Design of DSP48A1 Slice

Spartan-6-FPGAs

Eng. Mina Hakim

SPARTAN6-FPGA

Supervised by: Eng. Kareem Waseem

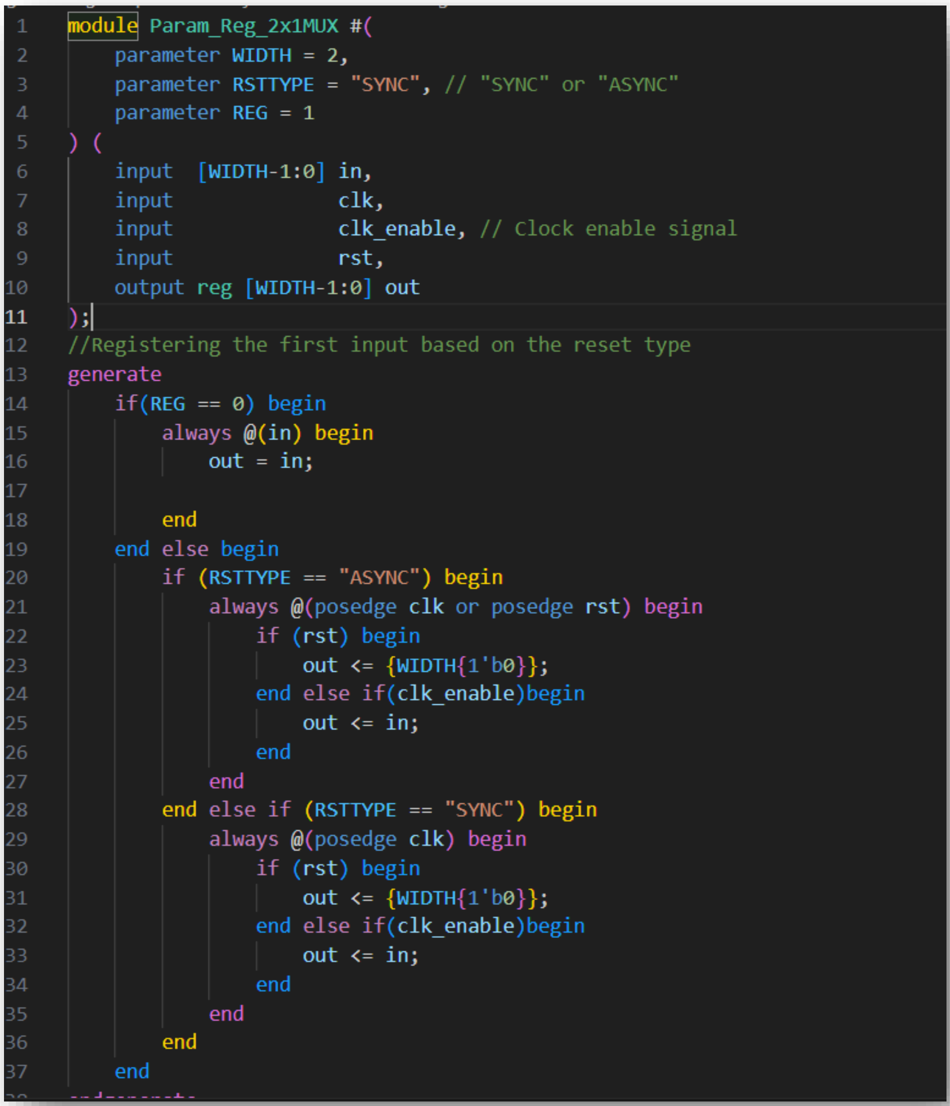
Table of Contents

Abstract:

This project presents the design and implementation of a high-performance DSP48A1 slice for Spartan-6 FPGAs, optimized for math-intensive digital signal processing applications. The DSP48A1 slice is a configurable arithmetic block capable of performing multiply-accumulate, addition, subtraction, and pre-adder operations with wide dynamic range and high precision. The design supports flexible pipeline configurations, independent clock enables, and both synchronous and asynchronous reset modes, enabling trade-offs between latency and throughput to meet diverse application requirements

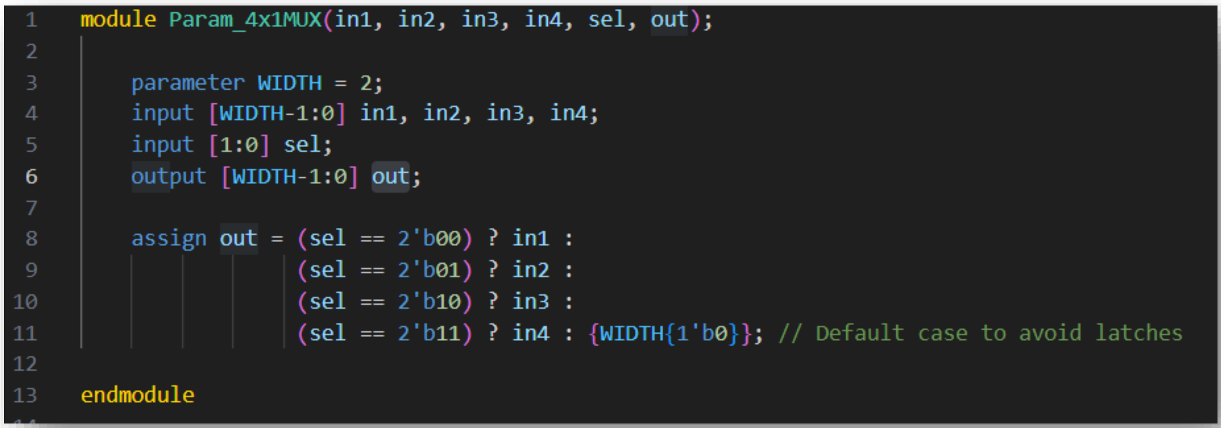
RTL code:

A. Parameterized Reg_Mux:



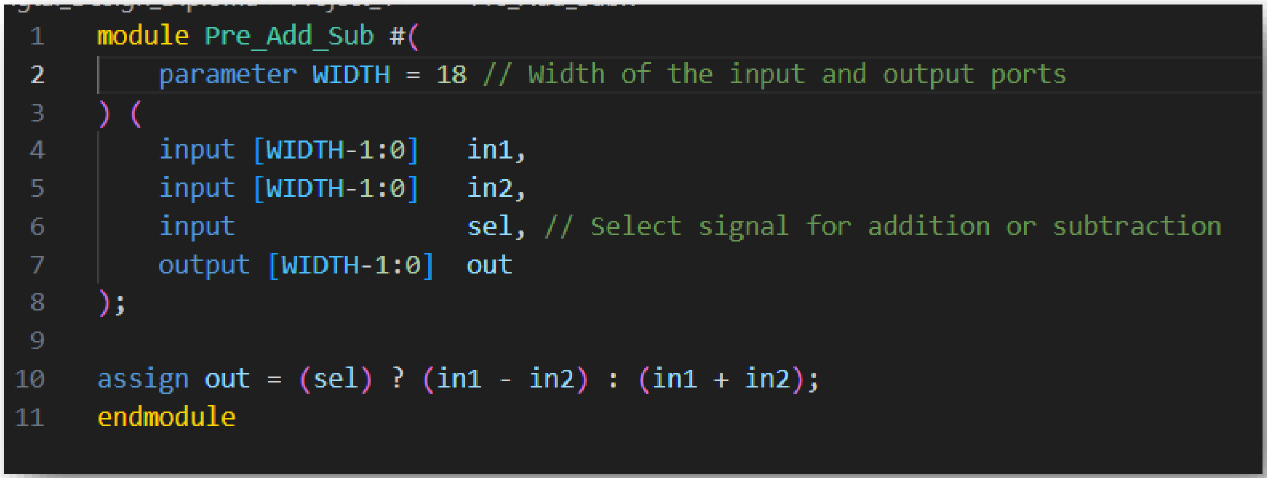
```verilog
module Param_Reg_2x1MUX #(
    parameter WIDTH = 2,
    parameter RSTTYPE = "SYNC", // "SYNC" or "ASYNC"
    parameter REG = 1
) (
    input  [WIDTH-1:0] in,
    input              clk,
    input              clk_enable, // Clock enable signal
    input              rst,
    output reg [WIDTH-1:0] out
);
//Registering the first input based on the reset type
generate
    if(REG == 0) begin
        always @(in) begin
            out = in;

        end
    end else begin
        if (RSTTYPE == "ASYNC") begin
            always @(posedge clk or posedge rst) begin
                if (rst) begin
                    out <= {WIDTH{1'b0}};
                end else if(clk_enable)begin
                    out <= in;
                end
            end
        end else if (RSTTYPE == "SYNC") begin
            always @(posedge clk) begin
                if (rst) begin
                    out <= {WIDTH{1'b0}};
                end else if(clk_enable)begin
                    out <= in;
                end
            end
        end
    end
```

B. Parameterized 4x1 MUX:



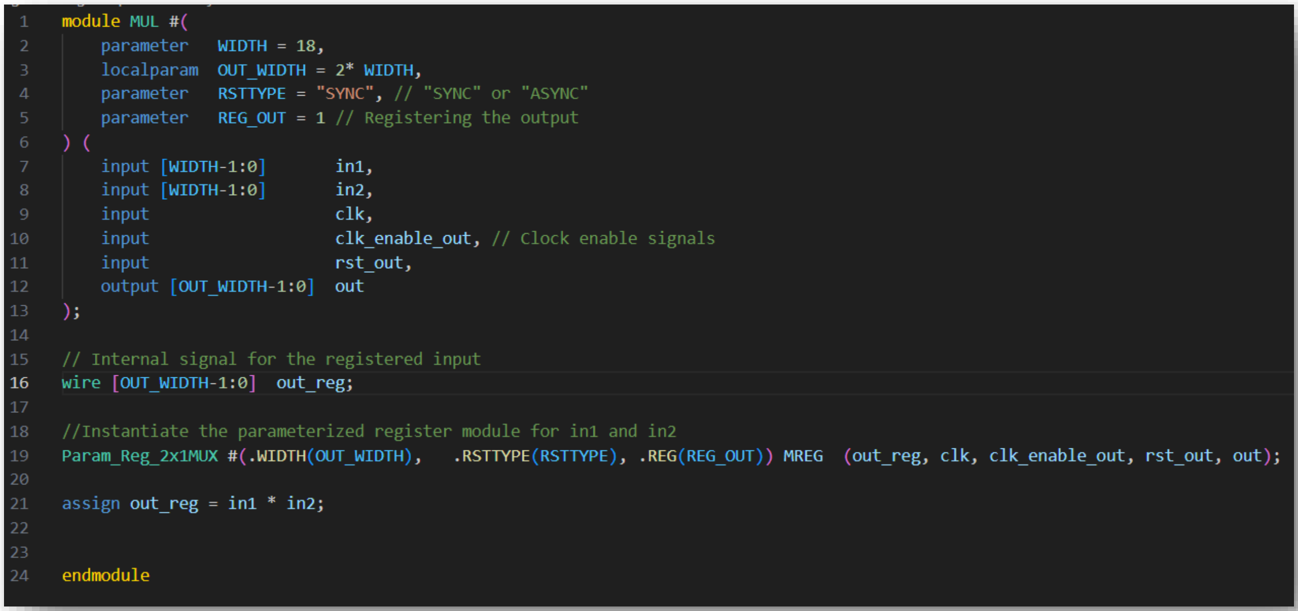
```verilog
module Param_4x1MUX(in1, in2, in3, in4, sel, out);

    parameter WIDTH = 2;
    input [WIDTH-1:0] in1, in2, in3, in4;
    input [1:0] sel;
    output [WIDTH-1:0] out;

    assign out = (sel == 2'b00) ? in1 :
                 (sel == 2'b01) ? in2 :
                 (sel == 2'b10) ? in3 :
                 (sel == 2'b11) ? in4 : {WIDTH{1'b0}}; // Default case to avoid latches

endmodule
```

C. Pre_Add_Sub:



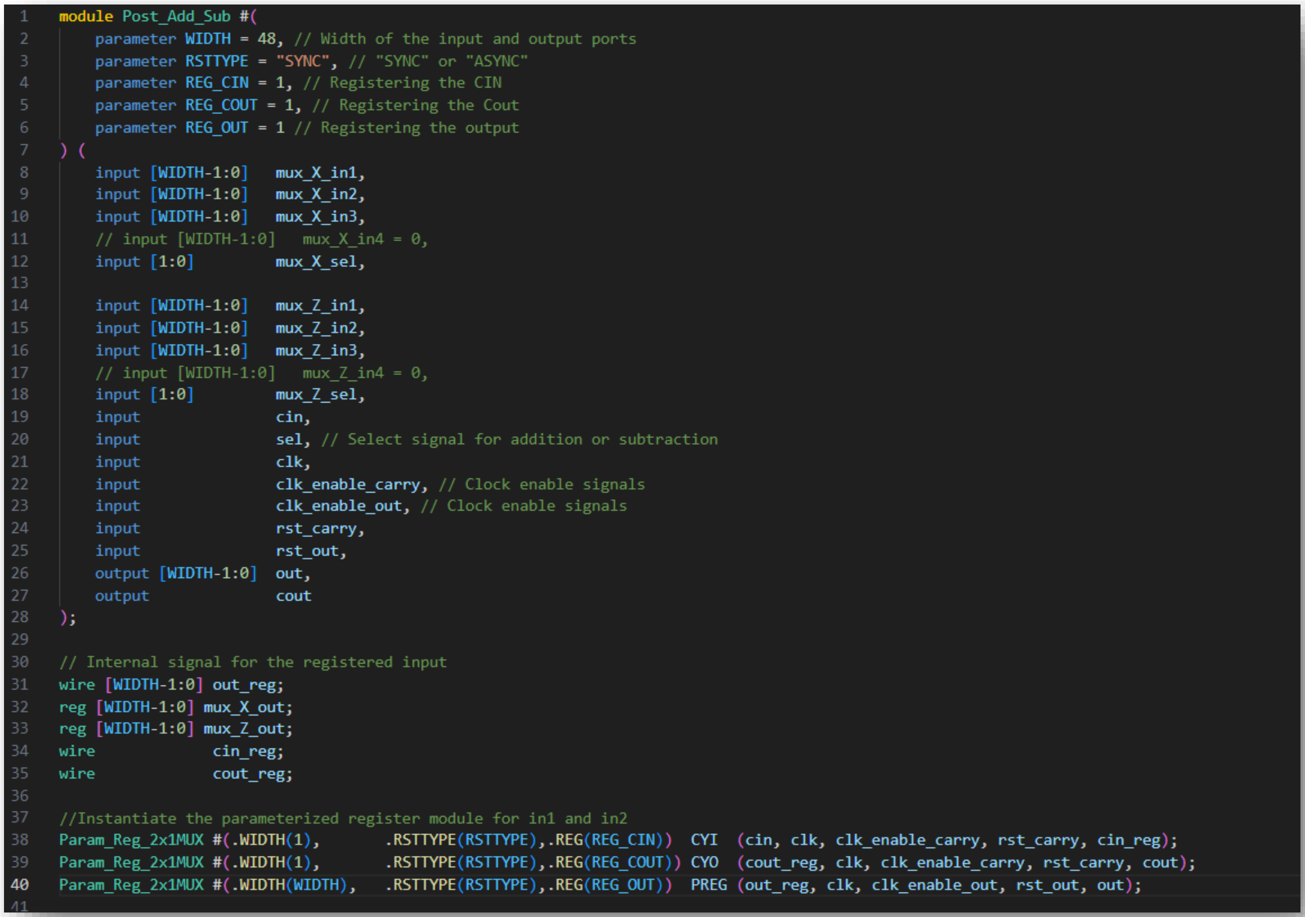
```verilog
module Pre_Add_Sub #(
    parameter WIDTH = 18 // Width of the input and output ports
) (
    input [WIDTH-1:0]   in1,
    input [WIDTH-1:0]   in2,
    input               sel, // Select signal for addition or subtraction
    output [WIDTH-1:0]  out
);

assign out = (sel) ? (in1 - in2) : (in1 + in2);
endmodule
```

D. Multiplier

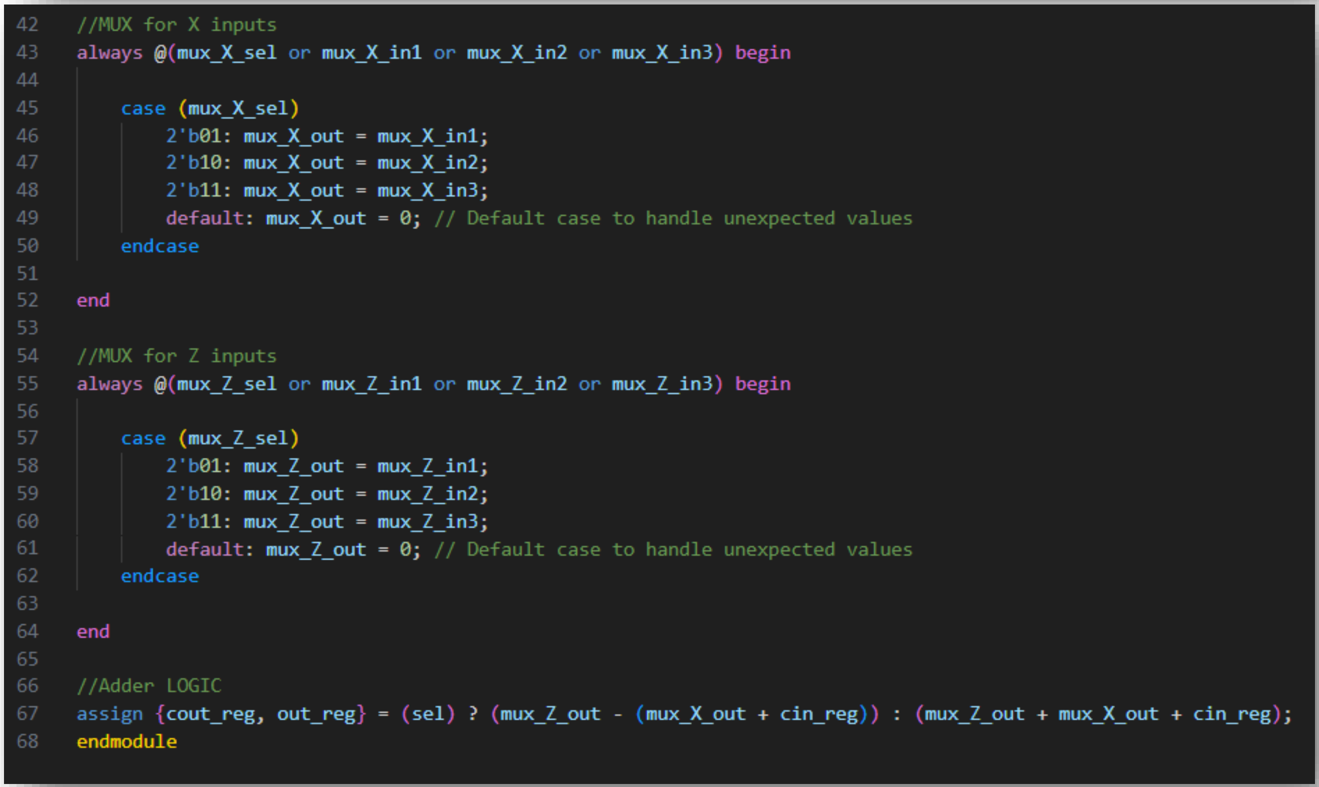


```verilog
module MUL #(
    parameter   WIDTH = 18,
    localparam  OUT_WIDTH = 2* WIDTH,
    parameter   RSTTYPE = "SYNC", // "SYNC" or "ASYNC"
    parameter   REG_OUT = 1 // Registering the output
) (
    input [WIDTH-1:0]       in1,
    input [WIDTH-1:0]       in2,
    input                   clk,
    input                   clk_enable_out, // Clock enable signals
    input                   rst_out,
    output [OUT_WIDTH-1:0]  out
);

// Internal signal for the registered input
wire [OUT_WIDTH-1:0]  out_reg;

//Instantiate the parameterized register module for in1 and in2
Param_Reg_2x1MUX #(.WIDTH(OUT_WIDTH),   .RSTTYPE(RSTTYPE), .REG(REG_OUT)) MREG  (out_reg, clk, clk_enable_out, rst_out, out);

assign out_reg = in1 * in2;


endmodule
```

E. Post_Add_Sub



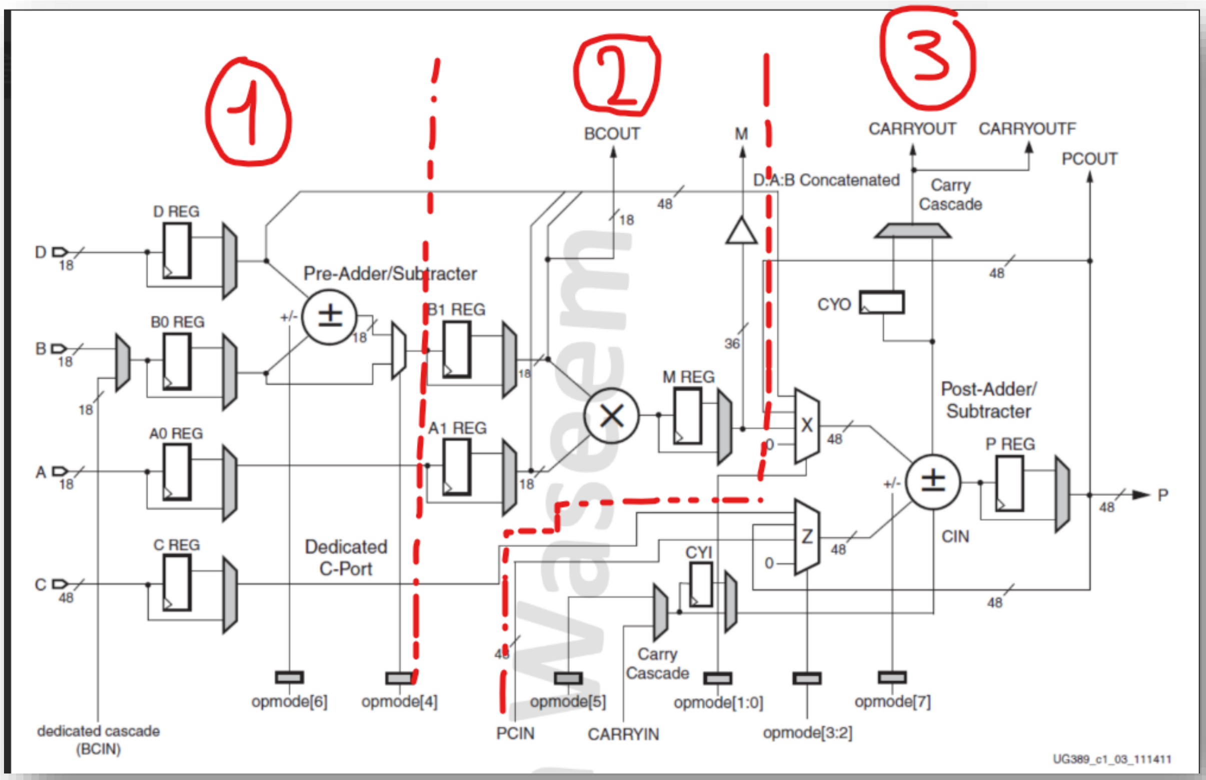
```verilog
module Post_Add_Sub #(
    parameter WIDTH = 48, // Width of the input and output ports
    parameter RSTTYPE = "SYNC", // "SYNC" or "ASYNC"
    parameter REG_CIN = 1, // Registering the CIN
    parameter REG_COUT = 1, // Registering the Cout
    parameter REG_OUT = 1 // Registering the output
) (
    input [WIDTH-1:0]   mux_X_in1,
    input [WIDTH-1:0]   mux_X_in2,
    input [WIDTH-1:0]   mux_X_in3,
    // input [WIDTH-1:0]   mux_X_in4 = 0,
    input [1:0]         mux_X_sel,

    input [WIDTH-1:0]   mux_Z_in1,
    input [WIDTH-1:0]   mux_Z_in2,
    input [WIDTH-1:0]   mux_Z_in3,
    // input [WIDTH-1:0]   mux_Z_in4 = 0,
    input [1:0]         mux_Z_sel,
    input               cin,
    input               sel, // Select signal for addition or subtraction
    input               clk,
    input               clk_enable_carry, // Clock enable signals
    input               clk_enable_out, // Clock enable signals
    input               rst_carry,
    input               rst_out,
    output [WIDTH-1:0]  out,
    output              cout
);

// Internal signal for the registered input
wire [WIDTH-1:0] out_reg;
reg [WIDTH-1:0] mux_X_out;
reg [WIDTH-1:0] mux_Z_out;
wire             cin_reg;
wire             cout_reg;

//Instantiate the parameterized register module for in1 and in2
Param_Reg_2x1MUX #(.WIDTH(1),       .RSTTYPE(RSTTYPE),.REG(REG_CIN))  CYI  (cin, clk, clk_enable_carry, rst_carry, cin_reg);
Param_Reg_2x1MUX #(.WIDTH(1),       .RSTTYPE(RSTTYPE),.REG(REG_COUT)) CYO  (cout_reg, clk, clk_enable_carry, rst_carry, cout);
Param_Reg_2x1MUX #(.WIDTH(WIDTH),   .RSTTYPE(RSTTYPE),.REG(REG_OUT))  PREG (out_reg, clk, clk_enable_out, rst_out, out);
```



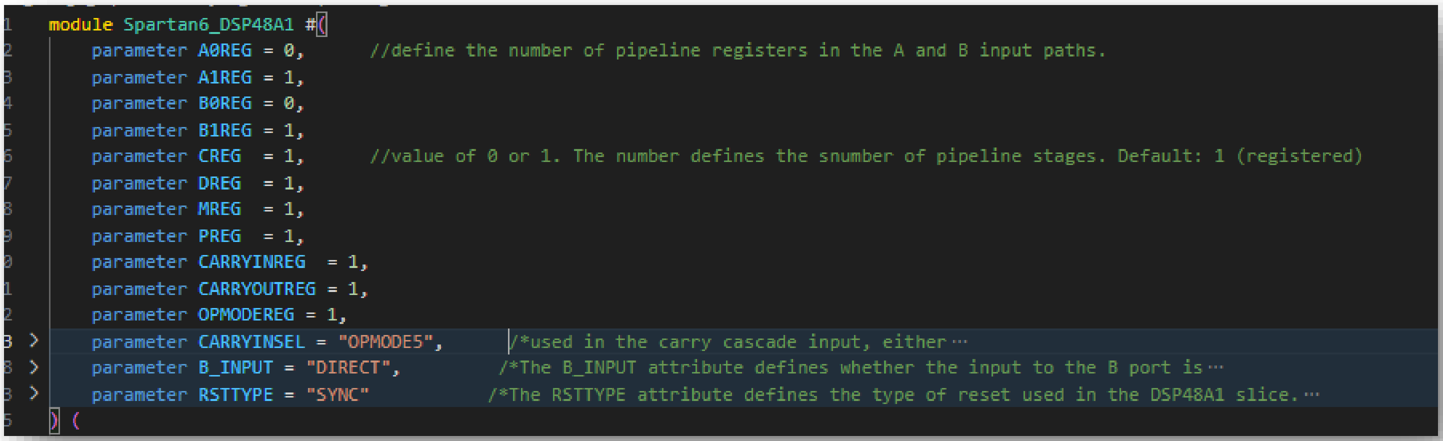
```verilog
//MUX for X inputs
always @(mux_X_sel or mux_X_in1 or mux_X_in2 or mux_X_in3) begin

    case (mux_X_sel)
        2'b01: mux_X_out = mux_X_in1;
        2'b10: mux_X_out = mux_X_in2;
        2'b11: mux_X_out = mux_X_in3;
        default: mux_X_out = 0; // Default case to handle unexpected values
    endcase

end

//MUX for Z inputs
always @(mux_Z_sel or mux_Z_in1 or mux_Z_in2 or mux_Z_in3) begin

    case (mux_Z_sel)
        2'b01: mux_Z_out = mux_Z_in1;
        2'b10: mux_Z_out = mux_Z_in2;
        2'b11: mux_Z_out = mux_Z_in3;
        default: mux_Z_out = 0; // Default case to handle unexpected values
    endcase

end

//Adder LOGIC
assign {cout_reg, out_reg} = (sel) ? (mux_Z_out - (mux_X_out + cin_reg)) : (mux_Z_out + mux_X_out + cin_reg);
endmodule
```

F. Top Module:

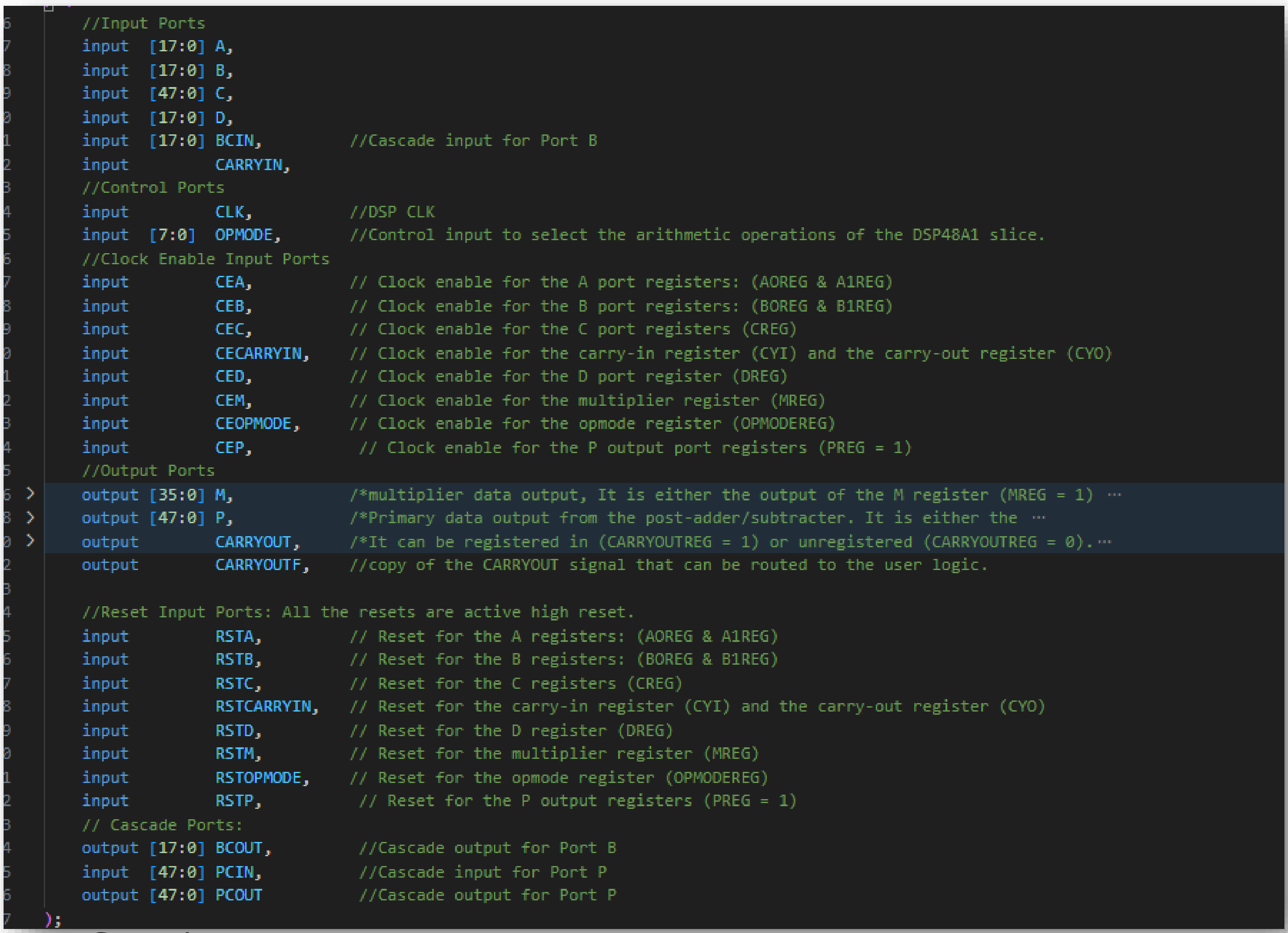
Note: The Design implementation is divide into the following Stages



Parameters and I/O ports:

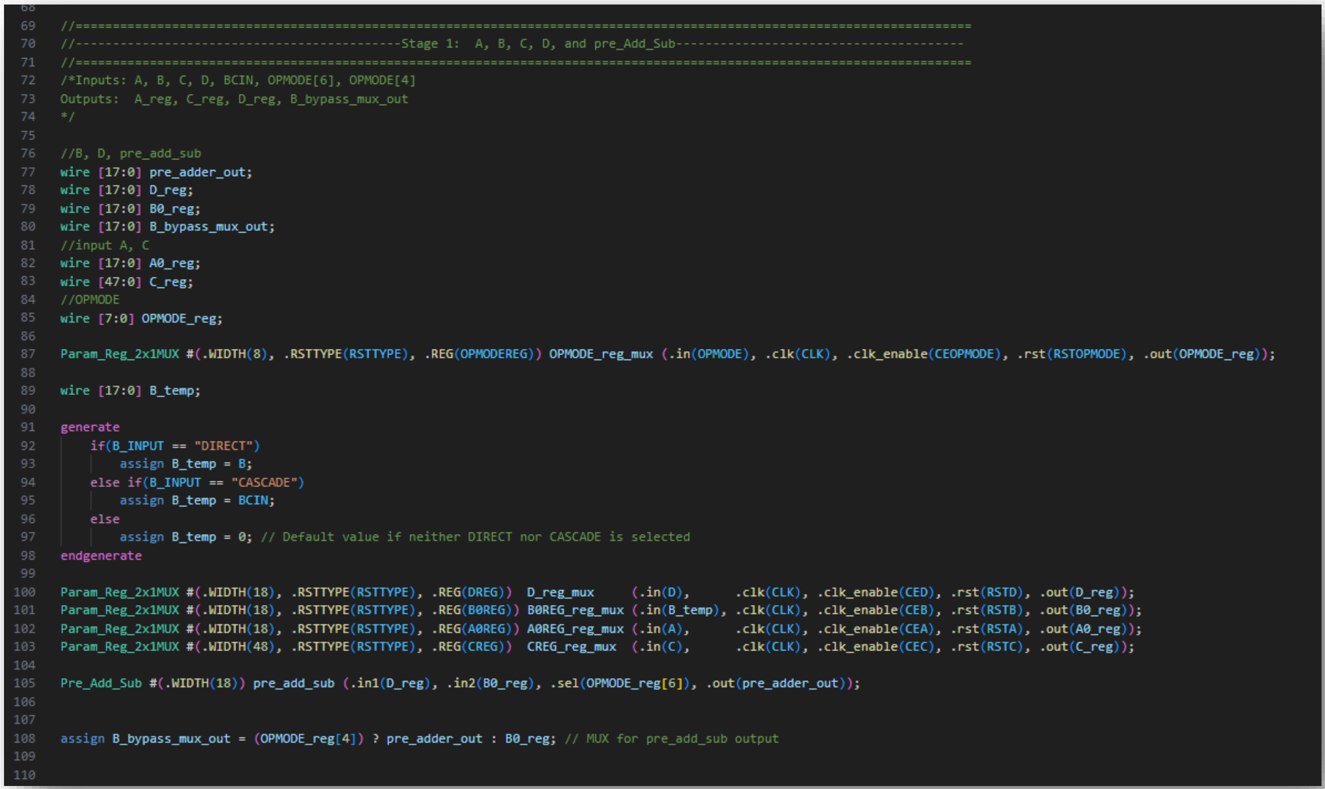


```
module Spartan6_DSP48A1 #(
    parameter A0REG = 0,      //define the number of pipeline registers in the A and B input paths.
    parameter A1REG = 1,
    parameter B0REG = 0,
    parameter B1REG = 1,
    parameter CREG  = 1,      //value of 0 or 1. The number defines the snumber of pipeline stages. Default: 1 (registered)
    parameter DREG  = 1,
    parameter MREG  = 1,
    parameter PREG  = 1,
    parameter CARRYINREG  = 1,
    parameter CARRYOUTREG = 1,
    parameter OPMODEREG = 1,
    parameter CARRYINSEL = "OPMODE5",      /*used in the carry cascade input, either⋯
    parameter B_INPUT = "DIRECT",         /*The B_INPUT attribute defines whether the input to the B port is⋯
    parameter RSTTYPE = "SYNC"           /*The RSTTYPE attribute defines the type of reset used in the DSP48A1 slice.⋯
) (
```



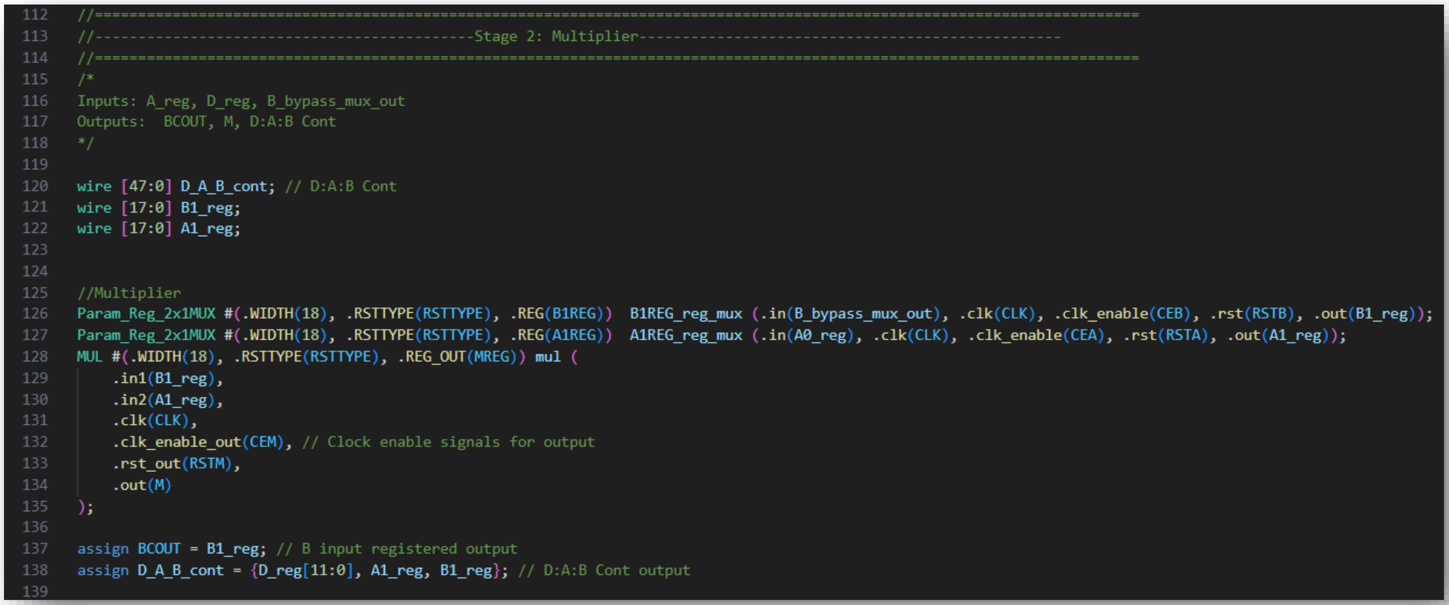
```verilog
    //Input Ports
    input  [17:0] A,
    input  [17:0] B,
    input  [47:0] C,
    input  [17:0] D,
    input  [17:0] BCIN,         //Cascade input for Port B
    input         CARRYIN,
    //Control Ports
    input         CLK,          //DSP CLK
    input  [7:0]  OPMODE,       //Control input to select the arithmetic operations of the DSP48A1 slice.
    //Clock Enable Input Ports
    input         CEA,          // Clock enable for the A port registers: (AOREG & A1REG)
    input         CEB,          // Clock enable for the B port registers: (BOREG & B1REG)
    input         CEC,          // Clock enable for the C port registers (CREG)
    input         CECARRYIN,    // Clock enable for the carry-in register (CYI) and the carry-out register (CYO)
    input         CED,          // Clock enable for the D port register (DREG)
    input         CEM,          // Clock enable for the multiplier register (MREG)
    input         CEOPMODE,     // Clock enable for the opmode register (OPMODEREG)
    input         CEP,           // Clock enable for the P output port registers (PREG = 1)
    //Output Ports
    output [35:0] M,            /*multiplier data output, It is either the output of the M register (MREG = 1) ⋯
    output [47:0] P,            /*Primary data output from the post-adder/subtracter. It is either the ⋯
    output        CARRYOUT,     /*It can be registered in (CARRYOUTREG = 1) or unregistered (CARRYOUTREG = 0).⋯
    output        CARRYOUTF,    //copy of the CARRYOUT signal that can be routed to the user logic.

    //Reset Input Ports: All the resets are active high reset.
    input         RSTA,         // Reset for the A registers: (AOREG & A1REG)
    input         RSTB,         // Reset for the B registers: (BOREG & B1REG)
    input         RSTC,         // Reset for the C registers (CREG)
    input         RSTCARRYIN,   // Reset for the carry-in register (CYI) and the carry-out register (CYO)
    input         RSTD,         // Reset for the D register (DREG)
    input         RSTM,         // Reset for the multiplier register (MREG)
    input         RSTOPMODE,    // Reset for the opmode register (OPMODEREG)
    input         RSTP,          // Reset for the P output registers (PREG = 1)
    // Cascade Ports:
    output [17:0] BCOUT,         //Cascade output for Port B
    input  [47:0] PCIN,          //Cascade input for Port P
    output [47:0] PCOUT          //Cascade output for Port P
);
```

Stage 1:



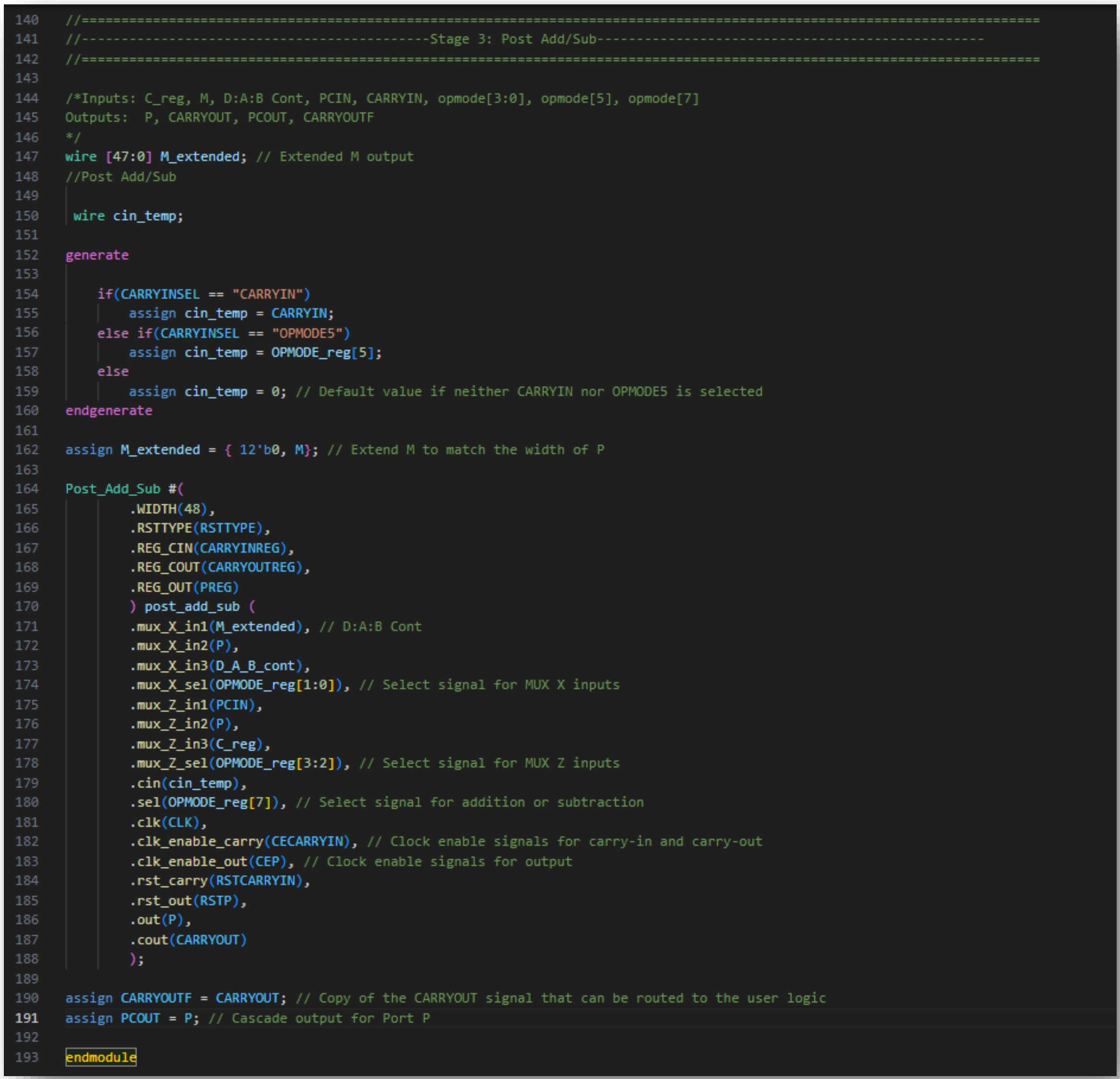
```verilog
//=====================================================================================================
//-------------------------------------------Stage 1:  A, B, C, D, and pre_Add_Sub-------------------------------------
//=====================================================================================================
/*Inputs: A, B, C, D, BCIN, OPMODE[6], OPMODE[4]
Outputs:  A_reg, C_reg, D_reg, B_bypass_mux_out
*/

//B, D, pre_add_sub
wire [17:0] pre_adder_out;
wire [17:0] D_reg;
wire [17:0] B0_reg;
wire [17:0] B_bypass_mux_out;
//input A, C
wire [17:0] A0_reg;
wire [47:0] C_reg;
//OPMODE
wire [7:0] OPMODE_reg;

Param_Reg_2x1MUX #(.WIDTH(8), .RSTTYPE(RSTTYPE), .REG(OPMODEREG)) OPMODE_reg_mux (.in(OPMODE), .clk(CLK), .clk_enable(CEOPMODE), .rst(RSTOPMODE), .out(OPMODE_reg));

wire [17:0] B_temp;

generate
    if(B_INPUT == "DIRECT")
        assign B_temp = B;
    else if(B_INPUT == "CASCADE")
        assign B_temp = BCIN;
    else
        assign B_temp = 0; // Default value if neither DIRECT nor CASCADE is selected
endgenerate

Param_Reg_2x1MUX #(.WIDTH(18), .RSTTYPE(RSTTYPE), .REG(DREG))  D_reg_mux     (.in(D),      .clk(CLK), .clk_enable(CED), .rst(RSTD), .out(D_reg));
Param_Reg_2x1MUX #(.WIDTH(18), .RSTTYPE(RSTTYPE), .REG(B0REG)) B0REG_reg_mux (.in(B_temp), .clk(CLK), .clk_enable(CEB), .rst(RSTB), .out(B0_reg));
Param_Reg_2x1MUX #(.WIDTH(18), .RSTTYPE(RSTTYPE), .REG(A0REG)) A0REG_reg_mux (.in(A),      .clk(CLK), .clk_enable(CEA), .rst(RSTA), .out(A0_reg));
Param_Reg_2x1MUX #(.WIDTH(48), .RSTTYPE(RSTTYPE), .REG(CREG))  CREG_reg_mux  (.in(C),      .clk(CLK), .clk_enable(CEC), .rst(RSTC), .out(C_reg));

Pre_Add_Sub #(.WIDTH(18)) pre_add_sub (.in1(D_reg), .in2(B0_reg), .sel(OPMODE_reg[6]), .out(pre_adder_out));


assign B_bypass_mux_out = (OPMODE_reg[4]) ? pre_adder_out : B0_reg; // MUX for pre_add_sub output

```

Stage 2:



```verilog
//==============================================================================================================
//-------------------------------------------Stage 2: Multiplier---------------------------------------------
//==============================================================================================================
/*
Inputs: A_reg, D_reg, B_bypass_mux_out
Outputs:  BCOUT, M, D:A:B Cont
*/

wire [47:0] D_A_B_cont; // D:A:B Cont
wire [17:0] B1_reg;
wire [17:0] A1_reg;


//Multiplier
Param_Reg_2x1MUX #(.WIDTH(18), .RSTTYPE(RSTTYPE), .REG(B1REG))  B1REG_reg_mux (.in(B_bypass_mux_out), .clk(CLK), .clk_enable(CEB), .rst(RSTB), .out(B1_reg));
Param_Reg_2x1MUX #(.WIDTH(18), .RSTTYPE(RSTTYPE), .REG(A1REG))  A1REG_reg_mux (.in(A0_reg), .clk(CLK), .clk_enable(CEA), .rst(RSTA), .out(A1_reg));
MUL #(.WIDTH(18), .RSTTYPE(RSTTYPE), .REG_OUT(MREG)) mul (
    .in1(B1_reg),
    .in2(A1_reg),
    .clk(CLK),
    .clk_enable_out(CEM), // Clock enable signals for output
    .rst_out(RSTM),
    .out(M)
);

assign BCOUT = B1_reg; // B input registered output
assign D_A_B_cont = {D_reg[11:0], A1_reg, B1_reg}; // D:A:B Cont output

```

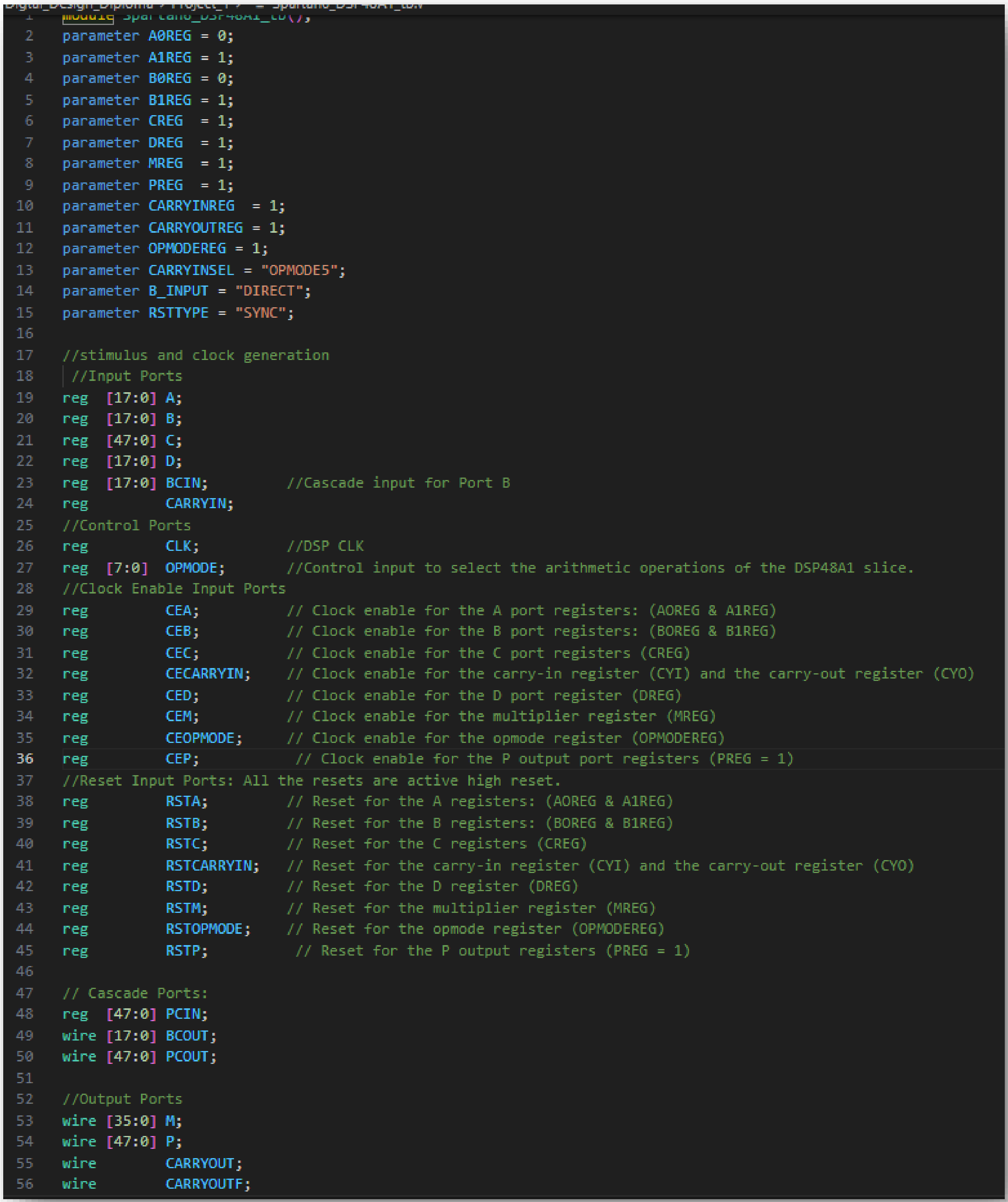
Stage 3:



```verilog
//==============================================================================================================
//-------------------------------------------Stage 3: Post Add/Sub-----------------------------------------------
//==============================================================================================================

/*Inputs: C_reg, M, D:A:B Cont, PCIN, CARRYIN, opmode[3:0], opmode[5], opmode[7]
Outputs:  P, CARRYOUT, PCOUT, CARRYOUTF
*/
wire [47:0] M_extended; // Extended M output
//Post Add/Sub

 wire cin_temp;

generate

    if(CARRYINSEL == "CARRYIN")
        assign cin_temp = CARRYIN;
    else if(CARRYINSEL == "OPMODE5")
        assign cin_temp = OPMODE_reg[5];
    else
        assign cin_temp = 0; // Default value if neither CARRYIN nor OPMODE5 is selected
endgenerate

assign M_extended = { 12'b0, M}; // Extend M to match the width of P

Post_Add_Sub #(
        .WIDTH(48),
        .RSTTYPE(RSTTYPE),
        .REG_CIN(CARRYINREG),
        .REG_COUT(CARRYOUTREG),
        .REG_OUT(PREG)
        ) post_add_sub (
        .mux_X_in1(M_extended), // D:A:B Cont
        .mux_X_in2(P),
        .mux_X_in3(D_A_B_cont),
        .mux_X_sel(OPMODE_reg[1:0]), // Select signal for MUX X inputs
        .mux_Z_in1(PCIN),
        .mux_Z_in2(P),
        .mux_Z_in3(C_reg),
        .mux_Z_sel(OPMODE_reg[3:2]), // Select signal for MUX Z inputs
        .cin(cin_temp),
        .sel(OPMODE_reg[7]), // Select signal for addition or subtraction
        .clk(CLK),
        .clk_enable_carry(CECARRYIN), // Clock enable signals for carry-in and carry-out
        .clk_enable_out(CEP), // Clock enable signals for output
        .rst_carry(RSTCARRYIN),
        .rst_out(RSTP),
        .out(P),
        .cout(CARRYOUT)
        );

assign CARRYOUTF = CARRYOUT; // Copy of the CARRYOUT signal that can be routed to the user logic
assign PCOUT = P; // Cascade output for Port P

endmodule
```

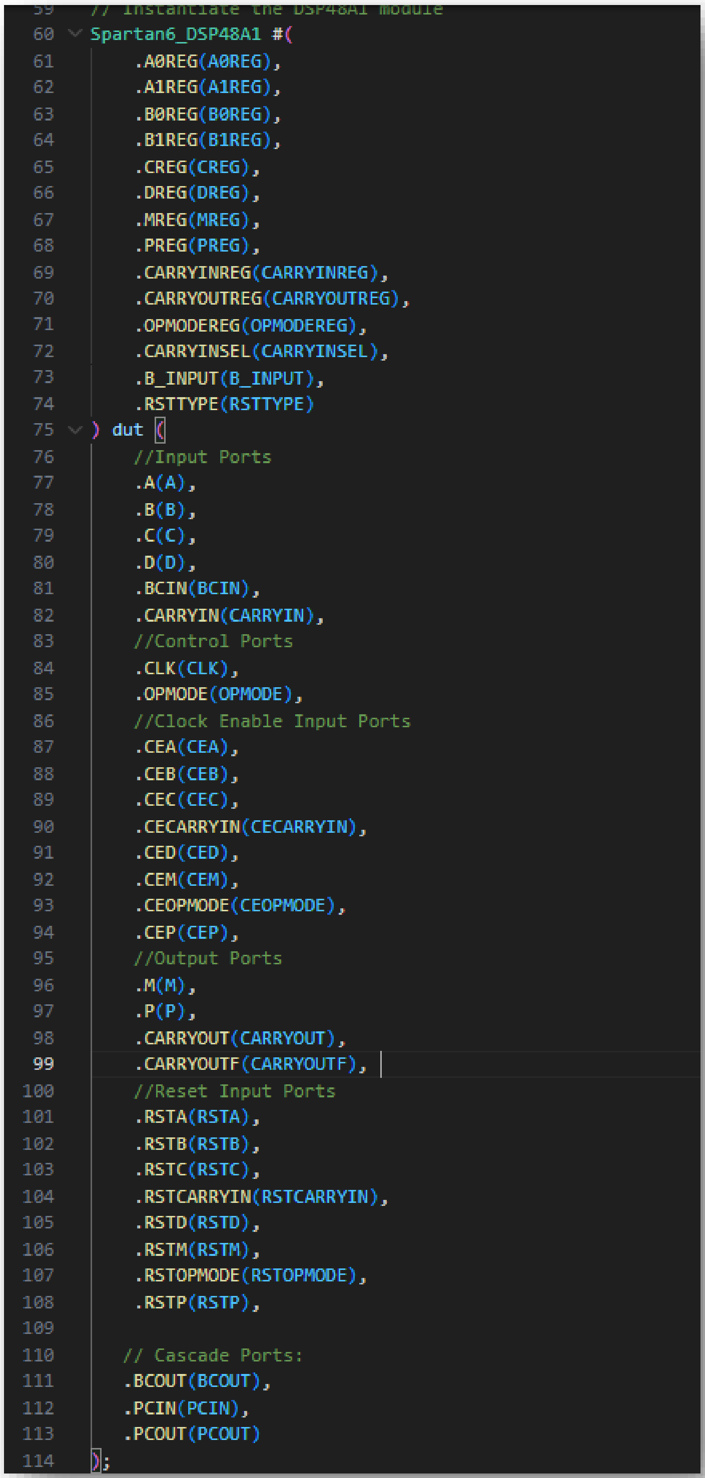
1. Test Bench Code:

Parameters and Stimulus declaration:

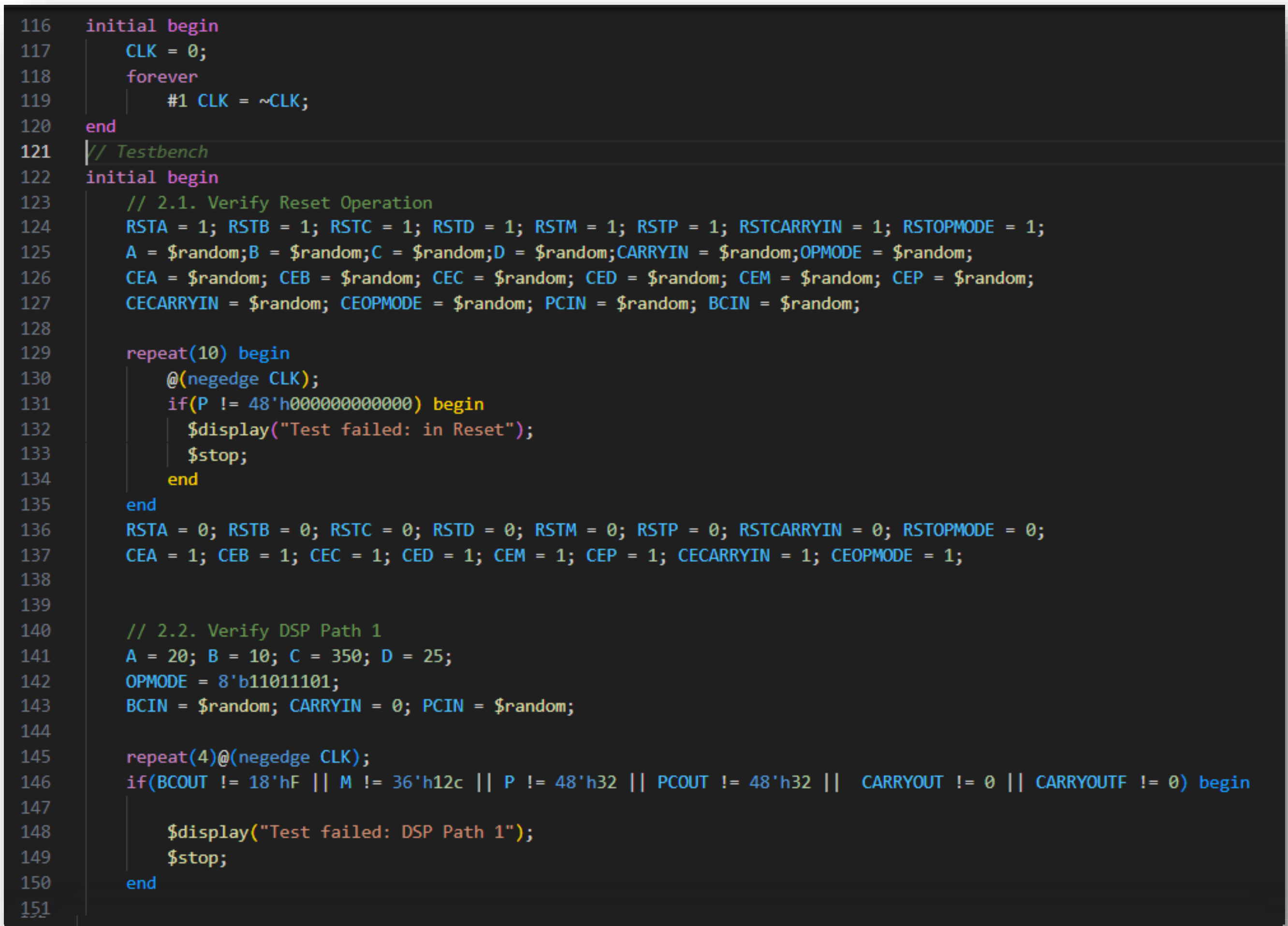


```verilog
module Spartan6_DSP48A1_tb();
parameter A0REG = 0;
parameter A1REG = 1;
parameter B0REG = 0;
parameter B1REG = 1;
parameter CREG  = 1;
parameter DREG  = 1;
parameter MREG  = 1;
parameter PREG  = 1;
parameter CARRYINREG  = 1;
parameter CARRYOUTREG = 1;
parameter OPMODEREG = 1;
parameter CARRYINSEL = "OPMODE5";
parameter B_INPUT = "DIRECT";
parameter RSTTYPE = "SYNC";

//stimulus and clock generation
 //Input Ports
reg  [17:0] A;
reg  [17:0] B;
reg  [47:0] C;
reg  [17:0] D;
reg  [17:0] BCIN;         //Cascade input for Port B
reg         CARRYIN;
//Control Ports
reg         CLK;          //DSP CLK
reg  [7:0]  OPMODE;       //Control input to select the arithmetic operations of the DSP48A1 slice.
//Clock Enable Input Ports
reg         CEA;          // Clock enable for the A port registers: (A0REG & A1REG)
reg         CEB;          // Clock enable for the B port registers: (B0REG & B1REG)
reg         CEC;          // Clock enable for the C port registers (CREG)
reg         CECARRYIN;    // Clock enable for the carry-in register (CYI) and the carry-out register (CYO)
reg         CED;          // Clock enable for the D port register (DREG)
reg         CEM;          // Clock enable for the multiplier register (MREG)
reg         CEOPMODE;     // Clock enable for the opmode register (OPMODEREG)
reg         CEP;           // Clock enable for the P output port registers (PREG = 1)
//Reset Input Ports: All the resets are active high reset.
reg         RSTA;         // Reset for the A registers: (A0REG & A1REG)
reg         RSTB;         // Reset for the B registers: (B0REG & B1REG)
reg         RSTC;         // Reset for the C registers (CREG)
reg         RSTCARRYIN;   // Reset for the carry-in register (CYI) and the carry-out register (CYO)
reg         RSTD;         // Reset for the D register (DREG)
reg         RSTM;         // Reset for the multiplier register (MREG)
reg         RSTOPMODE;    // Reset for the opmode register (OPMODEREG)
reg         RSTP;          // Reset for the P output registers (PREG = 1)

// Cascade Ports:
reg  [47:0] PCIN;
wire [17:0] BCOUT;
wire [47:0] PCOUT;

//Output Ports
wire [35:0] M;
wire [47:0] P;
wire        CARRYOUT;
wire        CARRYOUTF;
```

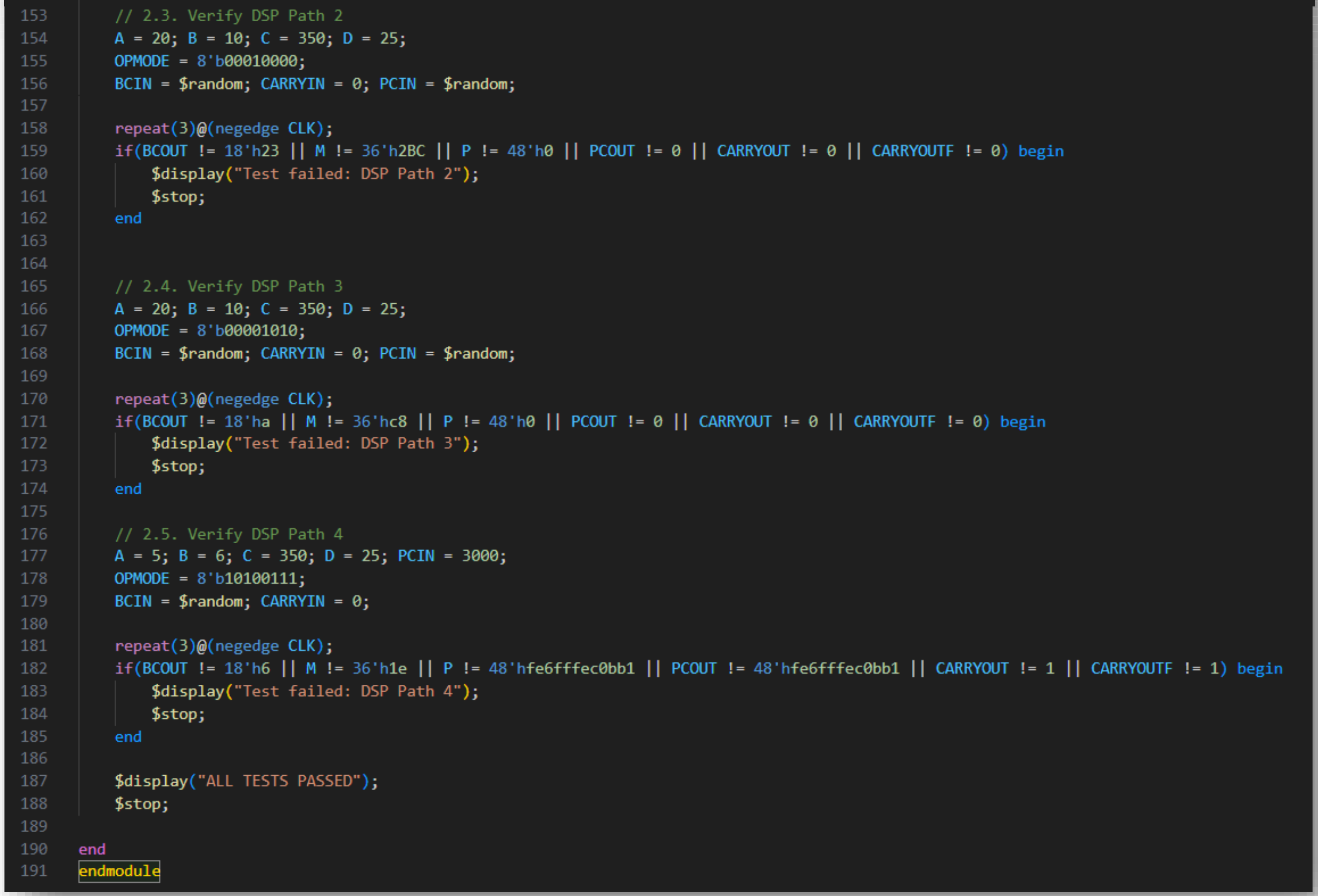
Instantiation:



```verilog
// Instantiate the DSP48A1 module
Spartan6_DSP48A1 #(
    .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 (
    //Input Ports
    .A(A),
    .B(B),
    .C(C),
    .D(D),
    .BCIN(BCIN),
    .CARRYIN(CARRYIN),
    //Control Ports
    .CLK(CLK),
    .OPMODE(OPMODE),
    //Clock Enable Input Ports
    .CEA(CEA),
    .CEB(CEB),
    .CEC(CEC),
    .CECARRYIN(CECARRYIN),
    .CED(CED),
    .CEM(CEM),
    .CEOPMODE(CEOPMODE),
    .CEP(CEP),
    //Output Ports
    .M(M),
    .P(P),
    .CARRYOUT(CARRYOUT),
    .CARRYOUTF(CARRYOUTF),
    //Reset Input Ports
    .RSTA(RSTA),
    .RSTB(RSTB),
    .RSTC(RSTC),
    .RSTCARRYIN(RSTCARRYIN),
    .RSTD(RSTD),
    .RSTM(RSTM),
    .RSTOPMODE(RSTOPMODE),
    .RSTP(RSTP),

   // Cascade Ports:
   .BCOUT(BCOUT),
   .PCIN(PCIN),
   .PCOUT(PCOUT)
);
```

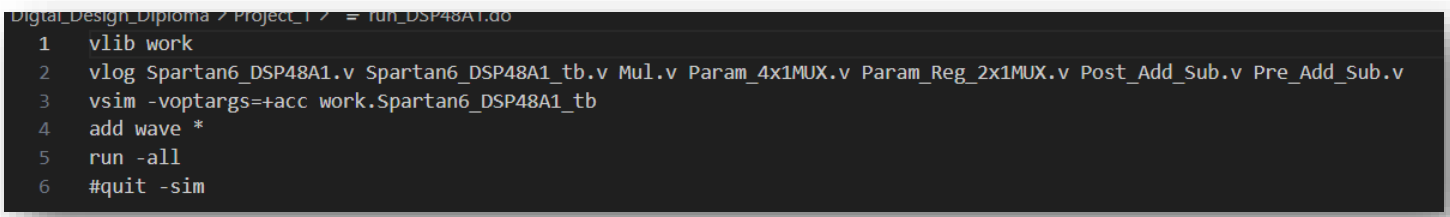


```
initial begin
    CLK = 0;
    forever
        #1 CLK = ~CLK;
end
// Testbench
initial begin
    // 2.1. Verify Reset Operation
    RSTA = 1; RSTB = 1; RSTC = 1; RSTD = 1; RSTM = 1; RSTP = 1; RSTCARRYIN = 1; RSTOPMODE = 1;
    A = $random;B = $random;C = $random;D = $random;CARRYIN = $random;OPMODE = $random;
    CEA = $random; CEB = $random; CEC = $random; CED = $random; CEM = $random; CEP = $random;
    CECARRYIN = $random; CEOPMODE = $random; PCIN = $random; BCIN = $random;

    repeat(10) begin
        @(negedge CLK);
        if(P != 48'h000000000000) begin
          $display("Test failed: in Reset");
          $stop;
        end
    end
    RSTA = 0; RSTB = 0; RSTC = 0; RSTD = 0; RSTM = 0; RSTP = 0; RSTCARRYIN = 0; RSTOPMODE = 0;
    CEA = 1; CEB = 1; CEC = 1; CED = 1; CEM = 1; CEP = 1; CECARRYIN = 1; CEOPMODE = 1;


    // 2.2. Verify DSP Path 1
    A = 20; B = 10; C = 350; D = 25;
    OPMODE = 8'b11011101;
    BCIN = $random; CARRYIN = 0; PCIN = $random;

    repeat(4)@(negedge CLK);
    if(BCOUT != 18'hF || M != 36'h12c || P != 48'h32 || PCOUT != 48'h32 ||  CARRYOUT != 0 || CARRYOUTF != 0) begin

        $display("Test failed: DSP Path 1");
        $stop;
    end

```



```
    // 2.3. Verify DSP Path 2
    A = 20; B = 10; C = 350; D = 25;
    OPMODE = 8'b00010000;
    BCIN = $random; CARRYIN = 0; PCIN = $random;

    repeat(3)@(negedge CLK);
    if(BCOUT != 18'h23 || M != 36'h2BC || P != 48'h0 || PCOUT != 0 || CARRYOUT != 0 || CARRYOUTF != 0) begin
        $display("Test failed: DSP Path 2");
        $stop;
    end


    // 2.4. Verify DSP Path 3
    A = 20; B = 10; C = 350; D = 25;
    OPMODE = 8'b00001010;
    BCIN = $random; CARRYIN = 0; PCIN = $random;

    repeat(3)@(negedge CLK);
    if(BCOUT != 18'ha || M != 36'hc8 || P != 48'h0 || PCOUT != 0 || CARRYOUT != 0 || CARRYOUTF != 0) begin
        $display("Test failed: DSP Path 3");
        $stop;
    end

    // 2.5. Verify DSP Path 4
    A = 5; B = 6; C = 350; D = 25; PCIN = 3000;
    OPMODE = 8'b10100111;
    BCIN = $random; CARRYIN = 0;

    repeat(3)@(negedge CLK);
    if(BCOUT != 18'h6 || M != 36'h1e || P != 48'hfe6fffec0bb1 || PCOUT != 48'hfe6fffec0bb1 || CARRYOUT != 1 || CARRYOUTF != 1) begin
        $display("Test failed: DSP Path 4");
        $stop;
    end

    $display("ALL TESTS PASSED");
    $stop;

end
endmodule
```

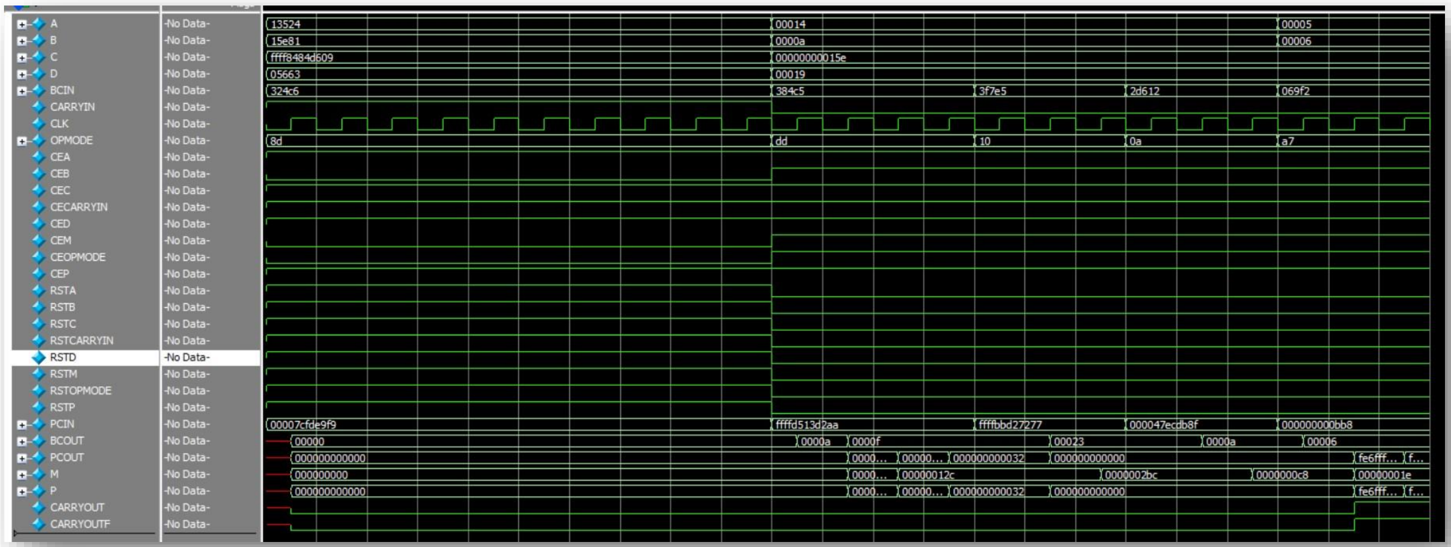
2. Do File:



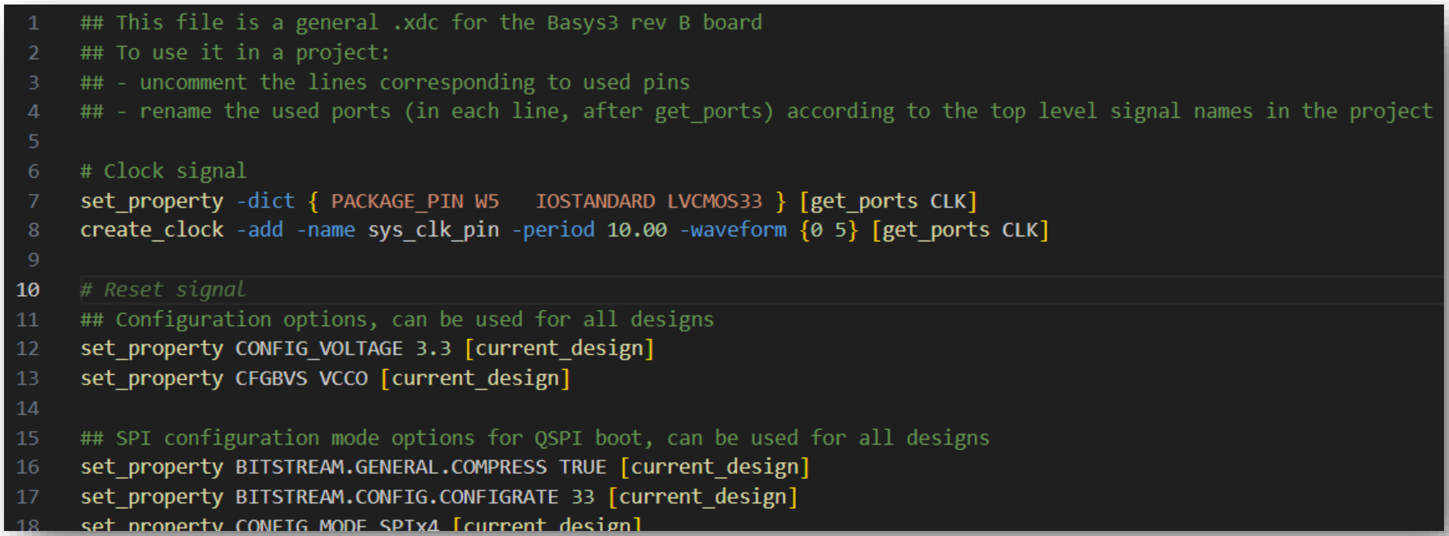
Digital_Design_Diploma > Project_1 > ≡ run_DSP48A1.do

```
vlib work
vlog Spartan6_DSP48A1.v Spartan6_DSP48A1_tb.v Mul.v Param_4x1MUX.v Param_Reg_2x1MUX.v Post_Add_Sub.v Pre_Add_Sub.v
vsim -voptargs=+acc work.Spartan6_DSP48A1_tb
add wave *
run -all
#quit -sim
```

3. QuestaSim Snippets



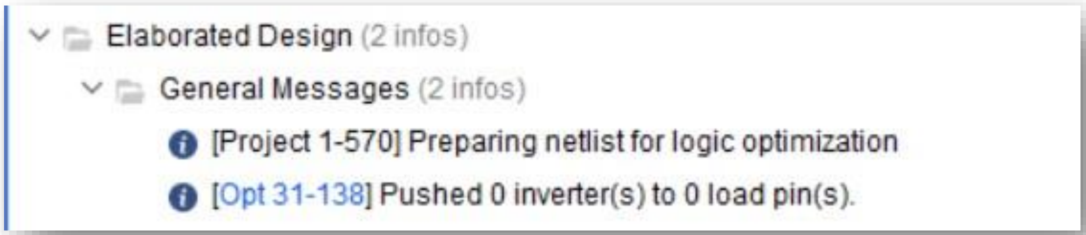
4. Constraint File



```
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project

# 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]

# Reset signal
## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [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]
```

5. Elaboration:

Message Tab:



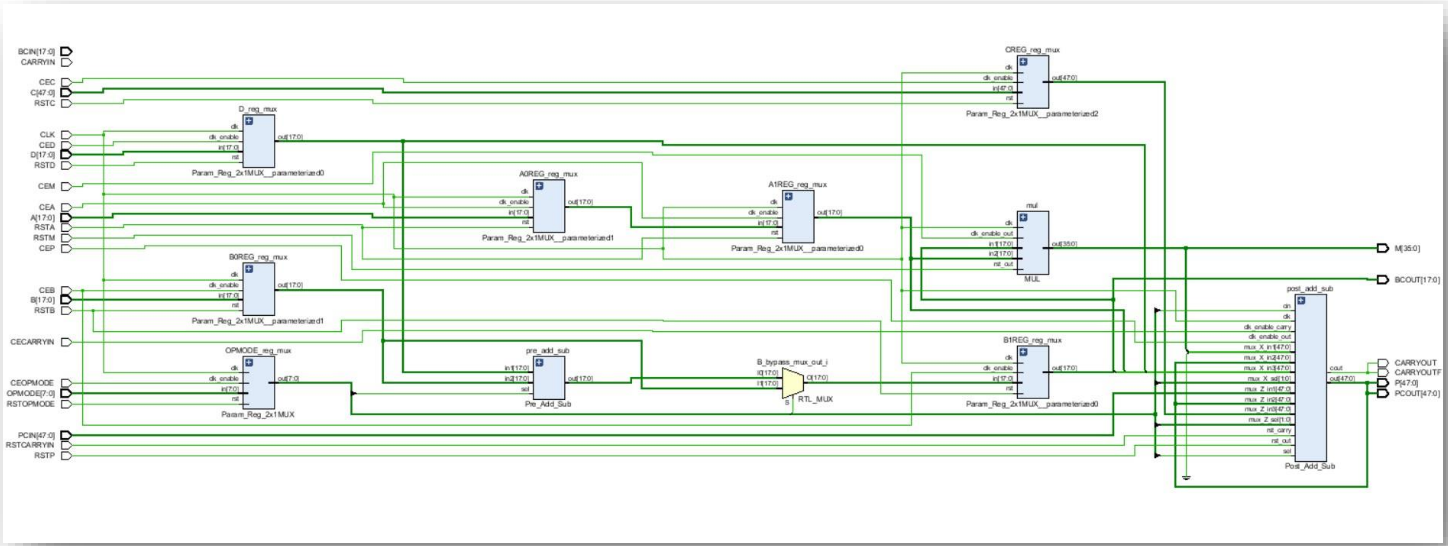
Elaborated Design (2 infos)

General Messages (2 infos)

[Project 1-570] Preparing netlist for logic optimization

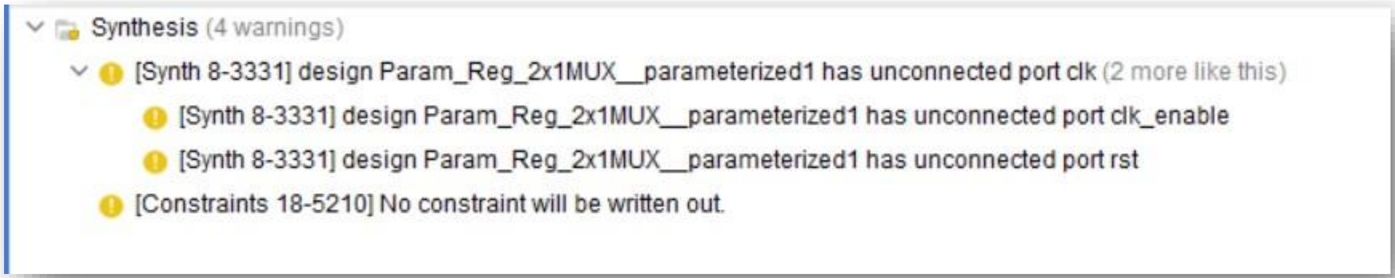
[Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).

Schematic:



6. Synthesis

Message Tab:



Synthesis (4 warnings)

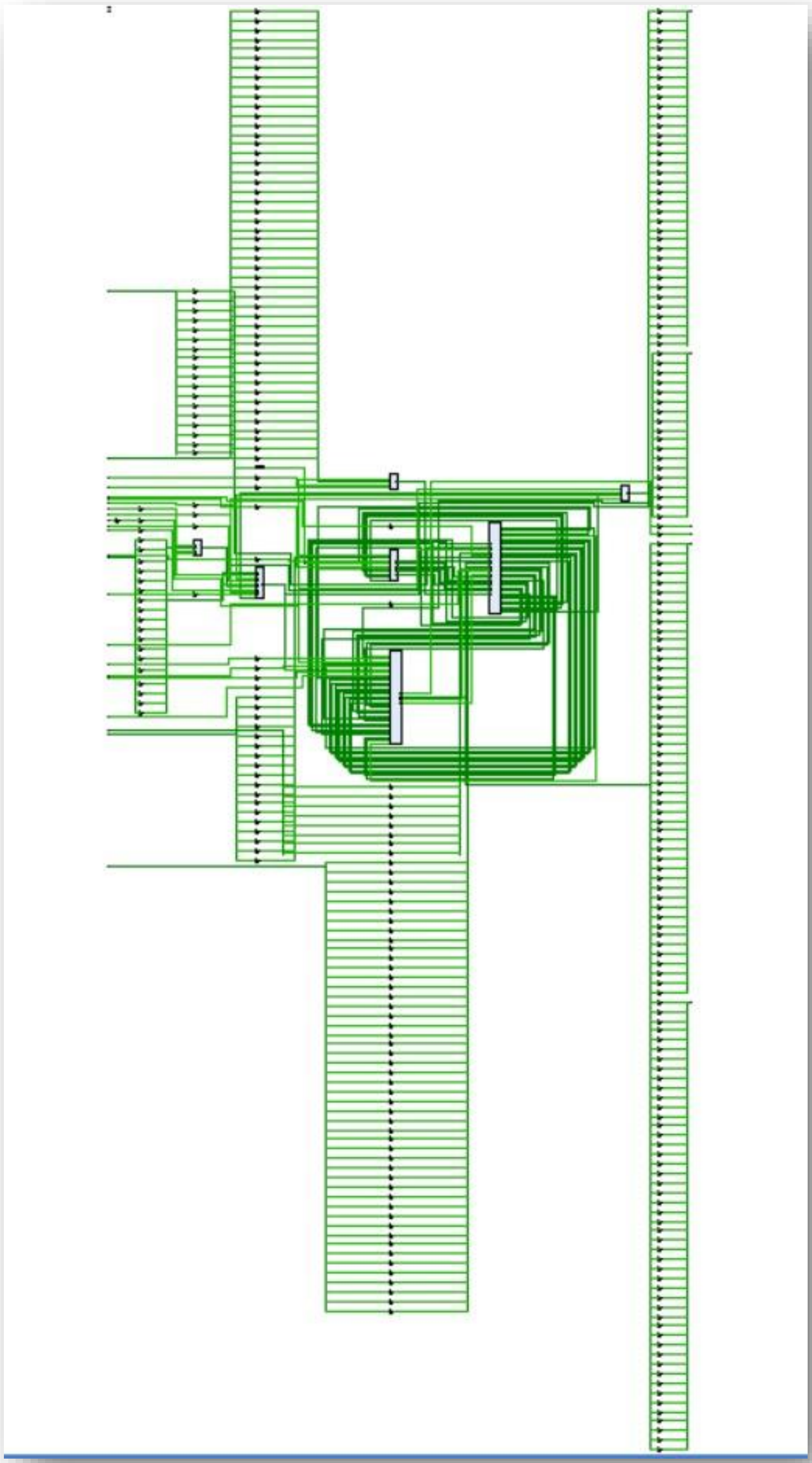
[Synth 8-3331] design Param_Reg_2x1MUX__parameterized1 has unconnected port clk (2 more like this)

[Synth 8-3331] design Param_Reg_2x1MUX__parameterized1 has unconnected port clk_enable

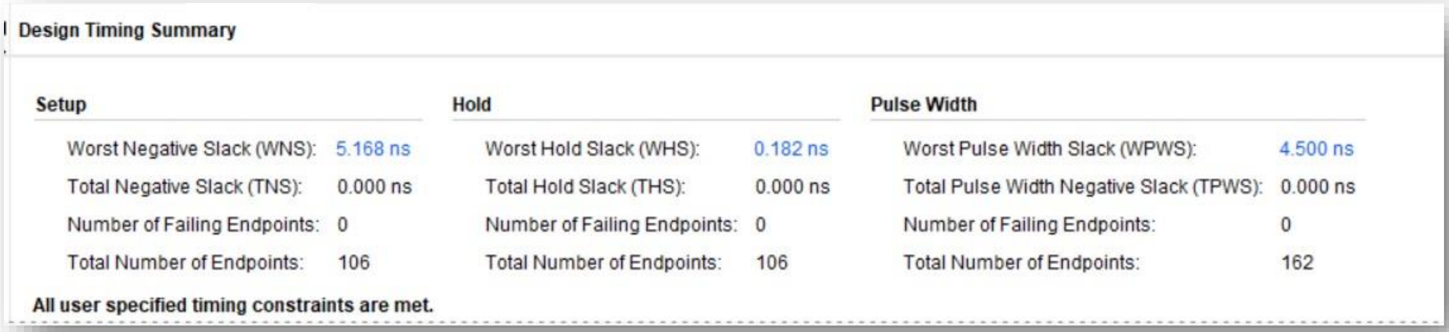
[Synth 8-3331] design Param_Reg_2x1MUX__parameterized1 has unconnected port rst

[Constraints 18-5210] No constraint will be written out.

Schematic:



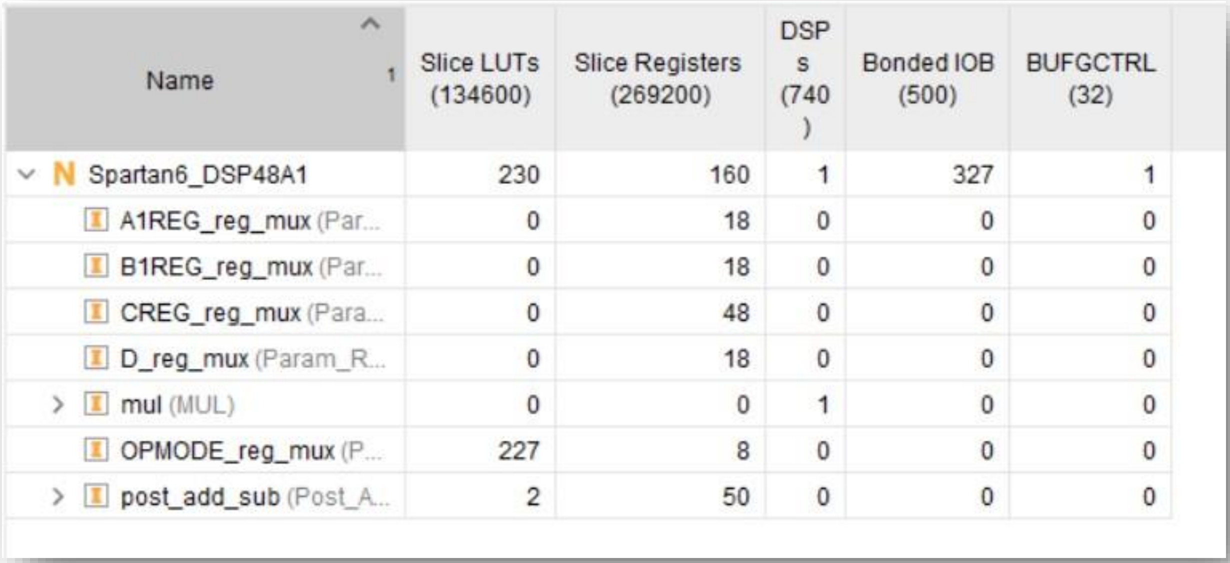
Timing Report:


Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.168 ns	Worst Hold Slack (WHS):	0.182 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:	106	Total Number of Endpoints:	106	Total Number of Endpoints:	162

All user specified timing constraints are met.

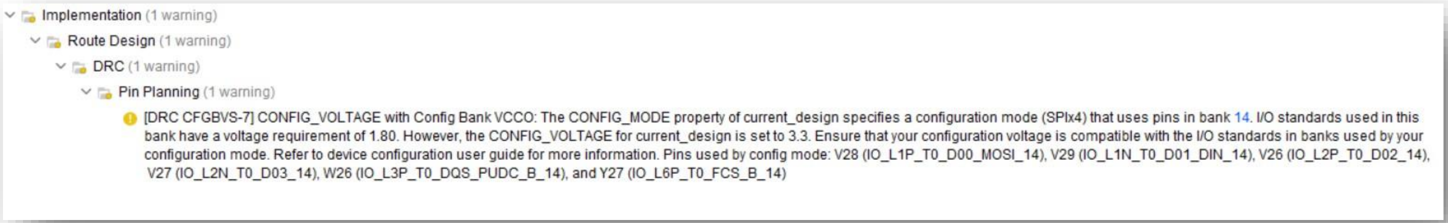
Utilization Report:



Name	Slice LUTs (134600)	Slice Registers (269200)	DSPs (740)	Bonded IOB (500)	BUFGCTRL (32)
Spartan6_DSP48A1	230	160	1	327	1
A1REG_reg_mux (Par...	0	18	0	0	0
B1REG_reg_mux (Par...	0	18	0	0	0
CREG_reg_mux (Para...	0	48	0	0	0
D_reg_mux (Param_R...	0	18	0	0	0
mul (MUL)	0	0	1	0	0
OPMODE_reg_mux (P...	227	8	0	0	0
post_add_sub (Post_A...	2	50	0	0	0

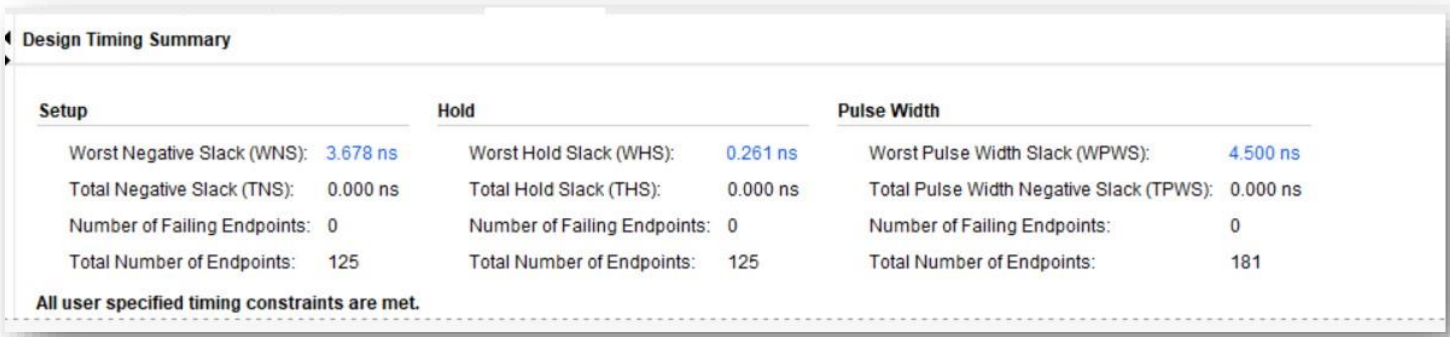
7. Implementation

Message Tab:



```
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)
```

Timing Report:

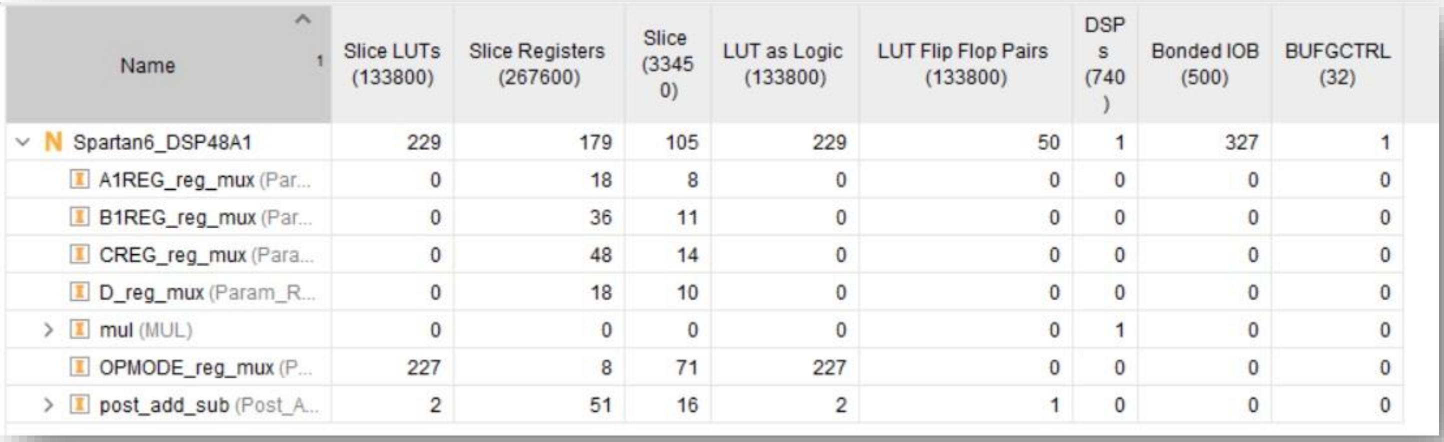


Design Timing Summary

Setup		**Hold**		**Pulse Width**	
Worst Negative Slack (WNS):	3.678 ns	Worst Hold Slack (WHS):	0.261 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

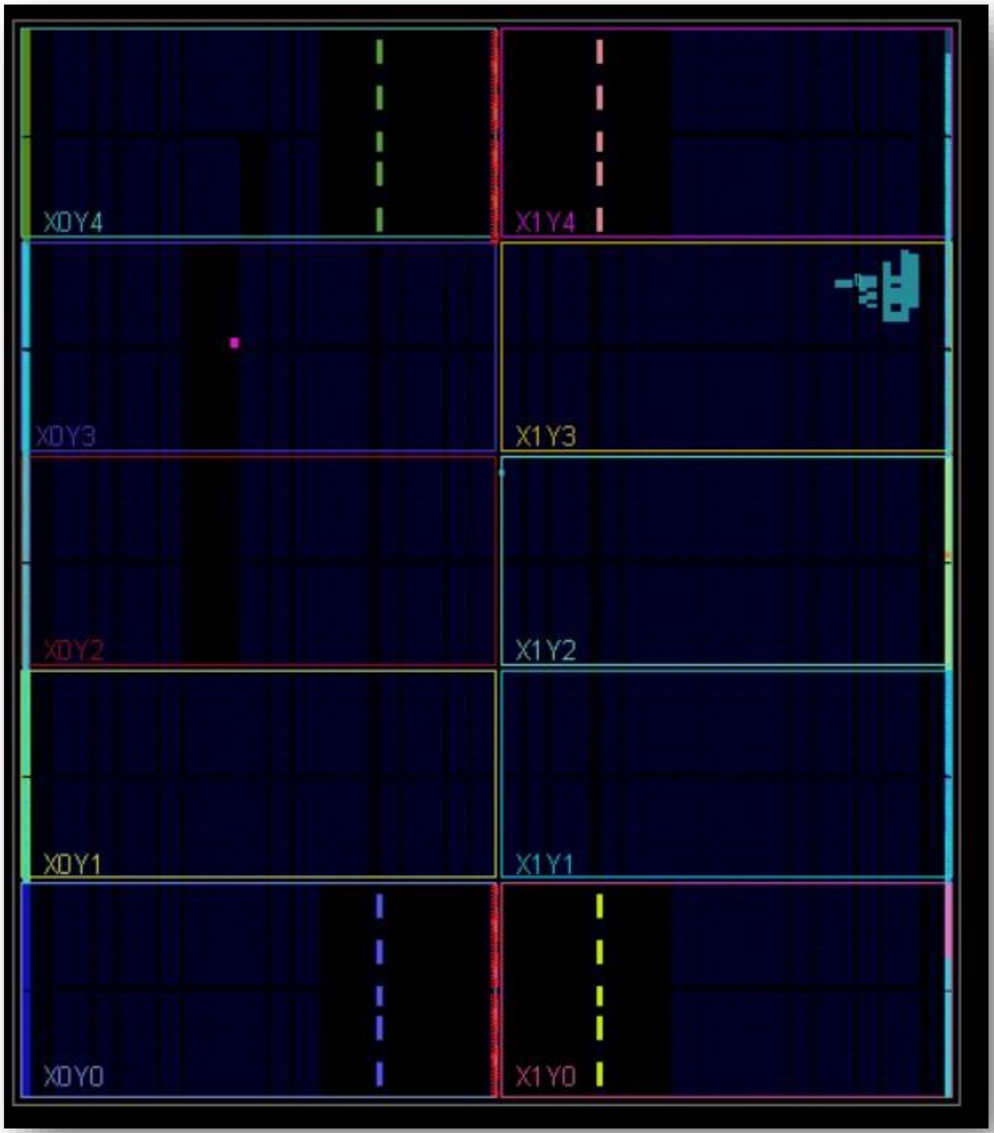
All user specified timing constraints are met.

Utilization Report:

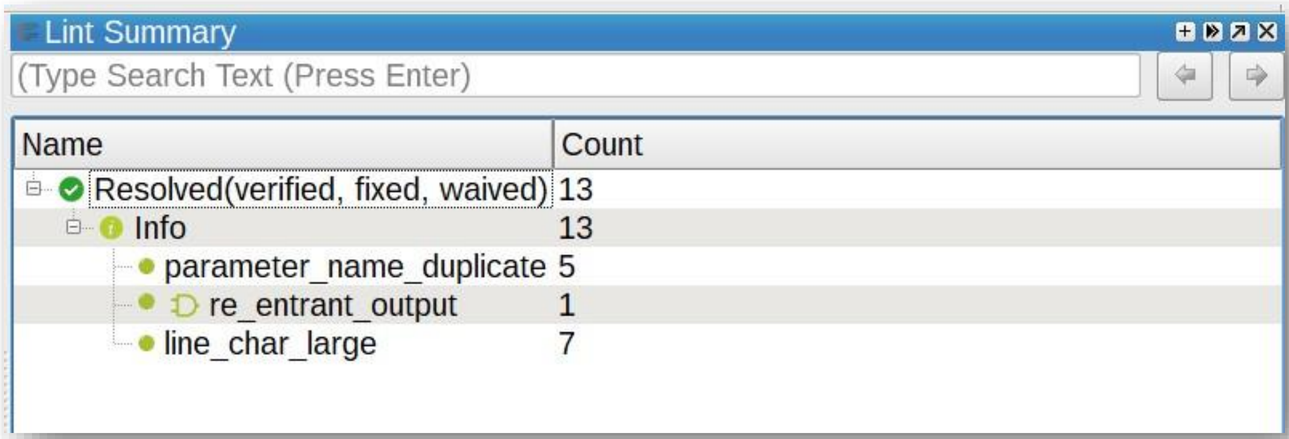


Name	Slice LUTs (133800)	Slice Registers (267600)	Slice (33450)	LUT as Logic (133800)	LUT Flip Flop Pairs (133800)	DSPs (740)	Bonded IOB (500)	BUFGCTRL (32)
Spartan6_DSP48A1	229	179	105	229	50	1	327	1
A1REG_reg_mux (Par...	0	18	8	0	0	0	0	0
B1REG_reg_mux (Par...	0	36	11	0	0	0	0	0
CREG_reg_mux (Para...	0	48	14	0	0	0	0	0
D_reg_mux (Param_R...	0	18	10	0	0	0	0	0
mul (MUL)	0	0	0	0	0	1	0	0
OPMODE_reg_mux (P...	227	8	71	227	0	0	0	0
post_add_sub (Post_A...	2	51	16	2	1	0	0	0

Device Snippet:



8. Linting:



Name	Count
Resolved(verified, fixed, waived)	13
Info	13
parameter_name_duplicate	5
re_entrant_output	1
line_char_large	7