 1:0] MUL_OUT;
    wire [P_DATA_WIDTH - 1:0] POST_ADD_SUB_OUT;

    wire [A_DATA_WIDTH - 1:0] A_rr;

    wire CARRY_REG_in;
    wire CIN;
    wire COUT;

    wire [P_DATA_WIDTH - 1:0] MUX_X_out;
    wire [P_DATA_WIDTH - 1:0] MUX_Z_out;


    REG_MUX #(.DATA_WIDTH(OPMODE_WIDTH), .RSTTYPE(RSTTYPE),
.REG_OUT(OPMODEREG))
        OPMODE_REG_inst (.CLK(CLK), .RST(RSTOPMODE), .CE(CEOPMODE),
.D(OPMODE), .Q(OPMODE_r));
```

```
    assign B0REG_in = (B_INPUT == "DIRECT") ? B :
                      ((B_INPUT == "CASCADED") ? BCIN : {B_DATA_WIDTH{1'b0}});

    REG_MUX #(.DATA_WIDTH(D_DATA_WIDTH), .RSTTYPE(RSTTYPE), .REG_OUT(DREG))
        DREG_inst (.CLK(CLK), .RST(RSTD), .CE(CED), .D(D), .Q(D_r));

    REG_MUX #(.DATA_WIDTH(B_DATA_WIDTH), .RSTTYPE(RSTTYPE), .REG_OUT(B0REG))
        B0REG_inst (.CLK(CLK), .RST(RSTB), .CE(CEB), .D(B0REG_in), .Q(B_r));

    REG_MUX #(.DATA_WIDTH(A_DATA_WIDTH), .RSTTYPE(RSTTYPE), .REG_OUT(A0REG))
        A0REG_inst (.CLK(CLK), .RST(RSTA), .CE(CEA), .D(A), .Q(A_r));

    REG_MUX #(.DATA_WIDTH(C_DATA_WIDTH), .RSTTYPE(RSTTYPE), .REG_OUT(CREG))
        CREG_inst (.CLK(CLK), .RST(RSTC), .CE(CEC), .D(C), .Q(C_r));

    Adder_Subtractor #(.DATA_WIDTH(B_DATA_WIDTH), .TYPE("PRE"))
        pre_add_sub_inst (.opmode(OPMODE_r[6]), .x(D_r), .y(B_r), .cin(),
.cout(), .z(PRE_ADD_SUB_OUT));

    MUX2x1 #(.DATA_WIDTH(B_DATA_WIDTH))
        B1MUX_inst (.x0(B_r), .x1(PRE_ADD_SUB_OUT), .sel(OPMODE_r[4]),
.y(B1REG_in));

    REG_MUX #(.DATA_WIDTH(B_DATA_WIDTH), .RSTTYPE(RSTTYPE), .REG_OUT(B1REG))
        B1REG_inst (.CLK(CLK), .RST(RSTB), .CE(CEB), .D(B1REG_in), .Q(BCOUT));

    REG_MUX #(.DATA_WIDTH(A_DATA_WIDTH), .RSTTYPE(RSTTYPE), .REG_OUT(A1REG))
        A1REG_inst (.CLK(CLK), .RST(RSTA), .CE(CEA), .D(A_r), .Q(A_rr));

    Multiplier #(.DATA_WIDTH(A_DATA_WIDTH))
        mult_inst (.x(A_rr), .y(BCOUT), .z(MUL_OUT));

    assign CARRY_REG_in = (CARRYINSEL == "OPMODE5") ? OPMODE_r[5] :
                          ((CARRYINSEL == "CARRYIN") ? CARRYIN : 1'b0);

    REG_MUX #(.DATA_WIDTH(M_DATA_WIDTH), .RSTTYPE(RSTTYPE), .REG_OUT(MREG))
        MREG_inst (.CLK(CLK), .RST(RSTM), .CE(CEM), .D(MUL_OUT), .Q(M_r));

    BUFFER #(.DATA_WIDTH(M_DATA_WIDTH)) MBUF(.in(M_r), .out(M));

    REG_MUX #(.DATA_WIDTH(1'b1), .RSTTYPE(RSTTYPE),
.REG_OUT(CARRYINREG))
        CYI_inst (.CLK(CLK), .RST(RSTCARRYIN), .CE(CECARRYIN),
.D(CARRY_REG_in), .Q(CIN));

    MUX4x1 #(.DATA_WIDTH(P_DATA_WIDTH))
        MUX_X_inst (.sel(OPMODE_r[1:0]), .x0({P_DATA_WIDTH{1'b0}}),
.x1({12'b0, M_r}), .x2(P), .x3({D_r[11:0], A_rr, BCOUT}), .y(MUX_X_out));
```

```verilog
    MUX4x1 #(.DATA_WIDTH(P_DATA_WIDTH))
        MUX_Z_inst (.sel(OPMODE_r[3:2]), .x0({P_DATA_WIDTH{1'b0}}), .x1(PCIN),
.x2(P), .x3(C_r), .y(MUX_Z_out));

    Adder_Subtractor #(.DATA_WIDTH(P_DATA_WIDTH), .TYPE("POST"))
        post_add_sub_inst (.opmode(OPMODE_r[7]), .x(MUX_Z_out), .y(MUX_X_out),
.cin(CIN), .cout(COUT), .z(POST_ADD_SUB_OUT));

    REG_MUX #(.DATA_WIDTH(P_DATA_WIDTH), .RSTTYPE(RSTTYPE),
.REG_OUT(PREG))
        PREG_inst (.CLK(CLK), .RST(RSTP), .CE(CEP), .D(POST_ADD_SUB_OUT),
.Q(P));

    REG_MUX #(.DATA_WIDTH(1'b1), .RSTTYPE(RSTTYPE),
.REG_OUT(CARRYOUTREG))
        CYO_inst (.CLK(CLK), .RST(RSTCARRYIN), .CE(CECARRYIN), .D(COUT),
.Q(CARRYOUT));


    assign PCOUT = P;
    assign CARRYOUTF = CARRYOUT;

endmodule
```

2. Testbench code

```
module DSP48A1_tb;
    // Parameter definitions
    integer NUM_TEST_CASES = 1000;
    integer pass_count = 0;
    integer error_count = 0;

    parameter OPMODE_WIDTH = 8;
    parameter D_DATA_WIDTH = 18;
    parameter B_DATA_WIDTH = 18;
    parameter A_DATA_WIDTH = 18;
    parameter C_DATA_WIDTH = 48;
    parameter P_DATA_WIDTH = 48;
    parameter M_DATA_WIDTH = 36;

    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"; // "CARRYIN"
    parameter B_INPUT    = "DIRECT";  // "CASCADED"
    parameter RSTTYPE    = "SYNC";

    // Signal definitions
    reg [A_DATA_WIDTH - 1:0] A;
    reg [B_DATA_WIDTH - 1:0] B;
    reg [D_DATA_WIDTH - 1:0] D;
    reg [C_DATA_WIDTH - 1:0] C;

    reg CLK;
    reg CARRYIN;
    reg [OPMODE_WIDTH - 1:0] OPMODE;
    reg [B_DATA_WIDTH - 1:0] BCIN;

    reg RSTA;
    reg RSTB;
    reg RSTC;
    reg RSTD;
    reg RSTM;
    reg RSTP;
    reg RSTCARRYIN;
    reg RSTOPMODE;

    reg CEA;
    reg CEB;
    reg CEC;
    reg CED;
    reg CEP;
    reg CEM;
    reg CEOPMODE;
    reg CECARRYIN;

    reg [P_DATA_WIDTH - 1:0] PCIN;

    wire [B_DATA_WIDTH - 1:0] BCOUT;
    wire [P_DATA_WIDTH - 1:0] PCOUT;
    wire [P_DATA_WIDTH - 1:0] P;
    wire [M_DATA_WIDTH - 1:0] M;

    wire CARRYOUT;
    wire CARRYOUTF;

    reg [A_DATA_WIDTH-1:0] A_tb;
    reg [B_DATA_WIDTH-1:0] B_tb;
    reg [D_DATA_WIDTH-1:0] D_tb;
    reg [C_DATA_WIDTH-1:0] C_tb;

    reg CARRYIN_tb;

    reg [OPMODE_WIDTH-1:0] OPMODE_tb;
    reg [B_DATA_WIDTH-1:0] BCIN_tb;
    reg [P_DATA_WIDTH-1:0] PCIN_tb;

    reg [B_DATA_WIDTH - 1:0] expected_BCOUT;
    reg [P_DATA_WIDTH - 1:0] expected_P;
    reg [M_DATA_WIDTH - 1:0] expected_M;
    reg expected_CARRYOUT;

    // Instantiate DUT
    DSP48A1 #(
        .OPMODE_WIDTH(OPMODE_WIDTH),
        .D_DATA_WIDTH(D_DATA_WIDTH),
        .B_DATA_WIDTH(B_DATA_WIDTH),
        .A_DATA_WIDTH(A_DATA_WIDTH),
        .C_DATA_WIDTH(C_DATA_WIDTH),
        .P_DATA_WIDTH(P_DATA_WIDTH),
        .M_DATA_WIDTH(M_DATA_WIDTH),
            .A0REG(A0REG),
            .A1REG(A1REG),
            .B0REG(B0REG),
            .B1REG(B1REG),
            .CREG(CREG),
            .DREG(DREG),
            .MREG(MREG),
            .PREG(PREG),
            .CARRYINREG(CARRYINREG),
            .CARRYOUTREG(CARRYOUTREG),
            .OPMODEREG(OPMODEREG),
            .CARRYINSEL(CARRYINSEL),
            .B_INPUT(B_INPUT),
            .RSTTYPE(RSTTYPE)
        ) dut (
            .A(A),
            .B(B),
            .D(D),
            .C(C),
            .CLK(CLK),
            .CARRYIN(CARRYIN),
            .OPMODE(OPMODE),
            .BCIN(BCIN),
            .RSTA(RSTA),
            .RSTB(RSTB),
            .RSTC(RSTC),
            .RSTD(RSTD),
            .RSTM(RSTM),
            .RSTP(RSTP),
            .RSTCARRYIN(RSTCARRYIN),
            .RSTOPMODE(RSTOPMODE),
            .CEA(CEA),
            .CEB(CEB),
            .CEC(CEC),
            .CED(CED),
            .CEP(CEP),
            .CEM(CEM),
            .CEOPMODE(CEOPMODE),
            .CECARRYIN(CECARRYIN),
            .PCIN(PCIN),
            .BCOUT(BCOUT),
            .PCOUT(PCOUT),
            .P(P),
            .M(M),
            .CARRYOUT(CARRYOUT),
            .CARRYOUTF(CARRYOUTF)
        );
```

```
always #5 CLK = ~CLK; // 10 ns clock period (100 MHz)

task wait_cycles;
    input integer num_cycles;
    begin
        repeat(num_cycles) @(posedge CLK);
    end
endtask

task assert_reset;
    begin
        // Assert reset signals
        RSTA = 1;
        RSTB = 1;
        RSTC = 1;
        RSTD = 1;
        RSTM = 1;
        RSTP = 1;
        RSTCARRYIN = 1;
        RSTOPMODE = 1;

        // Display reset assertion message
        $display("Reset asserted");

        // Wait for 1 cycle
        wait_cycles(1);

        // Deassert reset signals
        RSTA = 0;
        RSTB = 0;
        RSTC = 0;
        RSTD = 0;
        RSTM = 0;
        RSTP = 0;
        RSTCARRYIN = 0;
        RSTOPMODE = 0;

        // Display reset deassertion message
        // $display("Reset deasserted\n");
    end
endtask


task drive_inputs;
    input [A_DATA_WIDTH-1:0] A_in;
    input [B_DATA_WIDTH-1:0] B_in;
    input [D_DATA_WIDTH-1:0] D_in;
    input [C_DATA_WIDTH-1:0] C_in;
input CARRYIN_in;
input [OPMODE_WIDTH-1:0] OPMODE_in;
input [B_DATA_WIDTH-1:0] BCIN_in;
input [P_DATA_WIDTH-1:0] PCIN_in;

begin
    // Assign inputs
    A = A_in;
    B = B_in;
    D = D_in;
    C = C_in;
    CARRYIN = CARRYIN_in;
    OPMODE = OPMODE_in;
    BCIN = BCIN_in;
    PCIN = PCIN_in;

    // Enable clock signals
    // First stage
    CEA = 1;
    CEB = 1;
    CEC = 1;
    CED = 1;
    CEOPMODE = 1;
    wait_cycles(1);
    CEA = 0;
    CEB = 0;
    CEC = 0;
    CED = 0;
    CEOPMODE = 0;

    // Second stage
    CEA = 1;
    CEB = 1;
    wait_cycles(1);
    CEA = 0;
    CEB = 0;

    // Third stage
    CEM = 1;
    CECARRYIN = 1;
    wait_cycles(1);
    CEM = 0;
    CECARRYIN = 0;

    // Forth stage
    CEP = 1;
    CECARRYIN = 1;
    wait_cycles(1);
```

```
            CEP = 0;
            CECARRYIN = 0;

            // Display stimulus
            $display("A = %h | B = %h | D = %h | C = %h | CARRYIN = %b | OPMODE = 
%b | BCIN = %h | PCIN = %h\n", A_in, B_in, D_in, C_in, CARRYIN_in, OPMODE_in, 
BCIN_in, PCIN_in);

            // Wait for 1 clock cycles
            // wait_cycles(1);

            // Disable clock signals
            CEA = 0;
            CEB = 0;
            CEC = 0;
            CED = 0;
            CEP = 0;
            CEM = 0;
            CEOPMODE = 0;
            CECARRYIN = 0;

            // Display clock disable
            // $display("Time: %0t | Clock Enables Cleared", $time);
        end
    endtask

    task check_outputs;
        input [B_DATA_WIDTH - 1:0] expected_BCOUT;
        input [P_DATA_WIDTH - 1:0] expected_P;
        input [M_DATA_WIDTH - 1:0] expected_M;
        input expected_CARRYOUT;
        integer local_errors;

        begin
            local_errors = 0; // Track errors in the current test case

            if (BCOUT !== expected_BCOUT) begin
                $display("Mismatch: BCOUT = %h, Expected = %h", BCOUT, 
expected_BCOUT);
                local_errors = local_errors + 1;
            end else
                $display("Match: BCOUT = %h", BCOUT);

            if ((P !== expected_P) | (PCOUT !== expected_P)) begin
                $display("Mismatch: P = %h, Expected = %h", P, expected_P);
                local_errors = local_errors + 1;
            end else
                $display("Match: P = %h", P);
            if (M !== expected_M) begin
                $display("Mismatch: M = %h, Expected = %h", M, expected_M);
                local_errors = local_errors + 1;
            end else
                $display("Match: M = %h", M);

            if ((CARRYOUT !== expected_CARRYOUT) | (CARRYOUTF !==
expected_CARRYOUT)) begin
                $display("Mismatch: CARRYOUT = %b, Expected = %b", CARRYOUT,
expected_CARRYOUT);
                local_errors = local_errors + 1;
            end else
                $display("Match: CARRYOUT = %b", CARRYOUT);

            // Update pass and error counters
            if (local_errors == 0) begin
                pass_count = pass_count + 1;
                $display("Test Case PASSED!\n");
            end else begin
                error_count = error_count + 1;
                $display("Test Case FAILED! Mismatch Number: %0d\n",
local_errors);
            end
        end
    endtask

    function automatic void model_DSP48A1(
        input [A_DATA_WIDTH - 1:0] A_in,
        input [B_DATA_WIDTH - 1:0] B_in,
        input [D_DATA_WIDTH - 1:0] D_in,
        input [C_DATA_WIDTH - 1:0] C_in,
        input CARRYIN_in,
        input [OPMODE_WIDTH - 1:0] OPMODE_in,
        input [B_DATA_WIDTH - 1:0] BCIN_in,
        input [P_DATA_WIDTH - 1:0] PCIN_in,
        output [B_DATA_WIDTH - 1:0] BCOUT_out,
        output [P_DATA_WIDTH - 1:0] P_out,
        output [M_DATA_WIDTH - 1:0] M_out,
        output CARRYOUT_out
    );
        // Internal Registers
        reg [B_DATA_WIDTH - 1:0] B;
        reg CIN;
        reg COUT;
        reg [M_DATA_WIDTH - 1:0] mult_result;
        reg [B_DATA_WIDTH - 1:0] pre_add_result;
        reg [P_DATA_WIDTH - 1:0] post_add_result;
```

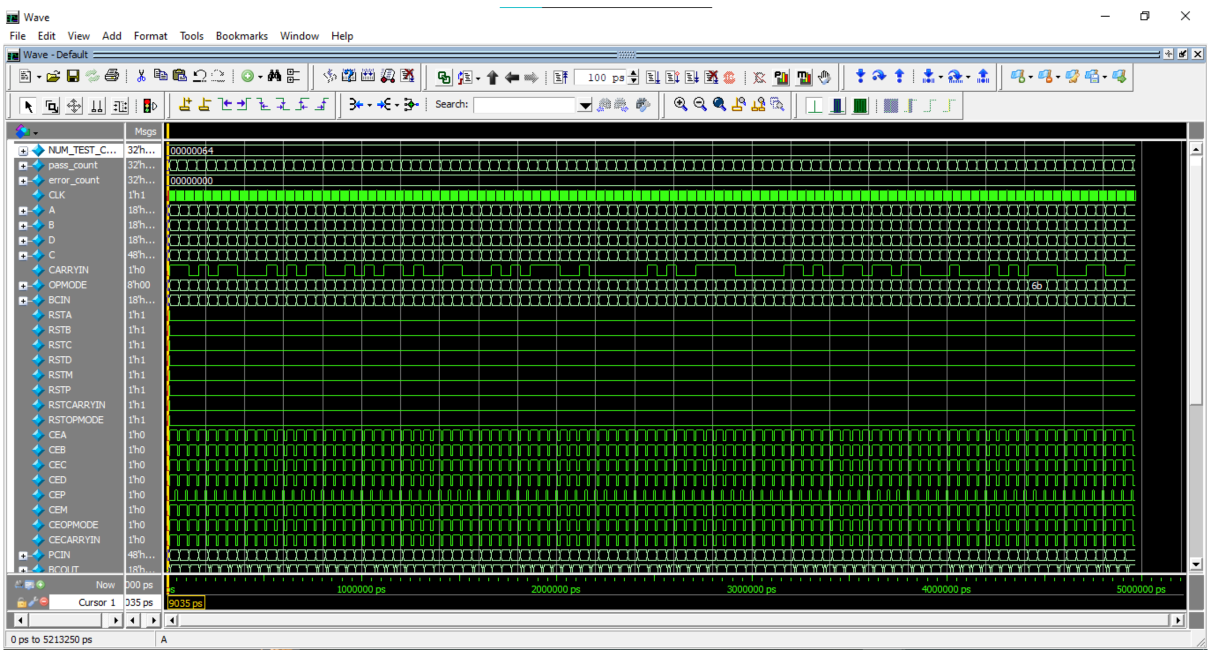
```verilog
    reg [P_DATA_WIDTH - 1:0] MUX_X_out;
    reg [P_DATA_WIDTH - 1:0] MUX_Z_out;
    static reg [P_DATA_WIDTH - 1:0] P = 0;

    // B Selection Logic
    B = (B_INPUT == "DIRECT") ? B_in :
        ((B_INPUT == "CASCADED") ? BCIN_in : {B_DATA_WIDTH{1'b0}});

    // Carry-In Selection
    CIN = (CARRYINSEL == "OPMODE5") ? OPMODE_in[5] :
          ((CARRYINSEL == "CARRYIN") ? CARRYIN_in : 1'b0);

    // Pre-Adder Operation
    pre_add_result = OPMODE_in[6] ? (D_in - B) : (D_in + B);
    BCOUT_out = OPMODE_in[4] ? pre_add_result : B;

    // Multiplier Operation
    mult_result = A_in * BCOUT_out;
    M_out = mult_result;

    // MUX X Logic
    case (OPMODE_in[1:0])
        2'b00: MUX_X_out = {P_DATA_WIDTH{1'b0}};
        2'b01: MUX_X_out = {12'b0, M_out};
        2'b10: MUX_X_out = P;
        2'b11: MUX_X_out = {D_in[11:0], A_in, BCOUT_out};
    endcase

    // MUX Z Logic
    case (OPMODE_in[3:2])
        2'b00: MUX_Z_out = {P_DATA_WIDTH{1'b0}};
        2'b01: MUX_Z_out = PCIN_in;
        2'b10: MUX_Z_out = P;
        2'b11: MUX_Z_out = C_in;
    endcase

    // Final Addition or Subtraction with Carry
    {COUT, post_add_result} = OPMODE_in[7] ?
                              (MUX_Z_out - (MUX_X_out + CIN)) :
                              (MUX_Z_out + (MUX_X_out + CIN));

    // Output Assignments
    P = post_add_result;
    P_out = post_add_result;
    CARRYOUT_out = COUT;
endfunction

task initialize_DSP48A1;
```

```verilog
        begin
            // Initialize all signals to default values
            A = 0;
            B = 0;
            D = 0;
            C = 0;
            BCIN = 0;
            PCIN = 0;
            CARRYIN = 0;
            OPMODE = 0;

            // Set reset signals
            RSTA = 0;
            RSTB = 0;
            RSTC = 0;
            RSTD = 0;
            RSTM = 0;
            RSTP = 0;
            RSTCARRYIN = 0;
            RSTOPMODE = 0;

            // Disable clock enables
            CEA = 0;
            CEB = 0;
            CEC = 0;
            CED = 0;
            CEP = 0;
            CEM = 0;
            CEOPMODE = 0;
            CECARRYIN = 0;

            // Initialize clock to low
            CLK = 0;

            $display("DSP48A1 Module Initialized.\n");
        end
    endtask

    initial
    begin
        initialize_DSP48A1;
        wait_cycles(1);

        // Display test case number
        $display("Running Test Case #0");
        assert_reset;
        expected_BCOUT = {B_DATA_WIDTH{1'b0}};
        expected_P = {P_DATA_WIDTH{1'b0}};
        expected_M = {M_DATA_WIDTH{1'b0}};
        expected_CARRYOUT = 1'b0;
        check_outputs(expected_BCOUT, expected_P, expected_M, expected_CARRYOUT);

        for (int i = 1; i < NUM_TEST_CASES; i++) begin
            // Generate random test inputs
            A_tb = $random;
            B_tb = $random;
            D_tb = $random;
            C_tb = $random;
            CARRYIN_tb = $random;
            OPMODE_tb = $random;
            BCIN_tb = $random;
            PCIN_tb = $random;

            // Display test case number
            $display("Running Test Case #%0d", i);

            // Drive inputs into the module
            drive_inputs(A_tb, B_tb, D_tb, C_tb, CARRYIN_tb, OPMODE_tb, BCIN_tb,
PCIN_tb);

            // Run expected model for reference output
            model_DSP48A1(A_tb, B_tb, D_tb, C_tb, CARRYIN_tb, OPMODE_tb, BCIN_tb,
PCIN_tb, expected_BCOUT, expected_P, expected_M, expected_CARRYOUT);

            // Check if actual outputs match expected outputs
            check_outputs(expected_BCOUT, expected_P, expected_M,
expected_CARRYOUT);

            // Wait for 1 clock cycle
            // wait_cycles(1);
        end

        // Display completion message
        $display("Simulation Completed: %0d test cases executed.",
NUM_TEST_CASES);
        $display("Test Summary: Passed = %0d, Failed = %0d", pass_count,
error_count);
        $stop;
    end
endmodule
```

3. Do file

```
vlog DSP48A1.v Multiplier.v MUX2x1.v MUX4x1.v REG_MUX.v Adder_Subtractor.v
BUF.v DSP48A1_tb.sv
vsim -voptargs=+acc work.DSP48A1_tb
add wave *
run -all
# quit -sim

> do run.do
```

4. QuestaSim snippet



```
Running Test Case #997
A = 0735c | B = 29a60 | D = 3ed68 | C = ffffba2aa474 | CARRYIN = 0 |
OPMODE = 00000101 | BCIN = 02c23 | PCIN = ffffb89c5671

Match: BCOUT = 29a60
Match: P = 0000e4e4f0f1
Match: M = 12c489a80
Match: CARRYOUT = 1
Test Case PASSED!
```

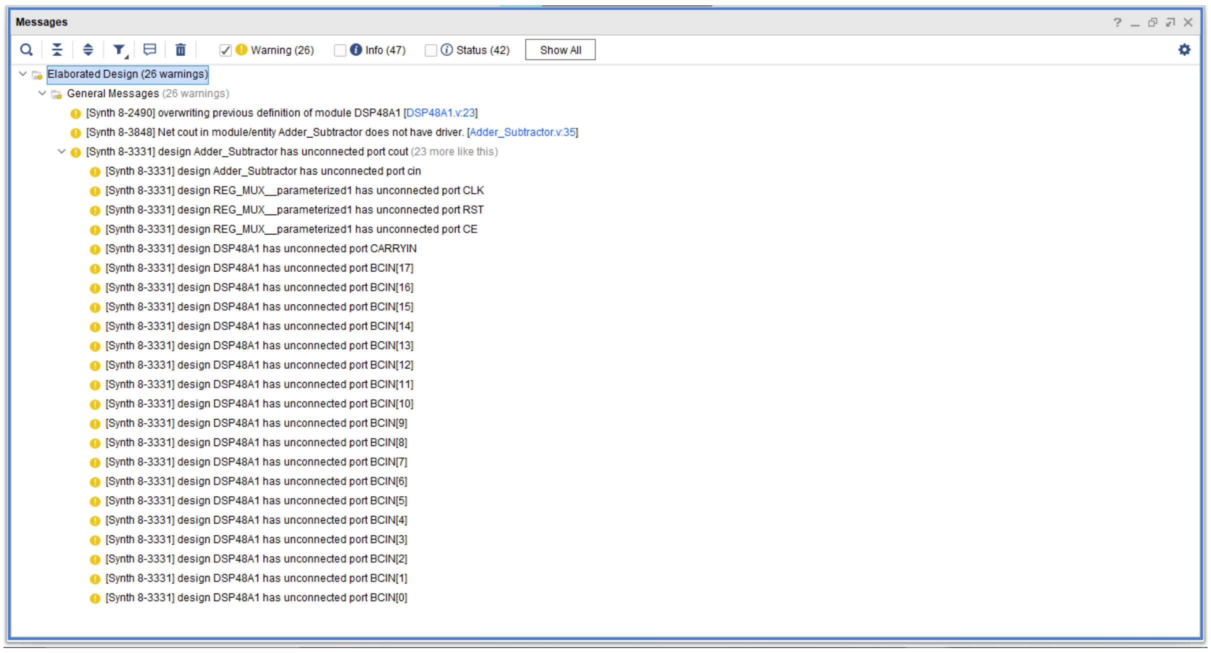
```
Running Test Case #998
A = 34e14 | B = 12ab3 | D = 1788b | C = ffffba213c74 | CARRYIN = 0 |
OPMODE = 11111100 | BCIN = 13cdc | PCIN = ffffe22dd6c4

Match: BCOUT = 04dd8
Match: P = ffffba213c73
Match: M = 10145e4e0
Match: CARRYOUT = 0
Test Case PASSED!

Running Test Case #999
A = 27829 | B = 09774 | D = 330df | C = 0000188f1331 | CARRYIN = 0 |
OPMODE = 01100000 | BCIN = 3390c | PCIN = ffffb8805271

Match: BCOUT = 09774
Match: P = 000000000001
Match: M = 175fea194
Match: CARRYOUT = 0
Test Case PASSED!

Simulation Completed: 1000 test cases executed.
Test Summary: Passed = 1000, Failed = 0
```

5. Constraint 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]
```

6. Elaboration

1. “Messages” tab



Messages

Warning (26) Info (47) Status (42) Show All

Elaborated Design (26 warnings)

General Messages (26 warnings)

[Synth 8-2490] overwriting previous definition of module DSP48A1 [DSP48A1.v:23]

[Synth 8-3848] Net cout in module/entity Adder_Subtractor does not have driver. [Adder_Subtractor.v:35]

[Synth 8-3331] design Adder_Subtractor has unconnected port cout (23 more like this)

[Synth 8-3331] design Adder_Subtractor has unconnected port cin

[Synth 8-3331] design REG_MUX__parameterized1 has unconnected port CLK

[Synth 8-3331] design REG_MUX__parameterized1 has unconnected port RST

[Synth 8-3331] design REG_MUX__parameterized1 has unconnected port CE

[Synth 8-3331] design DSP48A1 has unconnected port CARRYIN

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[17]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[16]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[15]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[14]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[13]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[12]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[11]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[10]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[9]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[8]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[7]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[6]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[5]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[4]

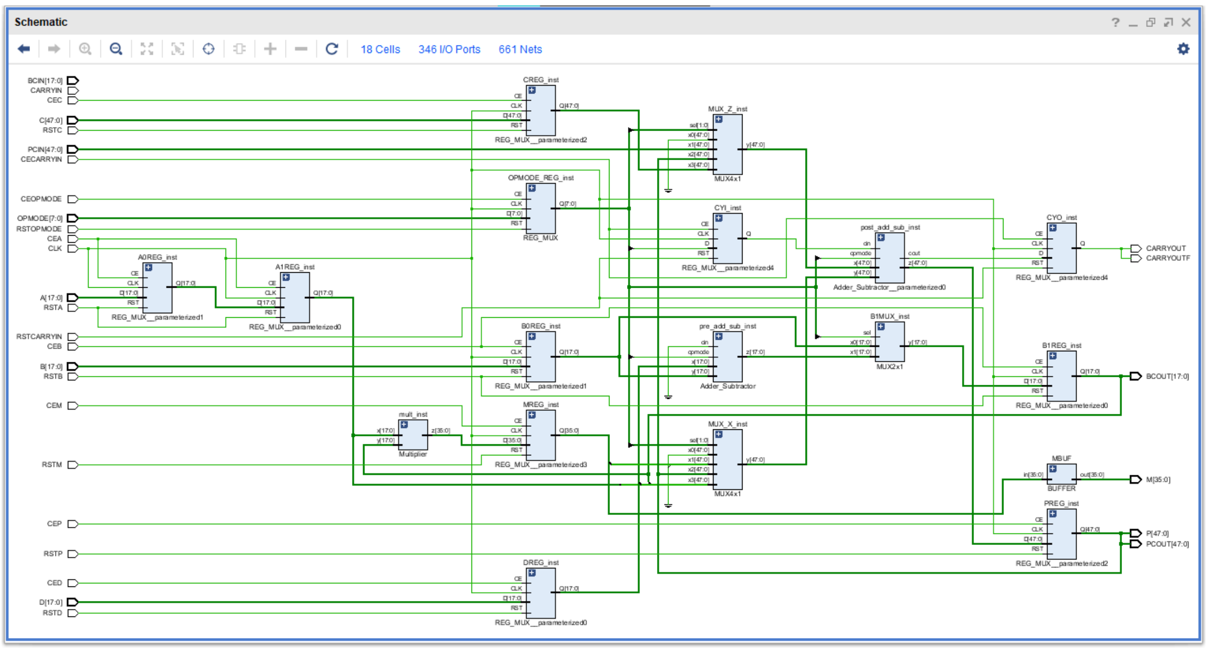
[Synth 8-3331] design DSP48A1 has unconnected port BCIN[3]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[2]

[Synth 8-3331] design DSP48A1 has unconnected port BCIN[1]

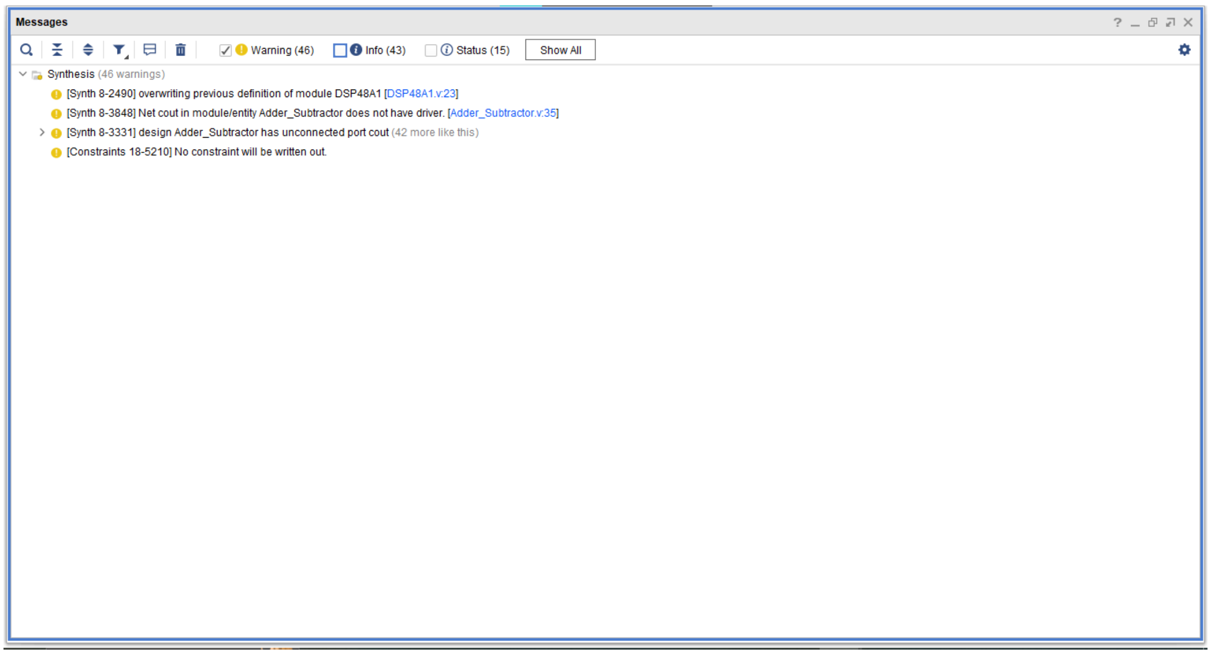
[Synth 8-3331] design DSP48A1 has unconnected port BCIN[0]

2. Schematic snippet



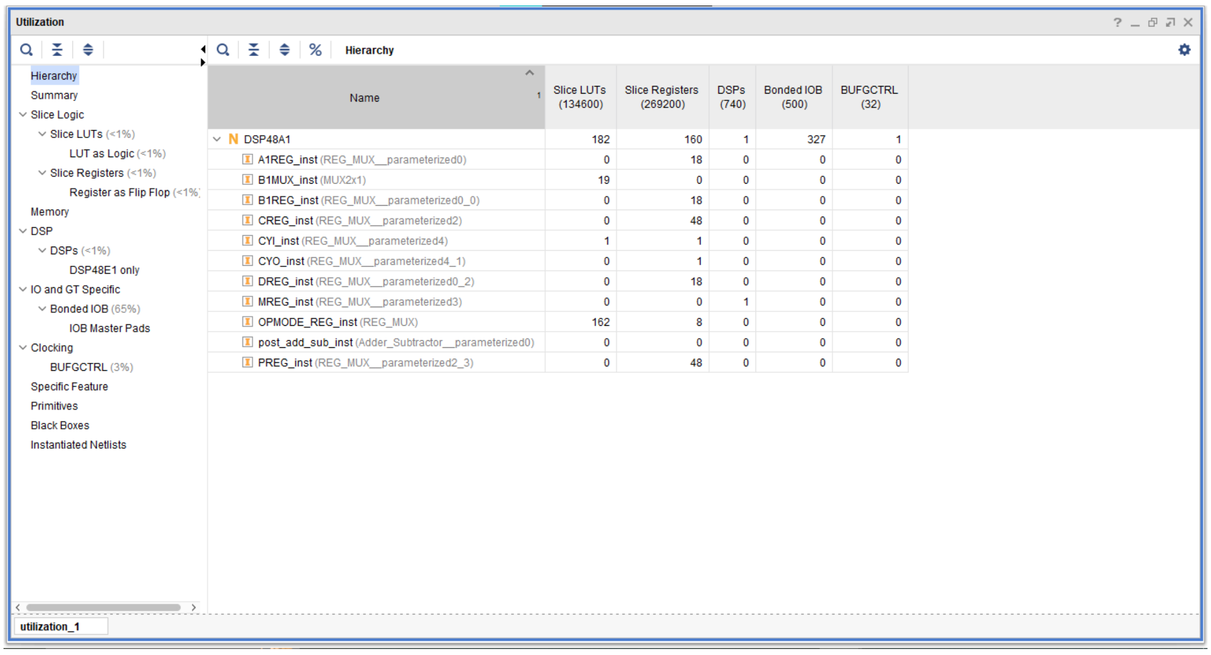
7. Synthesis

1. “Messages” tab



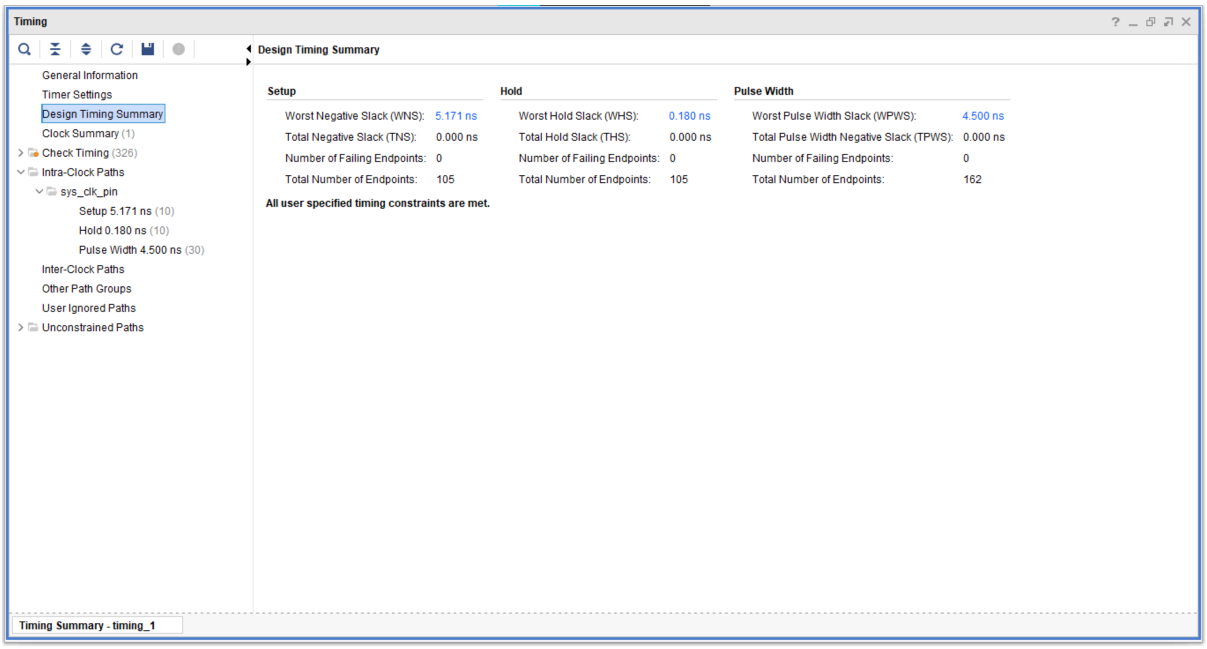
```
Synthesis (46 warnings)
    [Synth 8-2490] overwriting previous definition of module DSP48A1 [DSP48A1.v:23]
    [Synth 8-3848] Net cout in module/entity Adder_Subtractor does not have driver. [Adder_Subtractor.v:35]
  > [Synth 8-3331] design Adder_Subtractor has unconnected port cout (42 more like this)
    [Constraints 18-5210] No constraint will be written out.
```

2. Utilization report



Name	Slice LUTs (134600)	Slice Registers (269200)	DSPs (740)	Bonded IOB (500)	BUFGCTRL (32)
DSP48A1	182	160	1	327	1
A1REG_inst (REG_MUX__parameterized0)	0	18	0	0	0
B1MUX_inst (MUX2x1)	19	0	0	0	0
B1REG_inst (REG_MUX__parameterized0_0)	0	18	0	0	0
CREG_inst (REG_MUX__parameterized2)	0	48	0	0	0
CYI_inst (REG_MUX__parameterized4)	1	1	0	0	0
CYO_inst (REG_MUX__parameterized4_1)	0	1	0	0	0
DREG_inst (REG_MUX__parameterized0_2)	0	18	0	0	0
MREG_inst (REG_MUX__parameterized3)	0	0	1	0	0
OPMODE_REG_inst (REG_MUX)	162	8	0	0	0
post_add_sub_inst (Adder_Subtractor__parameterized0)	0	0	0	0	0
PREG_inst (REG_MUX__parameterized2_3)	0	48	0	0	0

3. Timing report


Timing

General Information
Timer Settings
Design Timing Summary
Clock Summary (1)
Check Timing (326)
Intra-Clock Paths
sys_clk_pin
Setup 5.171 ns (10)
Hold 0.180 ns (10)
Pulse Width 4.500 ns (30)
Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths

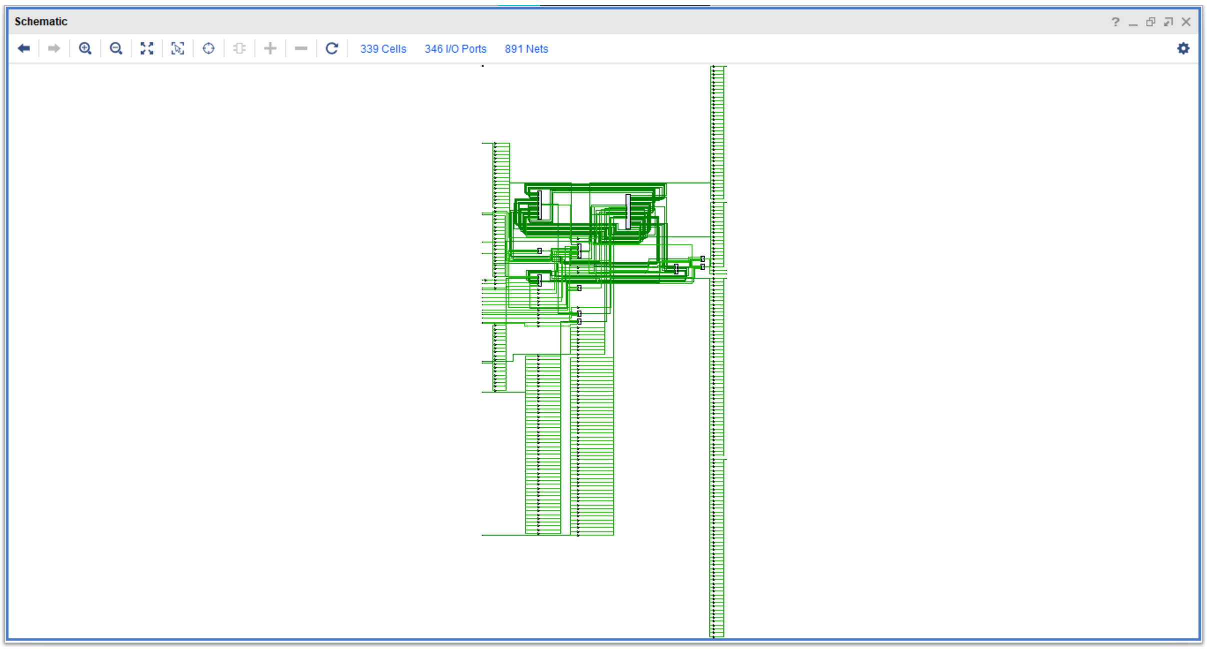
Design Timing Summary

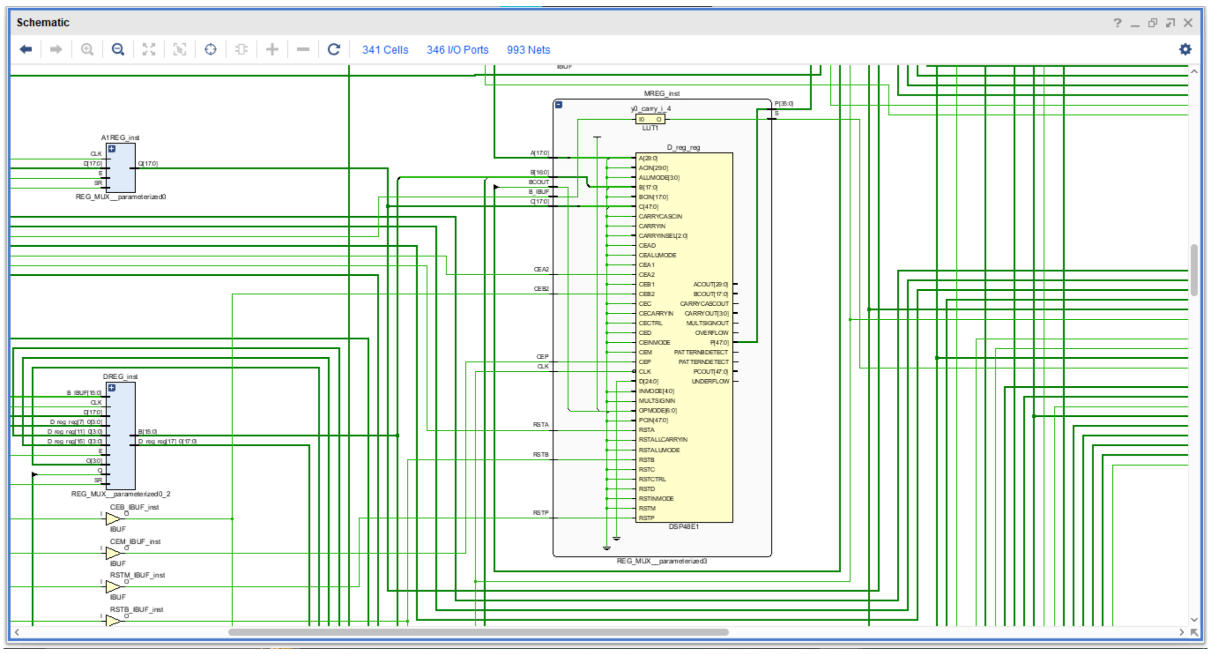
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.171 ns	Worst Hold Slack (WHS):	0.180 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	105	Total Number of Endpoints:	105	Total Number of Endpoints:	162

All user specified timing constraints are met.

Timing Summary - timing_1

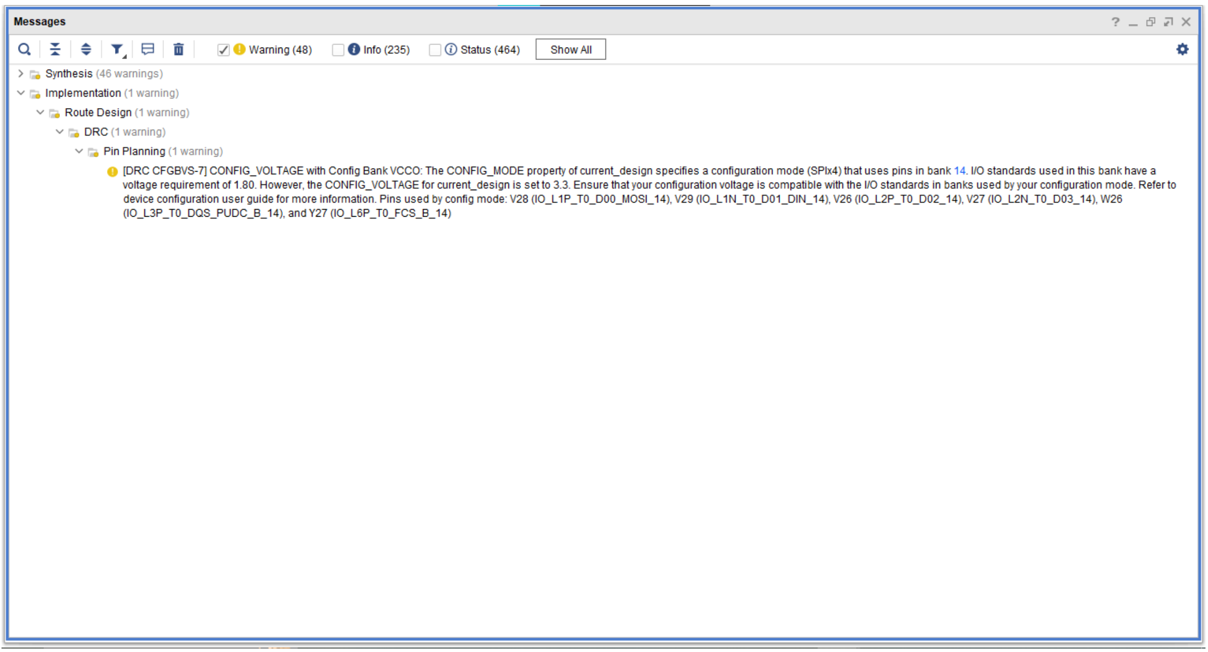
4. Schematic snippet




Schematic
341 Cells
346 I/O Ports
993 Nets
IBUF
A1REG_inst
CLK
D[17:0]
E
SR
Q[17:0]
REG_MUX__parameterized0
MREG_inst
y0_carry_i_4
I0
O
LUT1
P[35:0]
S
D_reg_reg
A[17:0]
B[16:0]
BCOUT
B_IBUF
C[17:0]
CEA2
CEB2
CEP
CLK
RSTA
RSTB
RSTP
A[29:0]
ACIN[29:0]
ALUMODE[3:0]
B[17:0]
BCIN[17:0]
C[47:0]
CARRYCASCIN
CARRYIN
CARRYINSEL[2:0]
CEAD
CEALUMODE
CEA1
CEA2
CEB1
CEB2
CEC
CECARRYIN
CECTRL
CED
CEINMODE
CEM
CEP
CLK
D[24:0]
INMODE[4:0]
MULTSIGNIN
OPMODE[6:0]
PCIN[47:0]
RSTA
RSTALLCARRYIN
RSTALUMODE
RSTB
RSTC
RSTCTRL
RSTD
RSTINMODE
RSTM
RSTP
ACOUT[29:0]
BCOUT[17:0]
CARRYCASCOUT
CARRYOUT[3:0]
MULTSIGNOUT
OVERFLOW
P[47:0]
PATTERNBDETECT
PATTERNDETECT
PCOUT[47:0]
UNDERFLOW
DSP48E1
REG_MUX__parameterized3
DREG_inst
B_IBUF[15:0]
CLK
D[17:0]
D_reg_reg[7]_0[3:0]
D_reg_reg[11]_0[3:0]
D_reg_reg[15]_0[3:0]
E
O[3:0]
Q
SR
B[15:0]
D_reg_reg[17]_0[17:0]
REG_MUX__parameterized0_2
CEB_IBUF_inst
I
O
IBUF
CEM_IBUF_inst
I
O
IBUF
RSTM_IBUF_inst
I
O
IBUF
RSTB_IBUF_inst
I
O

8. Implementation

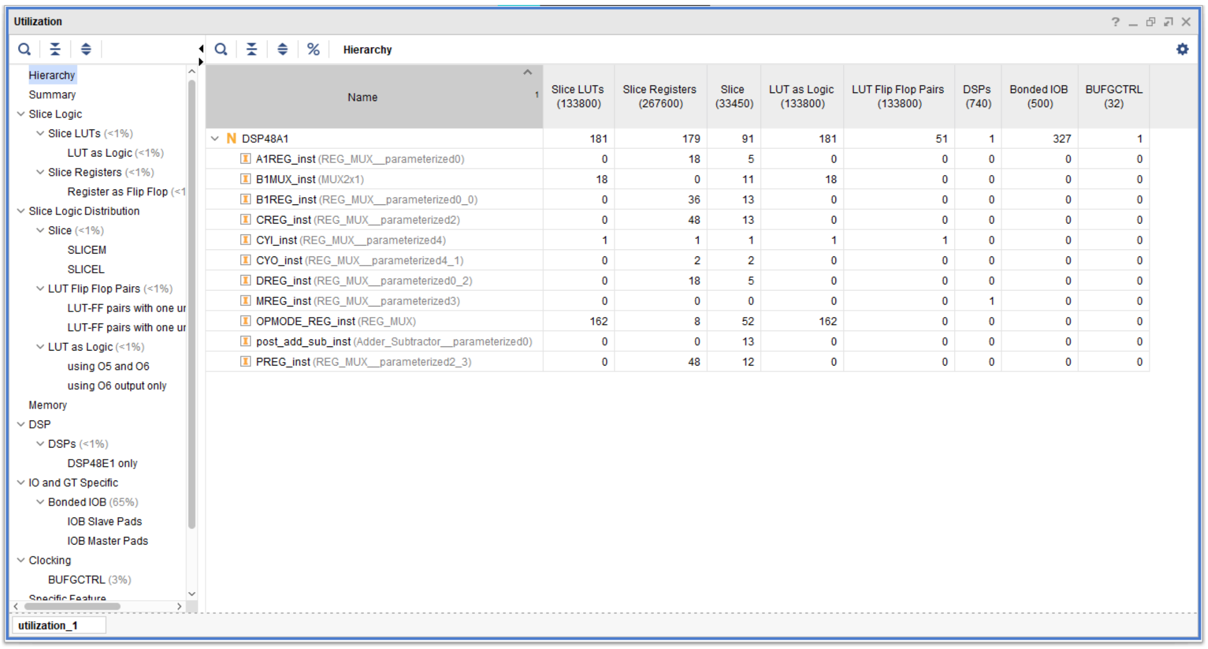
1. “Messages” tab



Synthesis (46 warnings)
Implementation (1 warning)
Route Design (1 warning)
DRC (1 warning)
Pin Planning (1 warning)

```
[DRC CFGBVS-7] CONFIG_VOLTAGE with Config Bank VCCO: The CONFIG_MODE property of current_design specifies a configuration mode (SPIx4) that uses pins in bank 14. I/O standards used in this bank have a voltage requirement of 1.80. However, the CONFIG_VOLTAGE for current_design is set to 3.3. Ensure that your configuration voltage is compatible with the I/O standards in banks used by your configuration mode. Refer to device configuration user guide for more information. Pins used by config mode: V28 (IO_L1P_T0_D00_MOSI_14), V29 (IO_L1N_T0_D01_DIN_14), V26 (IO_L2P_T0_D02_14), V27 (IO_L2N_T0_D03_14), W26 (IO_L3P_T0_DQS_PUDC_B_14), and Y27 (IO_L6P_T0_FCS_B_14)
```

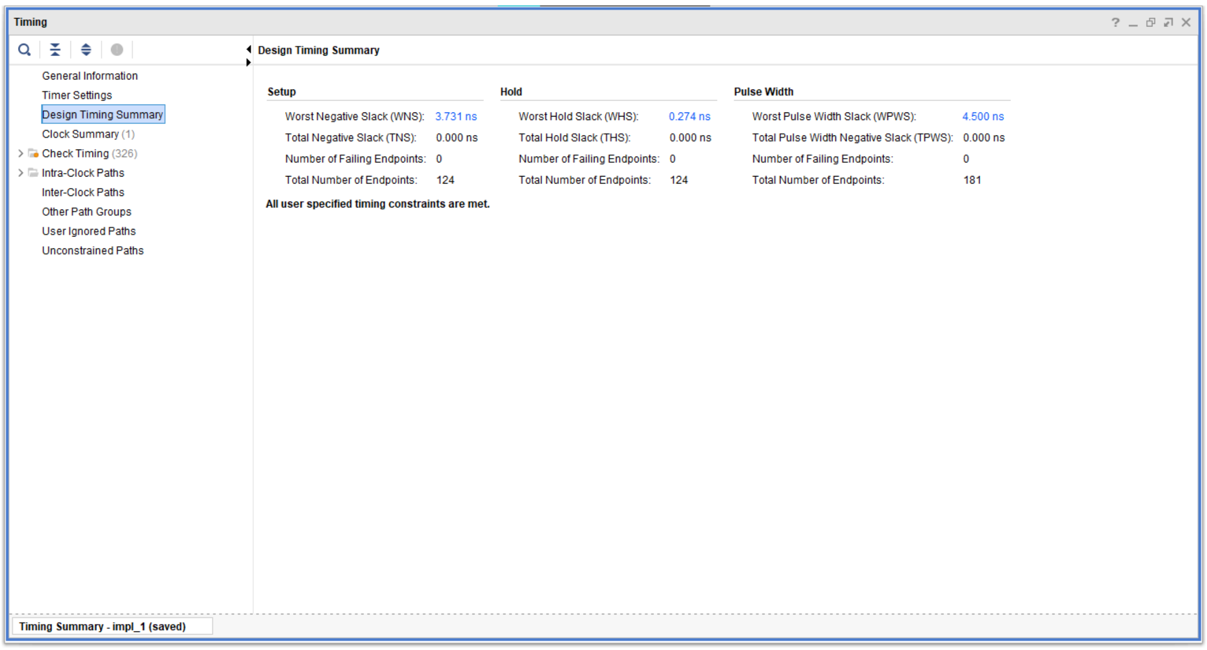
2. Utilization report



Hierarchy

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	181	179	91	181	51	1	327	1
A1REG_inst (REG_MUX__parameterized0)	0	18	5	0	0	0	0	0
B1MUX_inst (MUX2x1)	18	0	11	18	0	0	0	0
B1REG_inst (REG_MUX__parameterized0_0)	0	36	13	0	0	0	0	0
CREG_inst (REG_MUX__parameterized2)	0	48	13	0	0	0	0	0
CYI_inst (REG_MUX__parameterized4)	1	1	1	1	1	0	0	0
CYO_inst (REG_MUX__parameterized4_1)	0	2	2	0	0	0	0	0
DREG_inst (REG_MUX__parameterized0_2)	0	18	5	0	0	0	0	0
MREG_inst (REG_MUX__parameterized3)	0	0	0	0	0	1	0	0
OPMODE_REG_inst (REG_MUX)	162	8	52	162	0	0	0	0
post_add_sub_inst (Adder_Subtractor__parameterized0)	0	0	13	0	0	0	0	0
PREG_inst (REG_MUX__parameterized2_3)	0	48	12	0	0	0	0	0

3. Timing report



Timing

General Information
Timer Settings
Design Timing Summary
Clock Summary (1)
Check Timing (326)
Intra-Clock Paths
Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths

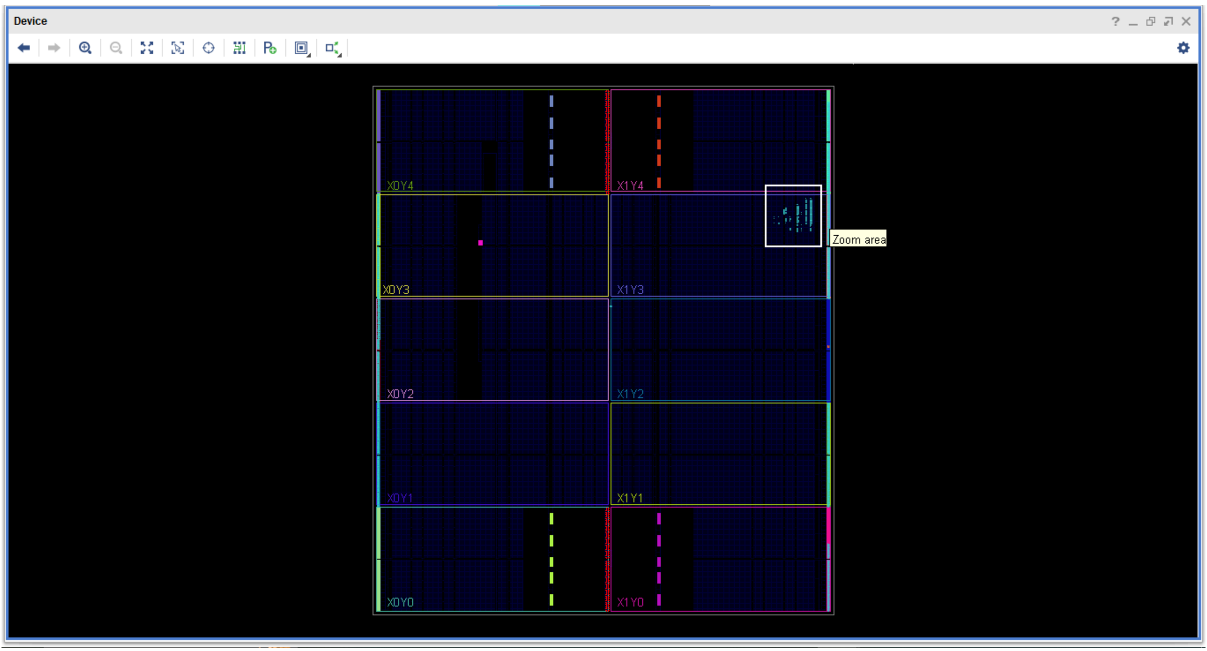
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.731 ns	Worst Hold Slack (WHS):	0.274 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:	124	Total Number of Endpoints:	124	Total Number of Endpoints:	181

All user specified timing constraints are met.

Timing Summary - impl_1 (saved)

4. Device snippet


Device
X0Y4
X1Y4
X0Y3
X1Y3
X0Y2
X1Y2
X0Y1
X1Y1
X0Y0
X1Y0
Zoom area