DSP48A1 Project

Verilog implementation and FPGA flow

BY: Youssif Ahmed Sayed

Table of Contents

1. Project Overview

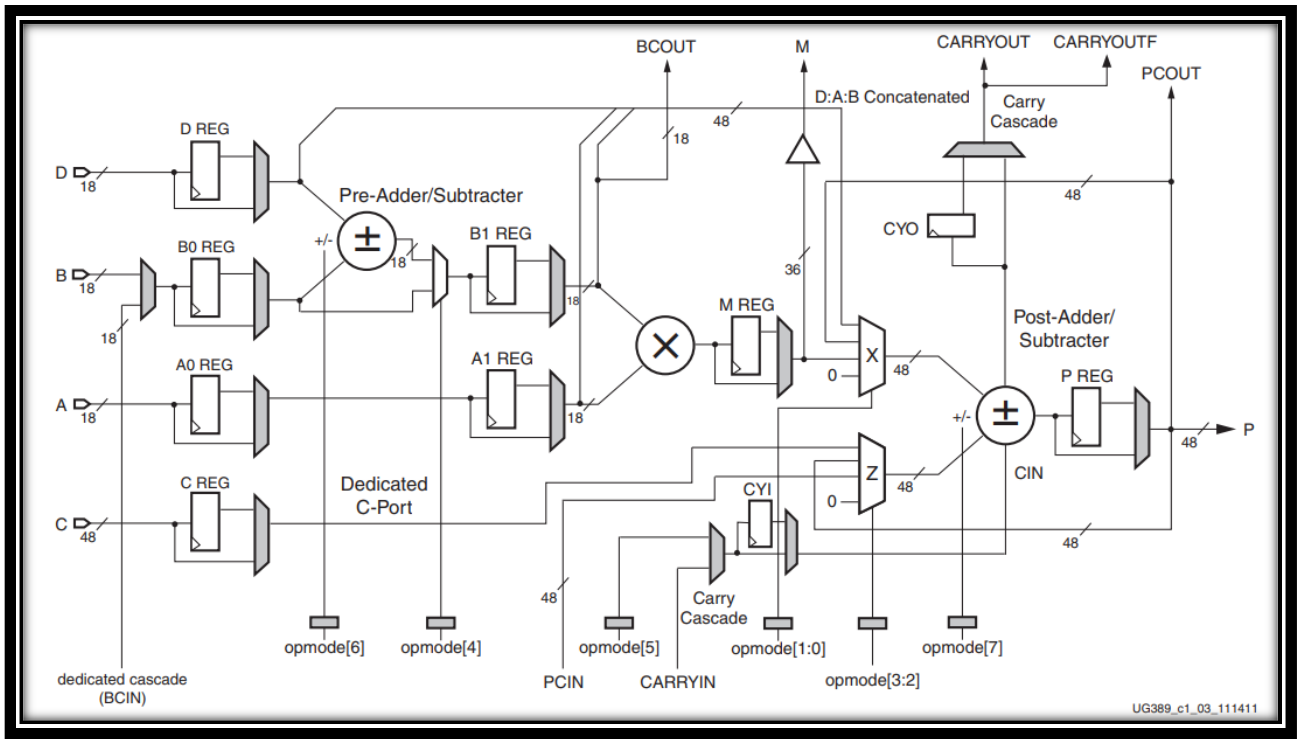
1.1. Description

The project focuses on the design and implementation of a DSP48A1 slice within a Spartan-6 FPGA, ideal for math-intensive applications. The DSP48A1 slice is a specialized hardware block that supports complex digital signal processing (DSP) tasks such as multiplication, addition, and accumulation, making it suitable for high-performance applications requiring efficient arithmetic operations.

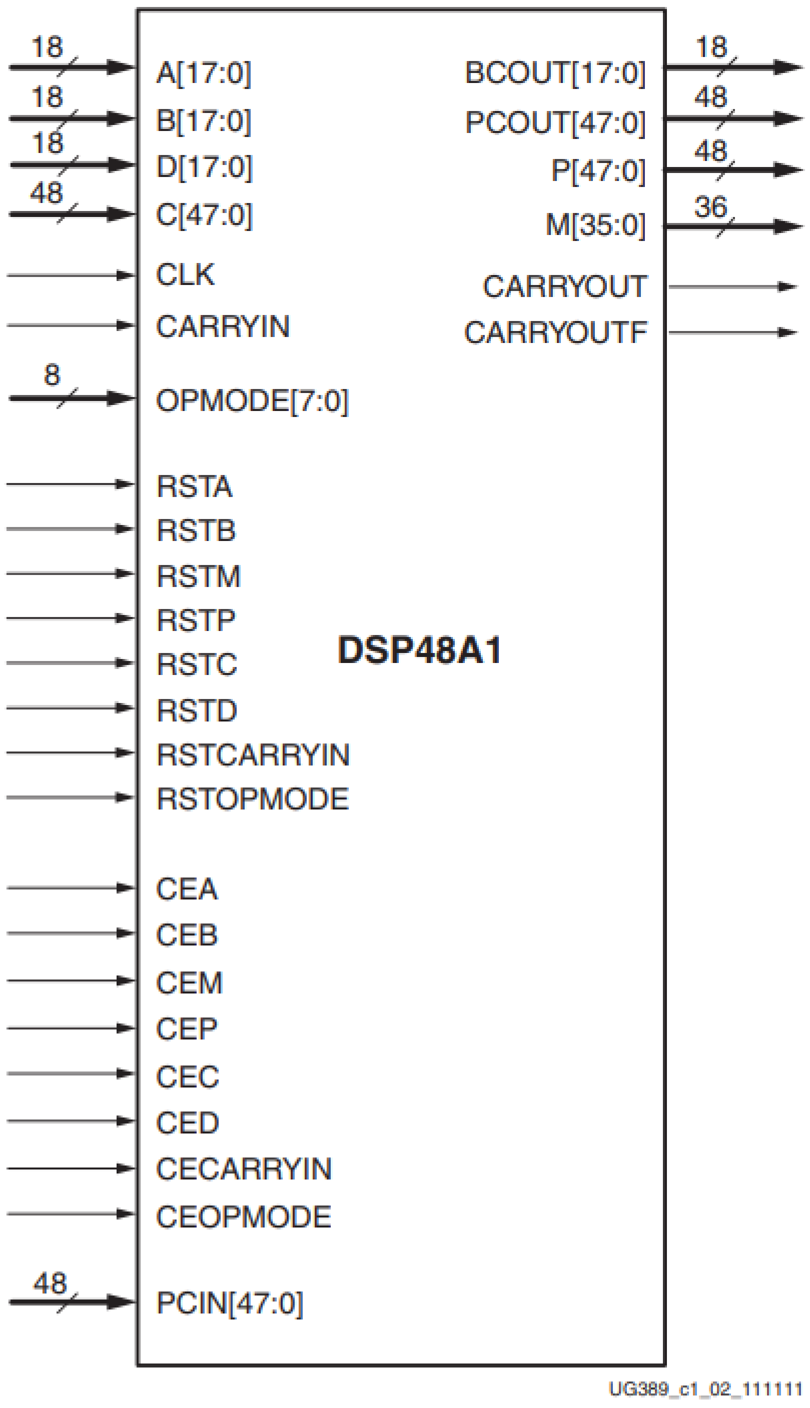
1.2. Key Features

1. **DSP48A1 Slice Design**: Implementation of a DSP48A1 slice utilizing the Spartan-6 FPGA, which includes pipeline stages for enhanced performance in arithmetic operations.
2. **Vivado Design Flow**: The project employs the Vivado design suite to perform elaboration, synthesis, and implementation, ensuring the design is error-free through rigorous checks.
3. **Testbench Development**: Directed test patterns are utilized in the testbench to verify the design against expected values or waveform outputs.
4. **Flexible Configuration**: The design includes configurable parameters such as A0REG, A1REG, B0REG, B1REG, and others, allowing for adjustable pipeline stages to meet specific application needs.
5. **Cascade and Reset Ports**: Supports cascading multiple DSP48A1 slices and includes asynchronous or synchronous reset capabilities.

1.3. DSP48A1 Slice in Detail

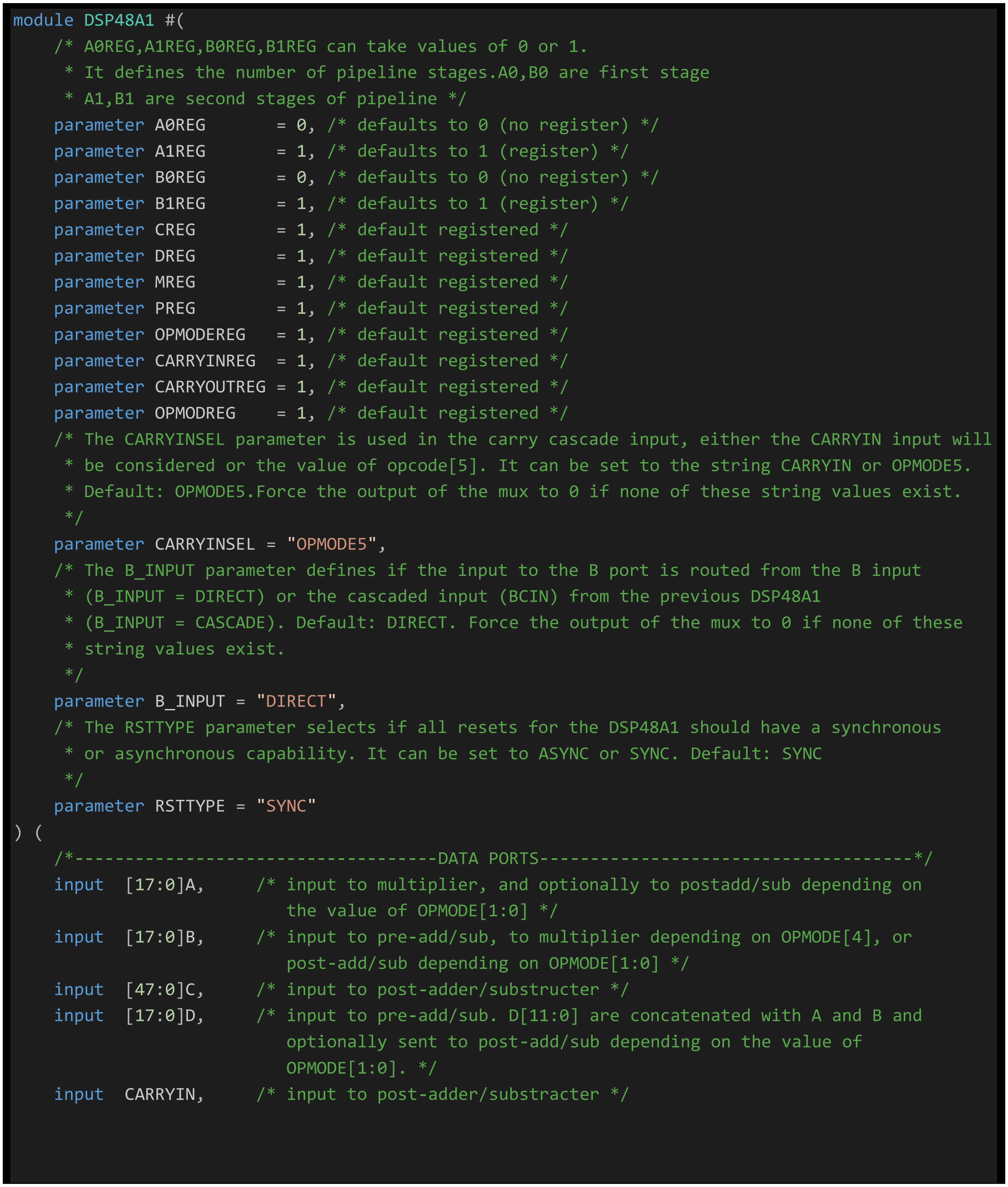


1.4. DSP48A1 Slice Primitive



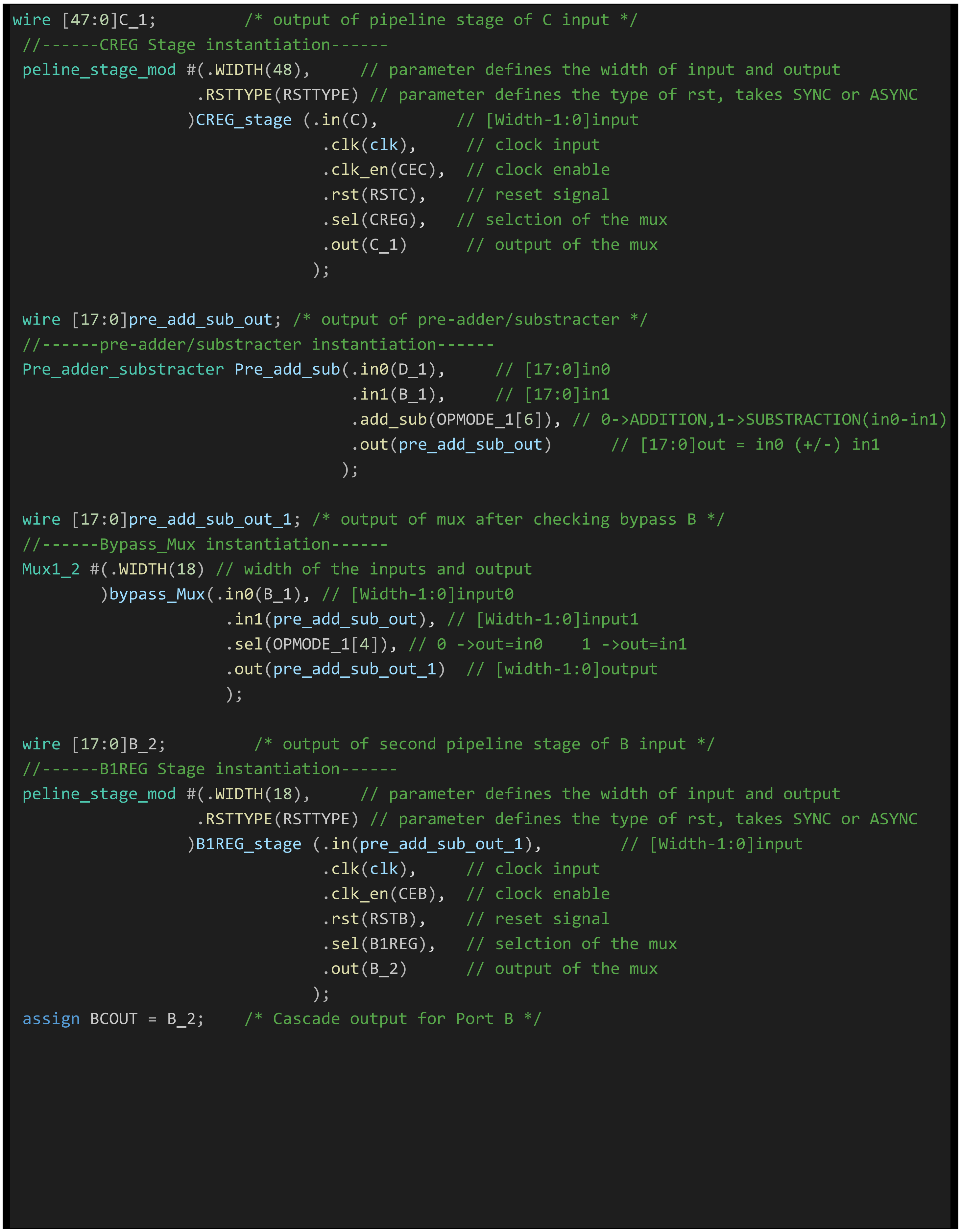
2. RTL Code

2.1. DSP48A1 Module

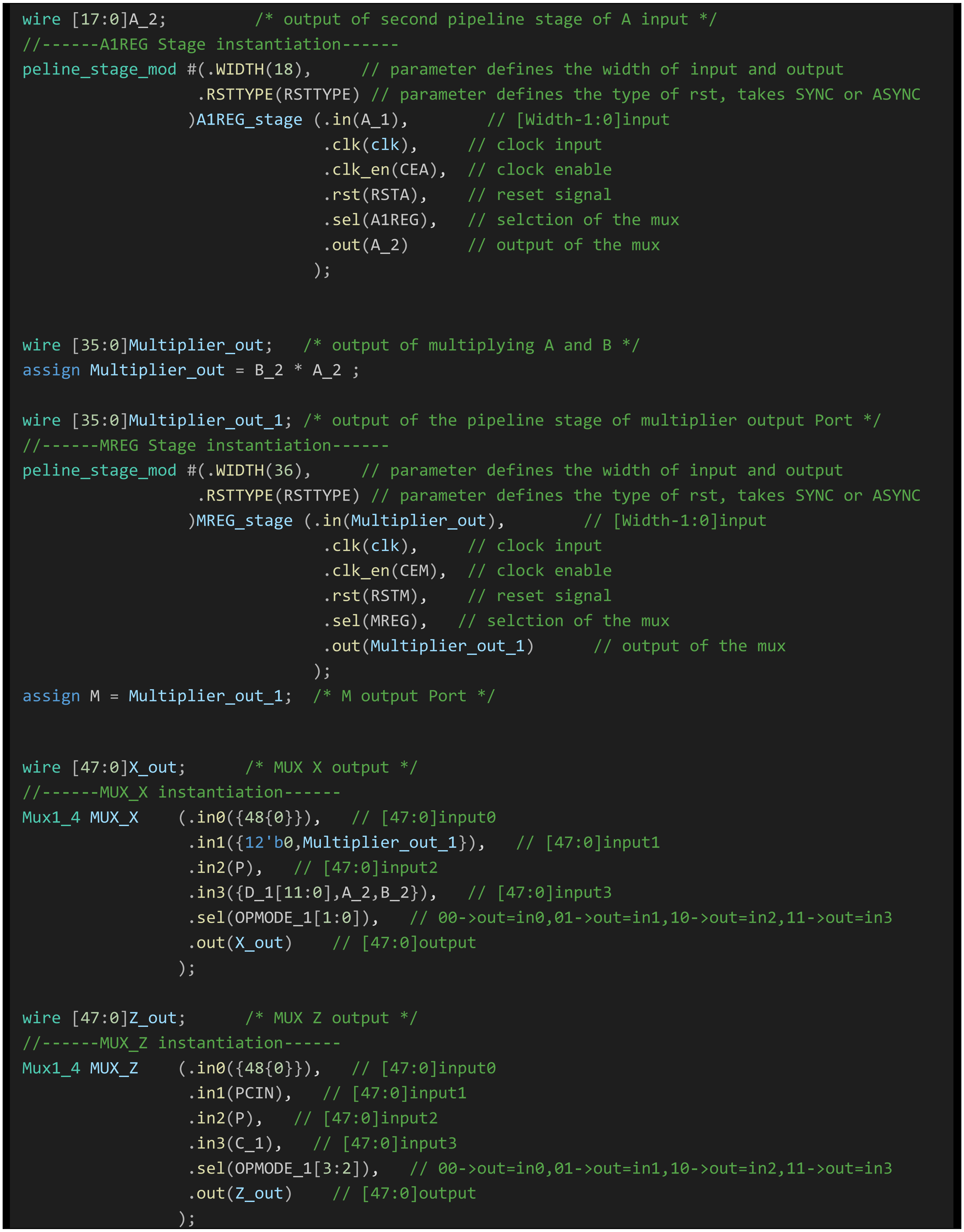


```verilog
module DSP48A1 #(
    /* A0REG,A1REG,B0REG,B1REG can take values of 0 or 1.
     * It defines the number of pipeline stages.A0,B0 are first stage
     * A1,B1 are second stages of pipeline */
    parameter A0REG       = 0, /* defaults to 0 (no register) */
    parameter A1REG       = 1, /* defaults to 1 (register) */
    parameter B0REG       = 0, /* defaults to 0 (no register) */
    parameter B1REG       = 1, /* defaults to 1 (register) */
    parameter CREG        = 1, /* default registered */
    parameter DREG        = 1, /* default registered */
    parameter MREG        = 1, /* default registered */
    parameter PREG        = 1, /* default registered */
    parameter OPMODEREG   = 1, /* default registered */
    parameter CARRYINREG  = 1, /* default registered */
    parameter CARRYOUTREG = 1, /* default registered */
    parameter OPMODREG    = 1, /* default registered */
    /* The CARRYINSEL parameter is used in the carry cascade input, either the CARRYIN input will
     * be considered or the value of opcode[5]. It can be set to the string CARRYIN or OPMODE5.
     * Default: OPMODE5.Force the output of the mux to 0 if none of these string values exist.
     */
    parameter CARRYINSEL = "OPMODE5",
    /* The B_INPUT parameter defines if the input to the B port is routed from the B input
     * (B_INPUT = DIRECT) or the cascaded input (BCIN) from the previous DSP48A1
     * (B_INPUT = CASCADE). Default: DIRECT. Force the output of the mux to 0 if none of these
     * string values exist.
     */
    parameter B_INPUT = "DIRECT",
    /* The RSTTYPE parameter selects if all resets for the DSP48A1 should have a synchronous
     * or asynchronous capability. It can be set to ASYNC or SYNC. Default: SYNC
     */
    parameter RSTTYPE = "SYNC"
) (
    /*-----------------------------------DATA PORTS------------------------------------*/
    input  [17:0]A,     /* input to multiplier, and optionally to postadd/sub depending on
                           the value of OPMODE[1:0] */
    input  [17:0]B,     /* input to pre-add/sub, to multiplier depending on OPMODE[4], or
                           post-add/sub depending on OPMODE[1:0] */
    input  [47:0]C,     /* input to post-adder/substructer */
    input  [17:0]D,     /* input to pre-add/sub. D[11:0] are concatenated with A and B and
                           optionally sent to post-add/sub depending on the value of
                           OPMODE[1:0]. */
    input  CARRYIN,     /* input to post-adder/substracter */
    output [35:0]M,     /* buffered multiplier output,routable to the FPGA logic. it is
                           either the MREG out or the direct out of multiplier */

    output [47:0]P,     /* Primary output from the post-adder/substracter. it is either the
                           PREG out or the direct out of post-adder/substracter */
    output CARRYOUT,    /* Cascade carry out signal from post-adder/subtracter it is either
                           the PREG out or the direct out of post-adder/substracter
                         * This output is to be connected only to CARRYIN of adjacent
                            DSP48A1 if multiple DSP blocks are used
                         */
    output CARRYOUTF,   /* Carry out signal from post-adder/subtracter for use in the FPGA logic.
                           It is a copy of the CARRYOUT signal that can be routed to the user */
    /*---------------------------------CONTROL INPUT PORTS--------------------------------*/
    input  clk,         /* DSP clock */
    input  [7:0]OPMODE, /* control input to select the arithmetic operations of the DSP48A1 */
    /*------------------------------CLOCK ENABLE INPUT PORTS------------------------------*/
    input  CEA,         /* Clock enable for the A port registers: (A0REG & A1REG) */
    input  CEB,         /* Clock enable for the B port registers: (B0REG & B1REG) */
    input  CEC,         /* Clock enable for the C port registers: (CREG) */
    input  CECARRYIN,   /* Clock enable for carry-in register(CYI) and 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) */
    /*----------------------------------RESET INPUT PORTS---------------------------------*/
    input  RSTA,         /* Reset for the A port registers: (A0REG & A1REG) */
    input  RSTB,         /* Reset for the B port registers: (B0REG & B1REG) */
    input  RSTC,         /* Reset for the C port registers: (CREG) */
    input  RSTCARRYIN,   /* Reset for the carry-in register (CYI) and carry-out register (CYO) */
    input  RSTD,         /* Reset for the D port 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 port registers: (PREG=1) */
    /*-----------------------------------Cascade PORTS------------------------------------*/
    output  [17:0]BCOUT,
    /* Cascade output for Port B.The tools translate the BCOUT cascade to the dedicated BCIN
     * input and set the B_INPUT attribute for implementation. If not used, this port should be
     * left unconnected. So, BCOUT is the same dedicated cascade(BCIN)
     */
    input  [47:0]PCIN,     /* Cascade input for Port P */
    output  [47:0]PCOUT    /* Cascade output for Port P */
);
```

```
wire [7:0]OPMODE_1;         /* output of pipeline stage of OPMODE input */
 //------OPMODEREG Stage instantiation------
 peline_stage_mod #(.WIDTH(8),      // parameter defines the width of input and output
                  .RSTTYPE(RSTTYPE) // parameter defines the type of rst, takes SYNC or ASYNC
                 )OPMODEREG_stage (.in(OPMODE),        // [Width-1:0]input
                              .clk(clk),      // clock input
                              .clk_en(CEOPMODE),  // clock enable
                              .rst(RSTOPMODE),    // reset signal
                              .sel(OPMODEREG),    // selction of the mux
                              .out(OPMODE_1)      // output of the mux
                             );
 wire [17:0]D_1;         /* output of pipeline stage of D input */
 //------DREG Stage instantiation------
 peline_stage_mod #(.WIDTH(18),      // parameter defines the width of input and output
                  .RSTTYPE(RSTTYPE) // parameter defines the type of rst, takes SYNC or ASYNC
                 )DREG_stage (.in(D),         // [Width-1:0]input
                              .clk(clk),      // clock input
                              .clk_en(CED),   // clock enable
                              .rst(RSTD),     // reset signal
                              .sel(DREG),     // selction of the mux
                              .out(D_1)       // output of the mux
                             );
 wire [17:0]B_0;   /*output of the mux after checking there is a dedicated cascade BCIN or not */
 // check B_INPUT parameter to detect if there is BCIN or not */
 assign B_0=(B_INPUT=="DIRECT")?  B :
           (B_INPUT=="CASCADE")? BCOUT : 18'b0;
 wire [17:0]B_1;         /* output of first pipeline stage of B input */
 //------B0REG Stage instantiation------
 peline_stage_mod #(.WIDTH(18),      // parameter defines the width of input and output
                  .RSTTYPE(RSTTYPE) // parameter defines the type of rst, takes SYNC or ASYNC
                 )B0REG_stage (.in(B_0),        // [Width-1:0]input
                               .clk(clk),      // clock input
                               .clk_en(CEB),   // clock enable
                               .rst(RSTB),     // reset signal
                               .sel(B0REG),     // selction of the mux
                               .out(B_1)       // output of the mux
                              );

 wire [17:0]A_1;         /* output of first pipeline stage of A input */
 //------A0REG Stage instantiation------
 peline_stage_mod #(.WIDTH(18),      // parameter defines the width of input and output
                  .RSTTYPE(RSTTYPE) // parameter defines the type of rst, takes SYNC or ASYNC
                 )A0REG_stage (.in(A),         // [Width-1:0]input
                               .clk(clk),      // clock input
                               .clk_en(CEA),   // clock enable
                               .rst(RSTA),     // reset signal
                               .sel(A0REG),    // selction of the mux
                               .out(A_1)       // output of the mux
                              );
```



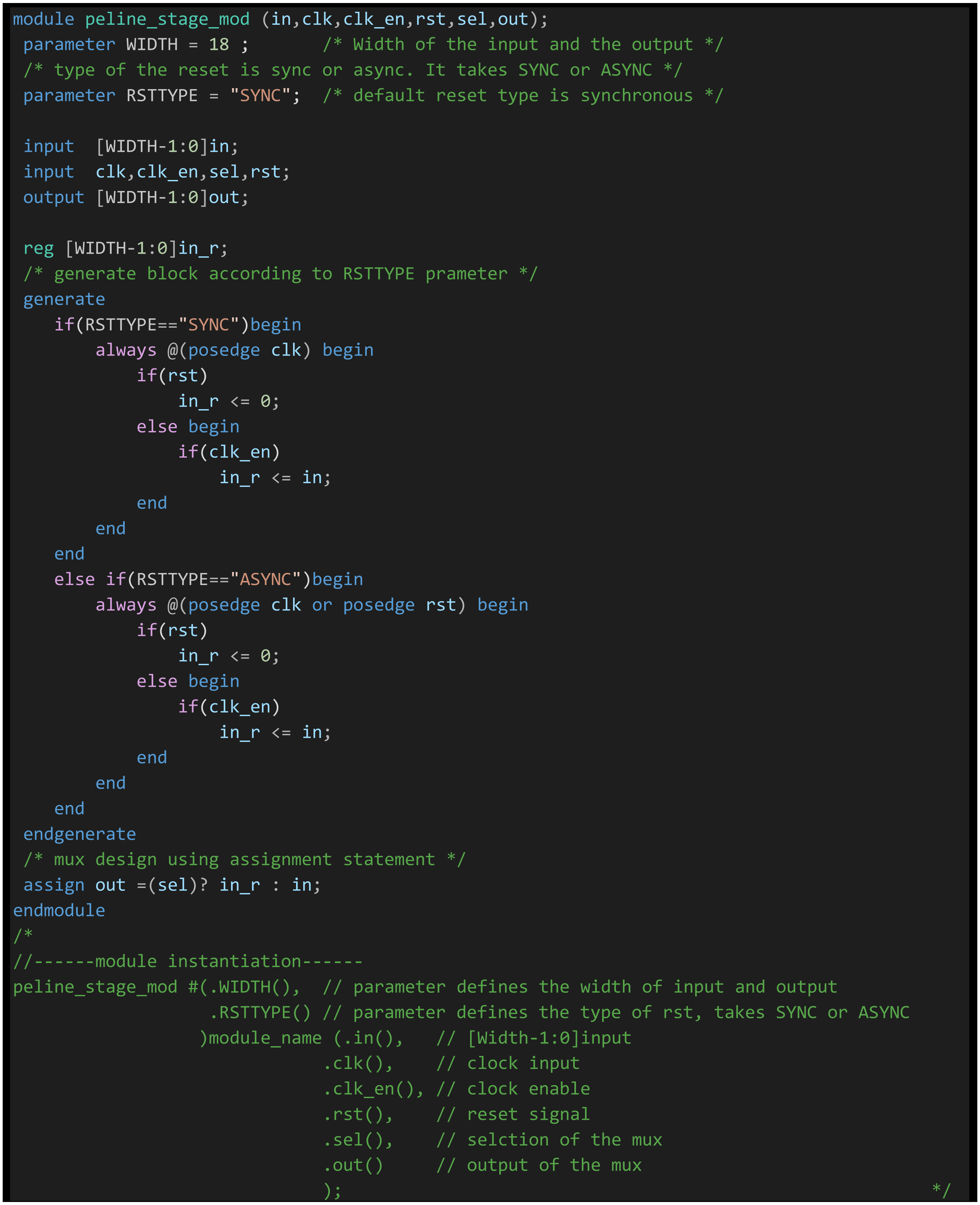
```
wire [47:0]C_1;         /* output of pipeline stage of C input */
 //------CREG Stage instantiation------
 peline_stage_mod #(.WIDTH(48),     // parameter defines the width of input and output
                   .RSTTYPE(RSTTYPE) // parameter defines the type of rst, takes SYNC or ASYNC
                  )CREG_stage (.in(C),        // [Width-1:0]input
                               .clk(clk),     // clock input
                               .clk_en(CEC),  // clock enable
                               .rst(RSTC),    // reset signal
                               .sel(CREG),   // selction of the mux
                               .out(C_1)      // output of the mux
                              );

 wire [17:0]pre_add_sub_out; /* output of pre-adder/substracter */
 //------pre-adder/substracter instantiation------
 Pre_adder_substracter Pre_add_sub(.in0(D_1),     // [17:0]in0
                                  .in1(B_1),     // [17:0]in1
                                  .add_sub(OPMODE_1[6]), // 0->ADDITION,1->SUBSTRACTION(in0-in1)
                                  .out(pre_add_sub_out)      // [17:0]out = in0 (+/-) in1
                                 );

 wire [17:0]pre_add_sub_out_1; /* output of mux after checking bypass B */
 //------Bypass_Mux instantiation------
 Mux1_2 #(.WIDTH(18) // width of the inputs and output
         )bypass_Mux(.in0(B_1), // [Width-1:0]input0
                     .in1(pre_add_sub_out), // [Width-1:0]input1
                     .sel(OPMODE_1[4]), // 0 ->out=in0    1 ->out=in1
                     .out(pre_add_sub_out_1)  // [width-1:0]output
                     );

 wire [17:0]B_2;         /* output of second pipeline stage of B input */
 //------B1REG Stage instantiation------
 peline_stage_mod #(.WIDTH(18),     // parameter defines the width of input and output
                   .RSTTYPE(RSTTYPE) // parameter defines the type of rst, takes SYNC or ASYNC
                  )B1REG_stage (.in(pre_add_sub_out_1),        // [Width-1:0]input
                               .clk(clk),     // clock input
                               .clk_en(CEB),  // clock enable
                               .rst(RSTB),    // reset signal
                               .sel(B1REG),   // selction of the mux
                               .out(B_2)      // output of the mux
                              );
 assign BCOUT = B_2;    /* Cascade output for Port B */
```



```verilog
wire [17:0]A_2;         /* output of second pipeline stage of A input */
//------A1REG Stage instantiation------
peline_stage_mod #(.WIDTH(18),     // parameter defines the width of input and output
                  .RSTTYPE(RSTTYPE) // parameter defines the type of rst, takes SYNC or ASYNC
                 )A1REG_stage (.in(A_1),        // [Width-1:0]input
                               .clk(clk),     // clock input
                               .clk_en(CEA),  // clock enable
                               .rst(RSTA),    // reset signal
                               .sel(A1REG),   // selction of the mux
                               .out(A_2)      // output of the mux
                              );


wire [35:0]Multiplier_out;   /* output of multiplying A and B */
assign Multiplier_out = B_2 * A_2 ;

wire [35:0]Multiplier_out_1; /* output of the pipeline stage of multiplier output Port */
//------MREG Stage instantiation------
peline_stage_mod #(.WIDTH(36),     // parameter defines the width of input and output
                  .RSTTYPE(RSTTYPE) // parameter defines the type of rst, takes SYNC or ASYNC
                 )MREG_stage (.in(Multiplier_out),        // [Width-1:0]input
                               .clk(clk),     // clock input
                               .clk_en(CEM),  // clock enable
                               .rst(RSTM),    // reset signal
                               .sel(MREG),   // selction of the mux
                               .out(Multiplier_out_1)      // output of the mux
                              );
assign M = Multiplier_out_1;  /* M output Port */


wire [47:0]X_out;      /* MUX X output */
//------MUX_X instantiation------
Mux1_4 MUX_X    (.in0({48{0}}),   // [47:0]input0
                 .in1({12'b0,Multiplier_out_1}),   // [47:0]input1
                 .in2(P),   // [47:0]input2
                 .in3({D_1[11:0],A_2,B_2}),   // [47:0]input3
                 .sel(OPMODE_1[1:0]),   // 00->out=in0,01->out=in1,10->out=in2,11->out=in3
                 .out(X_out)    // [47:0]output
                );

wire [47:0]Z_out;      /* MUX Z output */
//------MUX_Z instantiation------
Mux1_4 MUX_Z    (.in0({48{0}}),   // [47:0]input0
                 .in1(PCIN),   // [47:0]input1
                 .in2(P),   // [47:0]input2
                 .in3(C_1),   // [47:0]input3
                 .sel(OPMODE_1[3:2]),   // 00->out=in0,01->out=in1,10->out=in2,11->out=in3
                 .out(Z_out)    // [47:0]output
                );
```

```
 wire CARRYIN_0; /* output of the cin mux after checking Carry in will be OPMODE[5] or
                    CARRYIN*/
 // check CARRYINSEL parameter to detect if there is BCIN or not */
 assign CARRYIN_0 =(CARRYINSEL=="OPMODE5")?  OPMODE_1[5] :
                   (CARRYINSEL=="CARRYIN")? CARRYIN : 0;
 wire CARRYIN_1; /* output of the pipeline stage of CARRYIN input Port */
 //------CARRYINREG Stage instantiation------
 peline_stage_mod #(.WIDTH(1),     // parameter defines the width of input and output
                   .RSTTYPE(RSTTYPE) // parameter defines the type of rst, takes SYNC or ASYNC
                  )CYI_stage (.in(CARRYIN_0),        // [Width-1:0]input
                                .clk(clk),     // clock input
                                .clk_en(CECARRYIN),  // clock enable
                                .rst(RSTCARRYIN),    // reset signal
                                .sel(CARRYINREG),   // selction of the mux
                                .out(CARRYIN_1)      // output of the mux
                               );
 wire [47:0]post_add_sub_out;   /* output of post-adder/substracter */
 wire post_add_sub_cout;    /* carry output of post-adder/substracter */
 //------Post-adder/substracter instantiation------
 Post_adder_substracter Post_add_substract(.in0(Z_out),     // [47:0]in0
                                          .in1(X_out),     // [47:0]in1
                                          .cin(CARRYIN_1),     // carry in input
                                          .add_sub(OPMODE_1[7]), // 0->ADDITION,1-> SUBSTRACTION
                                          .out(post_add_sub_out),//[17:0]out= in0 "+/-" (in1+cin)
                                          .cout(post_add_sub_cout)     // carry out output
                                          );
 //------CARRYOUTREG Stage instantiation------
 peline_stage_mod #(.WIDTH(1),     // parameter defines the width of input and output
                   .RSTTYPE(RSTTYPE) // parameter defines the type of rst, takes SYNC or ASYNC
                  )CYO_stage (.in(post_add_sub_cout),        // [Width-1:0]input
                                .clk(clk),     // clock input
                                .clk_en(CECARRYIN),  // clock enable
                                .rst(RSTCARRYIN),    // reset signal
                                .sel(CARRYOUTREG),   // selction of the mux
                                .out(CARRYOUT)      // output of the mux
                               );
 assign CARRYOUTF = CARRYOUT;/* CARRYOUTF is the same CARRYOUT. CARRYOUTF sent to cascaded DSP */
 //------PREG Stage instantiation------
 peline_stage_mod #(.WIDTH(48),     // parameter defines the width of input and output
                   .RSTTYPE(RSTTYPE) // parameter defines the type of rst, takes SYNC or ASYNC
                  )PREG_stage (.in(post_add_sub_out),        // [Width-1:0]input
                                .clk(clk),     // clock input
                                .clk_en(CEP),  // clock enable
                                .rst(RSTP),    // reset signal
                                .sel(PREG),   // selction of the mux
                                .out(P)      // output of the mux
                               );
 assign PCOUT = P;       /* PCOUT is the same P */
endmodule
```

2.2. Pipeline stage module



```
module peline_stage_mod (in,clk,clk_en,rst,sel,out);
 parameter WIDTH = 18 ;       /* Width of the input and the output */
 /* type of the reset is sync or async. It takes SYNC or ASYNC */
 parameter RSTTYPE = "SYNC";  /* default reset type is synchronous */

 input  [WIDTH-1:0]in;
 input  clk,clk_en,sel,rst;
 output [WIDTH-1:0]out;

 reg [WIDTH-1:0]in_r;
 /* generate block according to RSTTYPE prameter */
 generate
    if(RSTTYPE=="SYNC")begin
        always @(posedge clk) begin
            if(rst)
                in_r <= 0;
            else begin
                if(clk_en)
                    in_r <= in;
            end
        end
    end
    else if(RSTTYPE=="ASYNC")begin
        always @(posedge clk or posedge rst) begin
            if(rst)
                in_r <= 0;
            else begin
                if(clk_en)
                    in_r <= in;
            end
        end
    end
 endgenerate
 /* mux design using assignment statement */
 assign out =(sel)? in_r : in;
endmodule
/*
//------module instantiation------
peline_stage_mod #(.WIDTH(),  // parameter defines the width of input and output
                   .RSTTYPE() // parameter defines the type of rst, takes SYNC or ASYNC
                  )module_name (.in(),   // [Width-1:0]input
                              .clk(),    // clock input
                              .clk_en(), // clock enable
                              .rst(),    // reset signal
                              .sel(),    // selction of the mux
                              .out()     // output of the mux
                              );                                                        */
```

2.3. Pre-adder/subtracter

```verilog
module Pre_adder_substracter (in0,in1,add_sub,out);
 input  [17:0]in0,in1;
 input  add_sub;
 output reg[17:0]out;
 always @(*) begin
    /* if add_sub control is LOW, addition occurs */
    if(~add_sub)
        out = in0 + in1;
    /* if add_sub control is LOW, Substraction occurs */
    else
        out = in0 - in1;
 end
endmodule
/*
//------module instantiation------
Pre_adder_substract Module_name(.in0(),     // [17:0]in0
                                .in1(),     // [17:0]in1
                                .add_sub(), // 0-> ADDITION  1-> SUBSTRACTION(in0-in1)
                                .out()      // [17:0]out = in0 (+/-) in1
                                );
*/
```

2.4. Post-adder/subtracter

```verilog
module Post_adder_substracter (in0,in1,cin,add_sub,out,cout);
 input  [47:0]in0,in1;
 input  add_sub,cin;
 output reg [47:0]out;
 output reg cout;
 always @(*) begin
    /* if add_sub control is LOW, addition occurs */
    if(~add_sub)
        {cout,out} = in0 +(in1+cin);
    /* if add_sub control is LOW, Substraction occurs */
    else
        {cout,out} = in0 -(in1+cin);
 end
endmodule
/*
//------module instantiation------
Post_adder_substracter Module_name(.in0(),     // [47:0]in0
                                   .in1(),     // [47:0]in1
                                   .cin(),     // carry in input
                                   .add_sub(), // 0-> ADDITION  1-> SUBSTRACTION
                                   .out(),     // [17:0]out = in0 "+/-" (in1+cin)
                                   .cout()     // carry out output
                                  );                                                      */
```

2.5. Mux1_4

```verilog
module Mux1_4 (in0,in1,in2,in3,sel,out);

 input  [47:0]in0,in1,in2,in3;
 input  [1:0]sel;
 output reg [47:0]out;
 always @(*) begin
    case (sel)
        2'b00:  out=in0;
        2'b01:  out=in1;
        2'b10:  out=in2;
        2'b11:  out=in3;
    endcase
 end
endmodule
 /*
 //------module instantiation------
 Mux1_4 MUX_name (.in0(),   // [47:0]input0
                  .in1(),   // [47:0]input1
                  .in2(),   // [47:0]input2
                  .in3(),   // [47:0]input3
                  .sel(),   // 00->out=in0,01->out=in1,10->out=in2,11->out=in3
                  .out()    // [47:0]output
                 );
 */
```

2.6. Mux1_2

```verilog
module Mux1_2 (in0,in1,sel,out);
 /* default width of inputs and outputs */
 parameter WIDTH = 18; /* Default width */

 input  [WIDTH-1:0]in0,in1;
 input  sel;
 output [WIDTH-1:0]out;

 assign out=(sel)?in1 : in0;
endmodule
 /*
 //------module instantiation------
 Mux1_2 #(.WIDTH() // width of the inputs and output
         )module_name(.in0(), // [Width-1:0]input0
                      .in1(), // [Width-1:0]input1
                      .sel(), // 0 ->out=in0    1 ->out=in1
                      .out()  // [width-1:0]output
                      );
 */
```

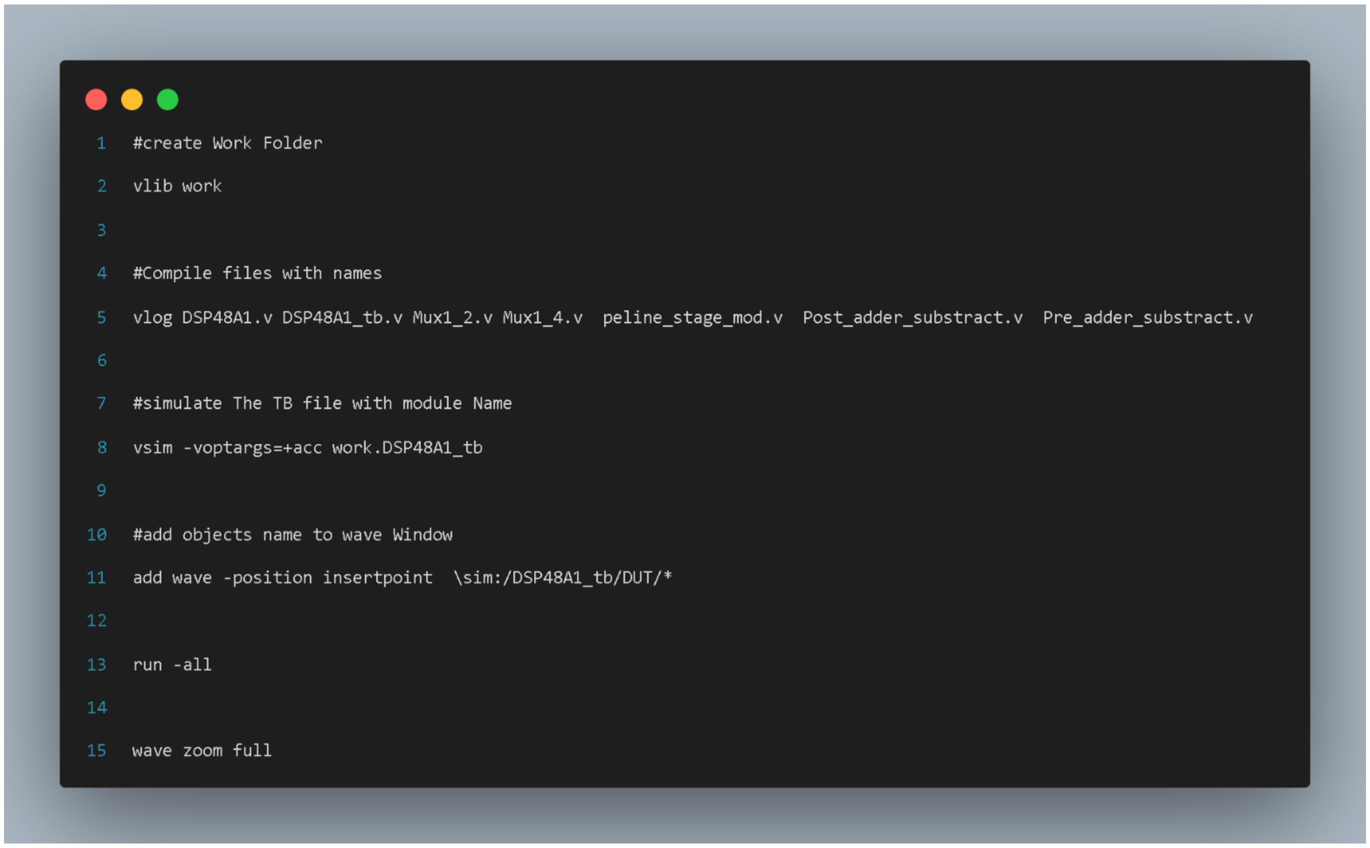
3. Testbench Code

```verilog
module DSP48A1_tb ();
 /* -------Parameters-------*/
  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 OPMODEREG = 1;
  parameter CARRYINREG = 1;
  parameter CARRYOUTREG = 1;
  parameter CARRYINSEL = "OPMODE5";
  parameter B_INPUT = "DIRECT";
  parameter RSTTYPE = "SYNC";
 /*---------inputs---------*/
  reg [17:0] A,B,D;
  reg [47:0] C;
  reg CARRYIN,clk;
  reg [7:0] OPMODE;
  reg CEA,CEB,CEC,CED,CECARRYIN,CEM,CEOPMODE,CEP;
  reg RSTA,RSTB,RSTC,RSTCARRYIN,RSTD,RSTM,RSTOPMODE,RSTP;
  reg [47:0] PCIN;
 /*---------outputs---------*/
  wire [35:0] M;
  wire [47:0] P,PCOUT;
  wire CARRYOUT,CARRYOUTF;
  wire [17:0] BCOUT;
 /*-----DUT INSTATIATIONS-----*/
 DSP48A1 #(
    .A0REG(A0REG),.A1REG(A1REG),.B0REG(B0REG),.B1REG(B1REG),
    .CREG(CREG),.DREG(DREG),.MREG(MREG),.PREG(PREG),
    .OPMODEREG(OPMODEREG),.CARRYINREG(CARRYINREG),
    .CARRYOUTREG(CARRYOUTREG),.CARRYINSEL(CARRYINSEL),
    .B_INPUT(B_INPUT),.RSTTYPE(RSTTYPE)
 ) DUT (.*);

 /*-----clock generation-----*/
 initial begin
    clk=0;
    forever begin
        #5;     clk=~clk;
    end
 end
```

```
/*-----Test Stimulus-----*/
initial begin
$display("----START SIMULATION----");
/*------intialize inputs------*/
/*---------------------------------DATA PORTS---------------------------------*/
 A = 0;    B = 0;      C=0;      D = 0;
 CARRYIN = 0;       OPMODE=0;
/*---------------------------CLOCK ENABLE INPUT PORTS--------------------------*/
 CEA = 0;     CEB = 0;     CEC = 0;
 CED = 0;     CEM = 0;     CEP = 0;
 CECARRYIN = 0;     CEOPMODE = 0;
/*------------------------------RESET INPUT PORTS------------------------------*/
 RSTA = 0;    RSTB = 0;    RSTC = 0;
 RSTD = 0;    RSTM = 0;    RSTP = 0;
 RSTOPMODE = 0;     RSTCARRYIN = 0;
/*-------------------------------Cascade PORTS--------------------------------*/
 PCIN = 0;
/*-----Delay-----*/
 #20;

//set reset signals
 RSTA = 1;    RSTB = 1;    RSTC = 1;
 RSTD = 1;    RSTM = 1;    RSTP = 1;
 RSTOPMODE = 1;     RSTCARRYIN = 1;
 @(negedge clk);
//clear reset signals
 RSTA = 0;    RSTB = 0;    RSTC = 0;
 RSTD = 0;    RSTM = 0;    RSTP = 0;
 RSTOPMODE = 0;     RSTCARRYIN = 0;
//set clock enable signals
 CEA = 1;     CEB = 1;     CEC = 1;
 CED = 1;     CEM = 1;     CEP = 1;
 CECARRYIN = 1;     CEOPMODE = 1;

//Test Multiplication
 A=$random;   B=$random;
 OPMODE=8'b0000_0000;   // M = A * B
 repeat(3) @(negedge clk);

//Test Pre-Adder/Substracter
 A=18'h0001;
 D=$random;   B=$random;
 OPMODE=8'b0001_0000;  // M = 1*(B+D)
 repeat(3) @(negedge clk);

//Test Post-Adder/Substracter
 C=$random;
 OPMODE=8'b0010_1100;  // P = 0 + C + OPMODE[5]
```

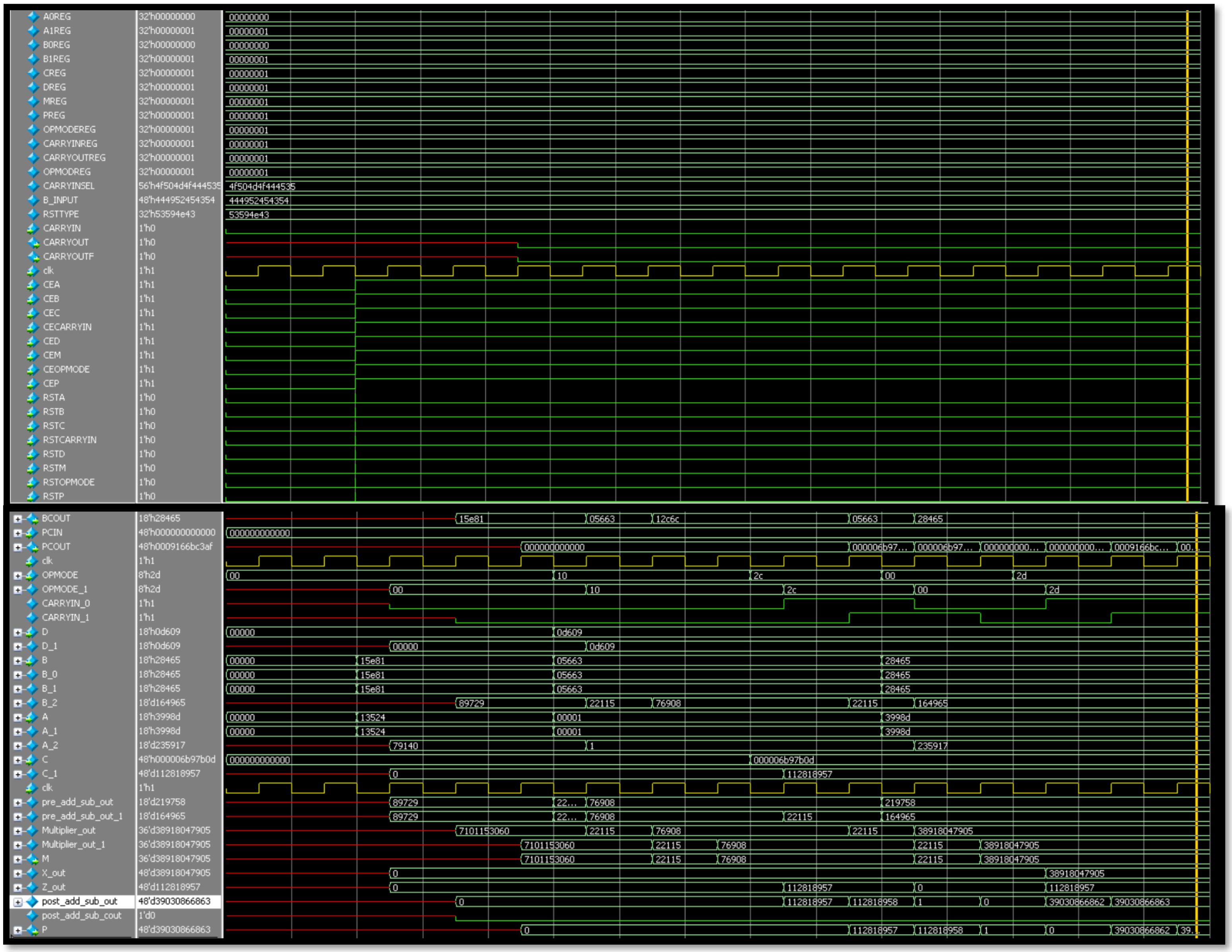
```
  repeat(2) @(negedge clk);

 //Test Post-Adder/Substracter after Multiplication
  A=$random;   B=$random;
  OPMODE=8'b0000_0000;    // M = A * B
  repeat(2) @(negedge clk);
  OPMODE=8'b0010_1101;    // P= M + C + OPMODE[5]
  repeat(3) @(negedge clk);

 $display("----END SIMULATION----");
 $stop;
 end
endmodule
```

4. Do File



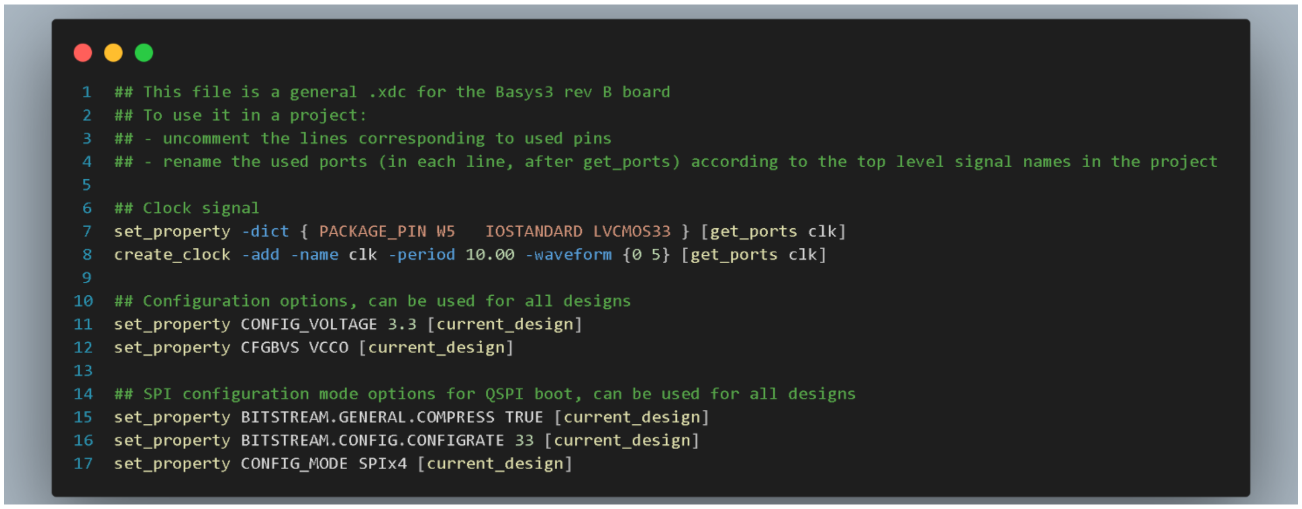
```
#create Work Folder
vlib work

#Compile files with names
vlog DSP48A1.v DSP48A1_tb.v Mux1_2.v Mux1_4.v  peline_stage_mod.v  Post_adder_substract.v  Pre_adder_substract.v

#simulate The TB file with module Name
vsim -voptargs=+acc work.DSP48A1_tb

#add objects name to wave Window
add wave -position insertpoint  \sim:/DSP48A1_tb/DUT/*

run -all

wave zoom full
```

5. QuestaSim Waveform



A0REG	32'h00000000
A1REG	32'h00000001
B0REG	32'h00000000
B1REG	32'h00000001
CREG	32'h00000001
DREG	32'h00000001
MREG	32'h00000001
PREG	32'h00000001
OPMODEREG	32'h00000001
CARRYINREG	32'h00000001
CARRYOUTREG	32'h00000001
OPMODREG	32'h00000001
CARRYINSEL	56'h4f504d4f444535
B_INPUT	48'h444952454354
RSTTYPE	32'h53594e43
CARRYIN	1'h0
CARRYOUT	1'h0
CARRYOUTF	1'h0
clk	1'h1
CEA	1'h1
CEB	1'h1
CEC	1'h1
CECARRYIN	1'h1
CED	1'h1
CEM	1'h1
CEOPMODE	1'h1
CEP	1'h1
RSTA	1'h0
RSTB	1'h0
RSTC	1'h0
RSTCARRYIN	1'h0
RSTD	1'h0
RSTM	1'h0
RSTOPMODE	1'h0
RSTP	1'h0
BCOUT	18'h28465
PCIN	48'h000000000000
PCOUT	48'h0009166bc3af
clk	1'h1
OPMODE	8'h2d
OPMODE_1	8'h2d
CARRYIN_0	1'h1
CARRYIN_1	1'h1
D	18'h0d609
D_1	18'h0d609
B	18'h28465
B_0	18'h28465
B_1	18'h28465
B_2	18'd164965
A	18'h3998d
A_1	18'h3998d
A_2	18'd235917
C	48'h000006b97b0d
C_1	48'd112818957
clk	1'h1
pre_add_sub_out	18'd219758
pre_add_sub_out_1	18'd164965
Multiplier_out	36'd38918047905
Multiplier_out_1	36'd38918047905
M	36'd38918047905
X_out	48'd38918047905
Z_out	48'd112818957
post_add_sub_out	48'd39030866863
post_add_sub_cout	1'd0
P	48'd39030866863

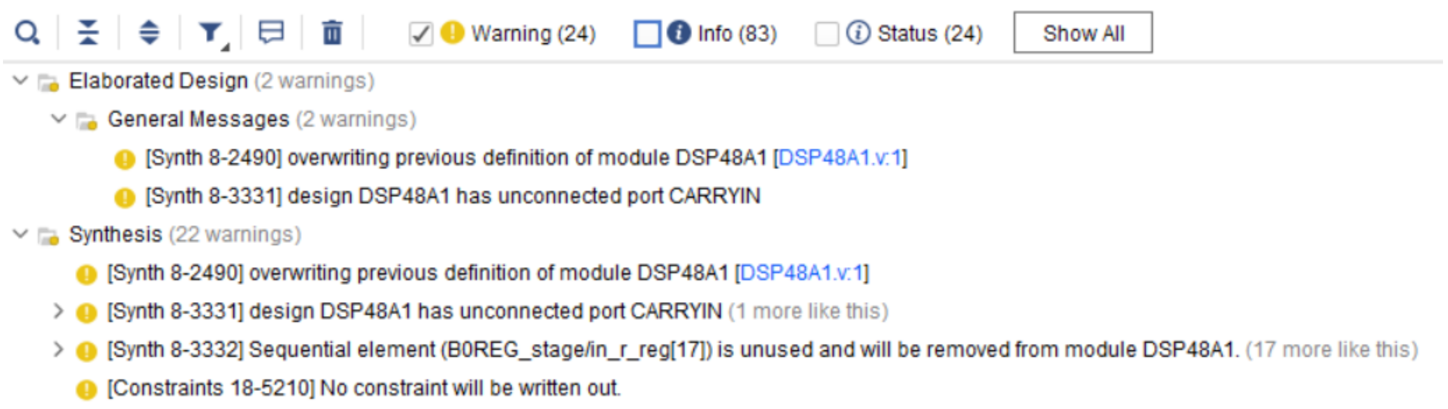
6. Constrains File



```
## 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 clk -period 10.00 -waveform {0 5} [get_ports clk]

## 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]
```

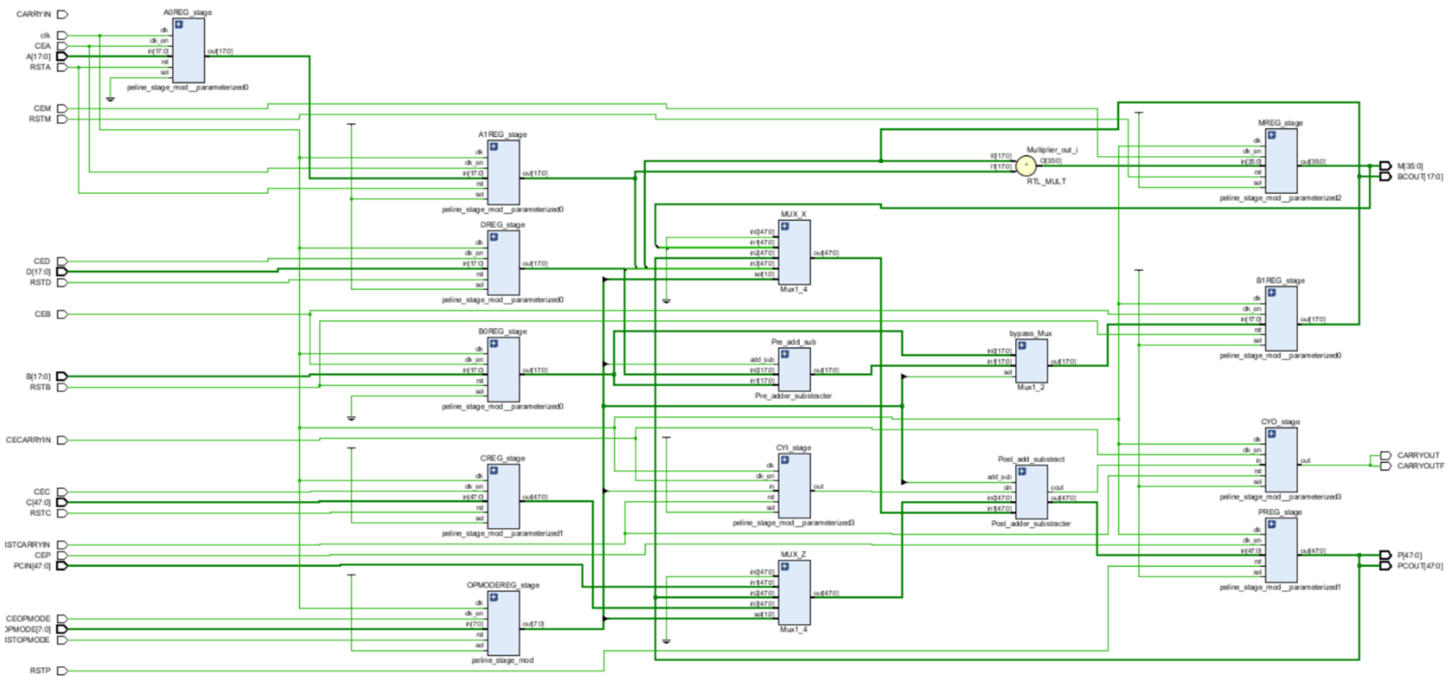
7. Elaboration

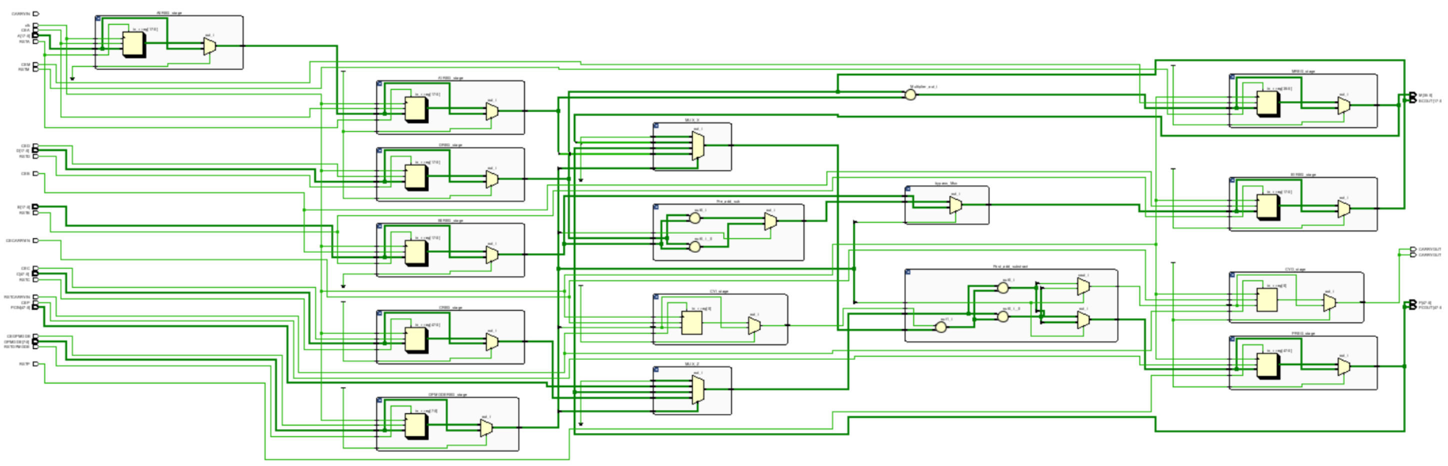
7.1. Messages tab



- Elaborated Design (2 warnings)
  - General Messages (2 warnings)
    - [Synth 8-2490] overwriting previous definition of module DSP48A1 [DSP48A1.v:1]
    - [Synth 8-3331] design DSP48A1 has unconnected port CARRYIN
- Synthesis (22 warnings)
  - [Synth 8-2490] overwriting previous definition of module DSP48A1 [DSP48A1.v:1]
  - [Synth 8-3331] design DSP48A1 has unconnected port CARRYIN (1 more like this)
  - [Synth 8-3332] Sequential element (B0REG_stage/in_r_reg[17]) is unused and will be removed from module DSP48A1. (17 more like this)
  - [Constraints 18-5210] No constraint will be written out.

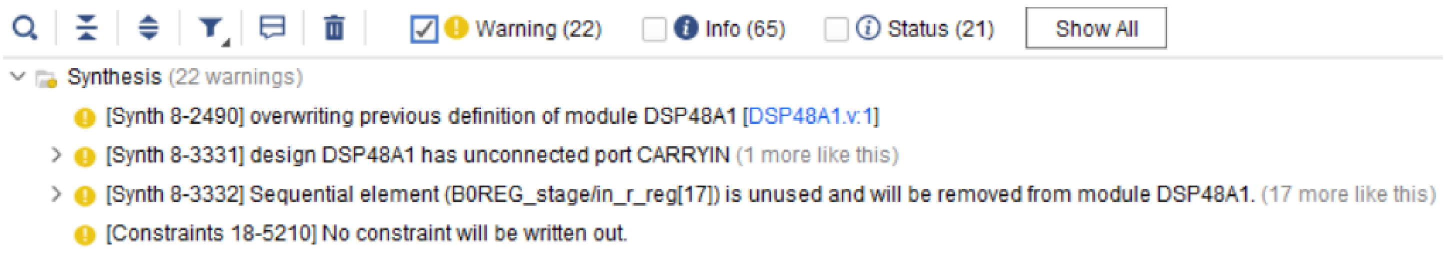
7.2. Schematic





8. Synthesis

8.1. Messages tab



Synthesis (22 warnings)

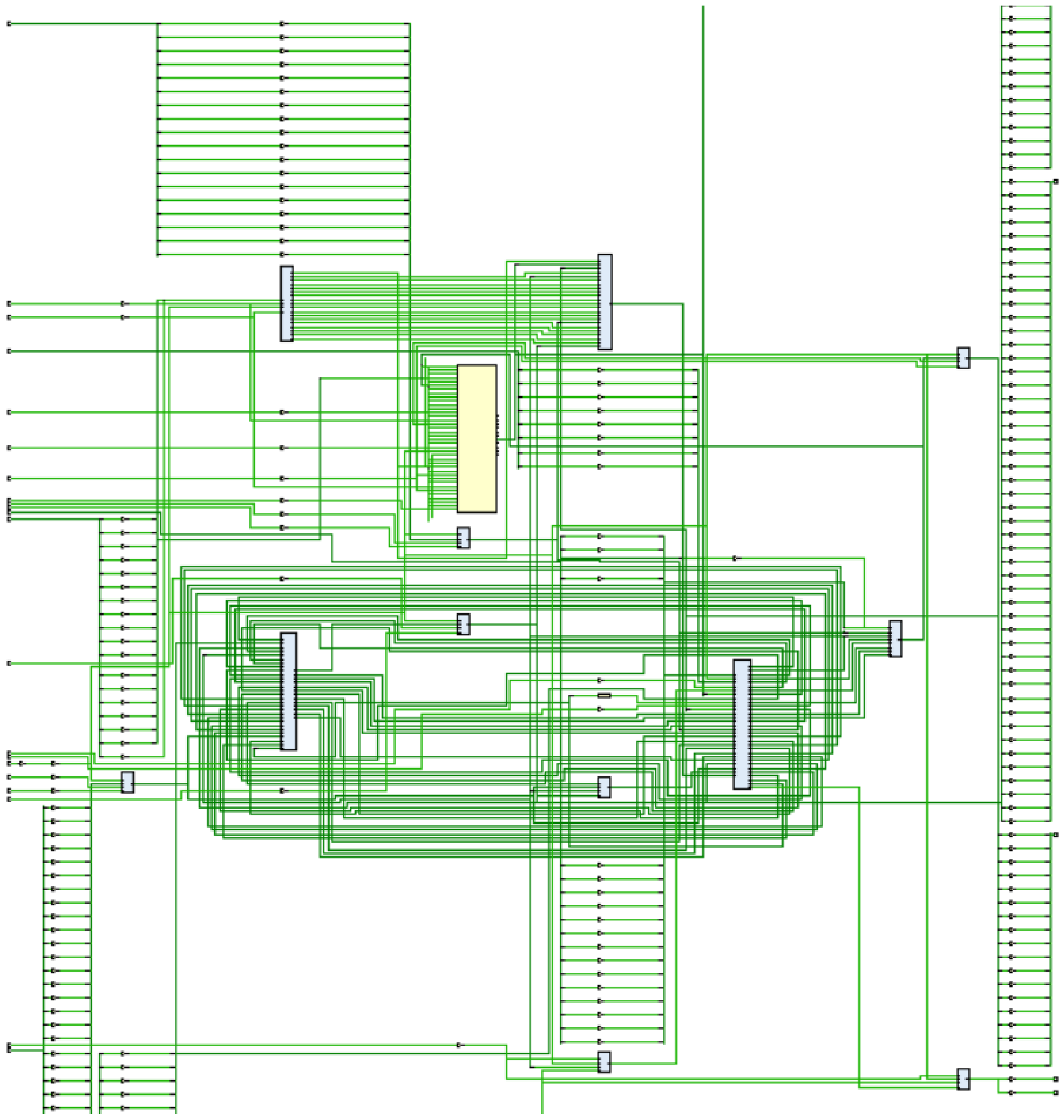
[Synth 8-2490] overwriting previous definition of module DSP48A1 [DSP48A1.v:1]

[Synth 8-3331] design DSP48A1 has unconnected port CARRYIN (1 more like this)

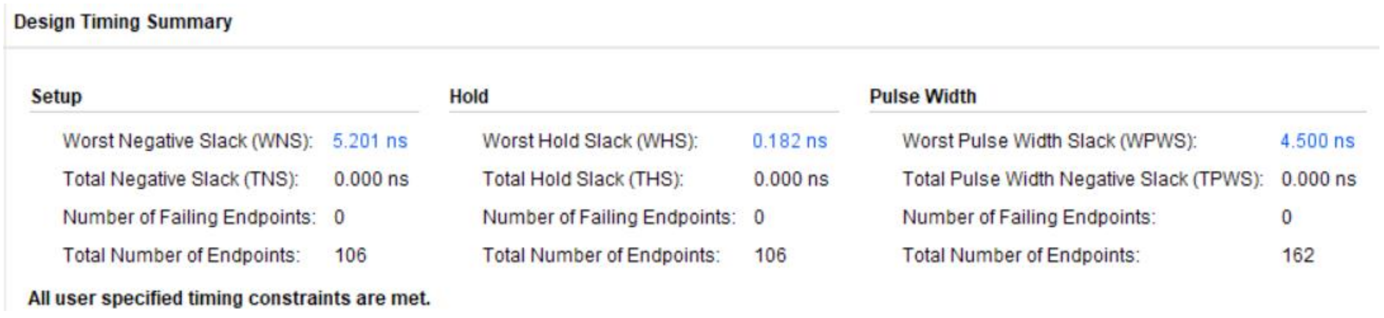
[Synth 8-3332] Sequential element (B0REG_stage/in_r_reg[17]) is unused and will be removed from module DSP48A1. (17 more like this)

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

8.2. Schematic



8.3. Report timing summary

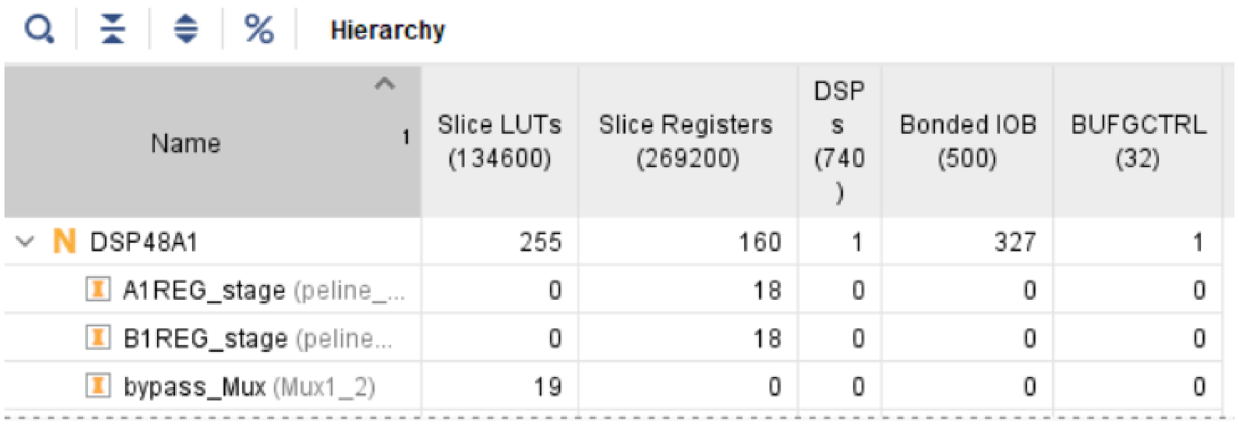


Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.201 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.

8.4. Utilization Report

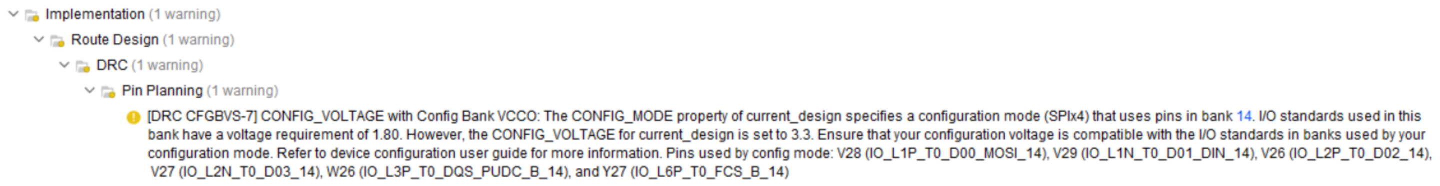


Hierarchy

Name	Slice LUTs (134600)	Slice Registers (269200)	DSPs (740)	Bonded IOB (500)	BUFGCTRL (32)
DSP48A1	255	160	1	327	1
A1REG_stage (peline_...	0	18	0	0	0
B1REG_stage (peline...	0	18	0	0	0
bypass_Mux (Mux1_2)	19	0	0	0	0

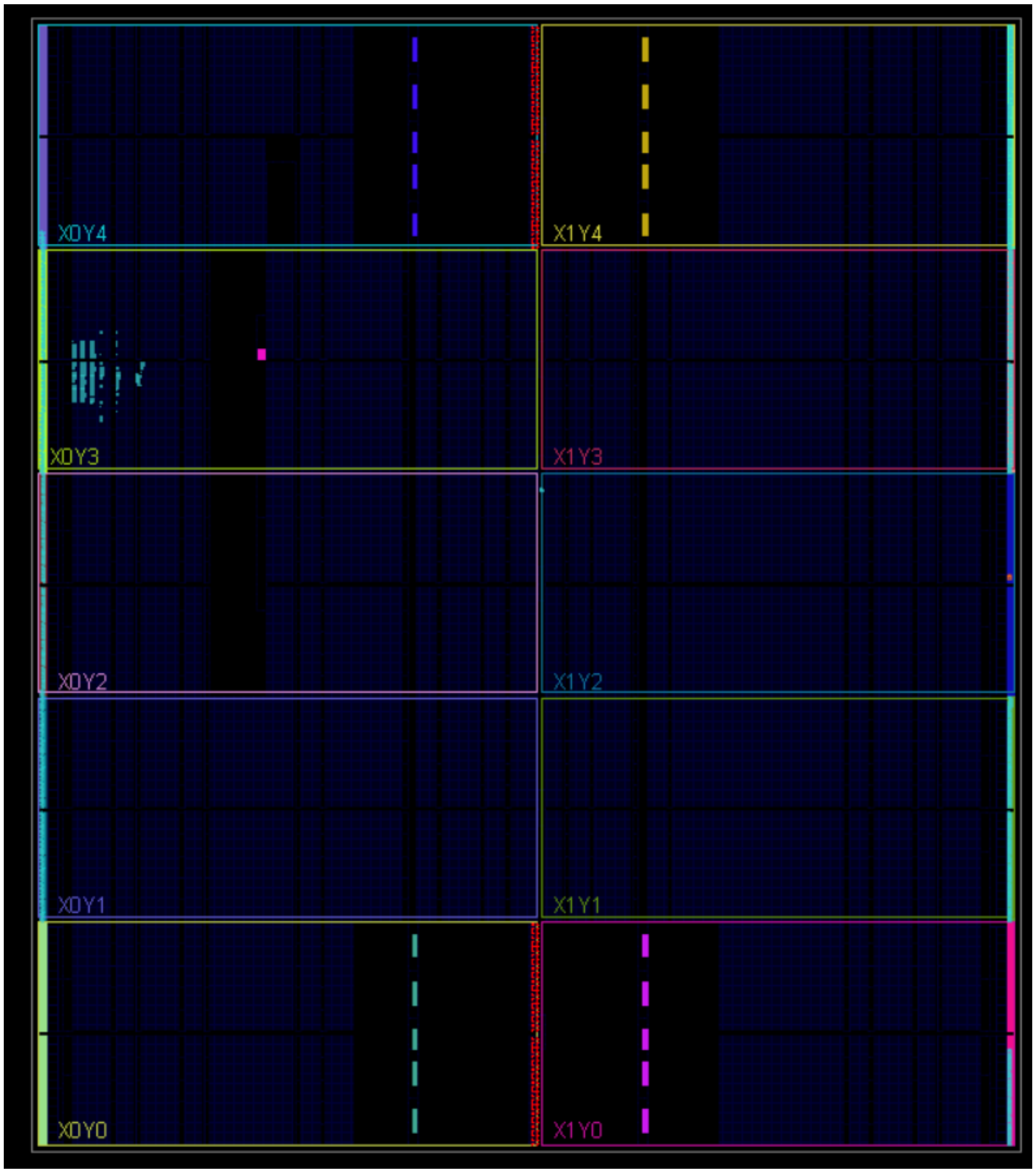
9. Implementation

9.1. Messages tab

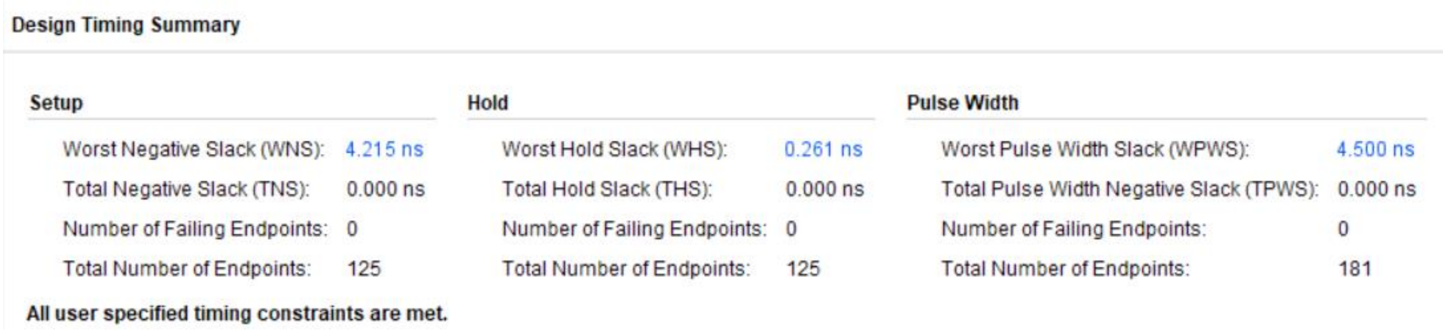


```
Implementation (1 warning)
  Route Design (1 warning)
    DRC (1 warning)
      Pin Planning (1 warning)
        [DRC CFGBVS-7] CONFIG_VOLTAGE with Config Bank VCCO: The CONFIG_MODE property of current_design specifies a configuration mode (SPIx4) that uses pins in bank 14. I/O standards used in 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)
```

9.2. Device



9.3. Design Timing Summary



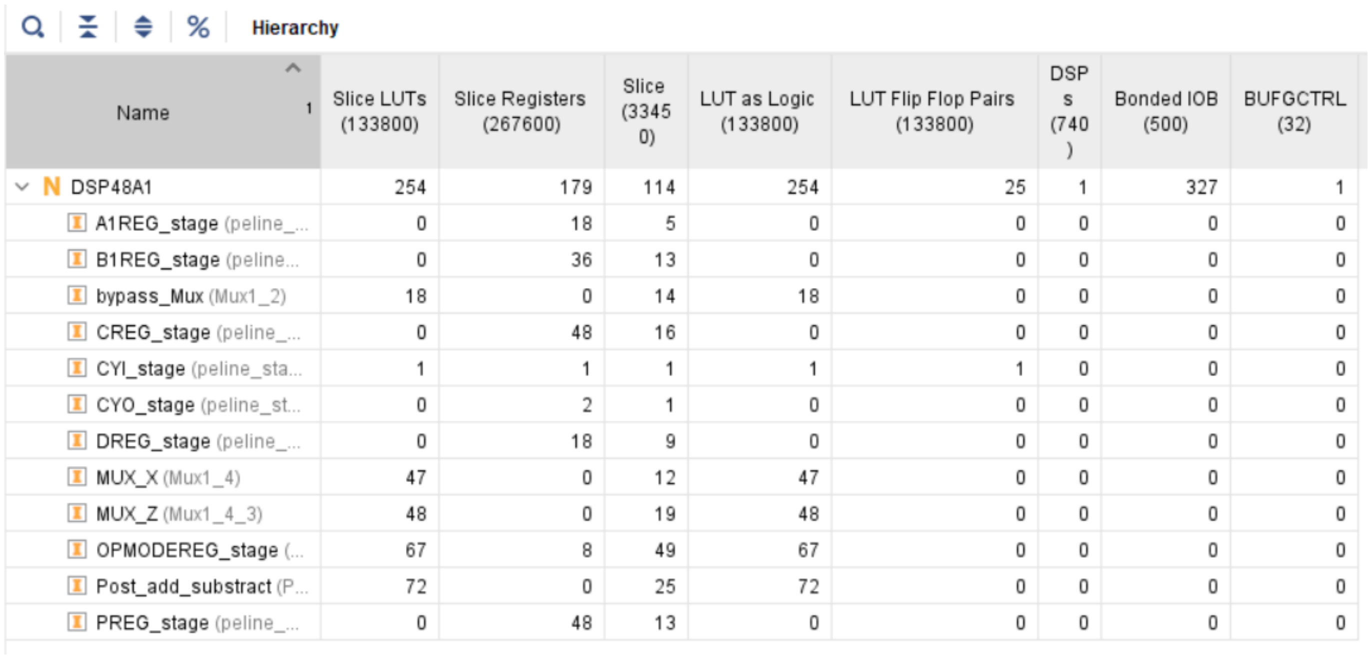
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	4.215 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.

9.4. Utilization Report

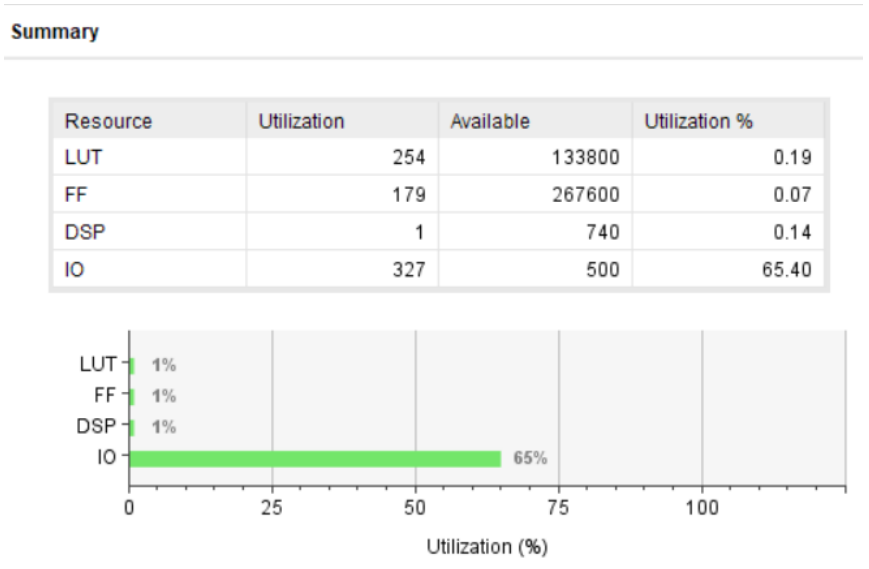
9.4.1. Hierarchy



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	254	179	114	254	25	1	327	1
A1REG_stage (peline_...	0	18	5	0	0	0	0	0
B1REG_stage (peline...	0	36	13	0	0	0	0	0
bypass_Mux (Mux1_2)	18	0	14	18	0	0	0	0
CREG_stage (peline_...	0	48	16	0	0	0	0	0
CYI_stage (peline_sta...	1	1	1	1	1	0	0	0
CYO_stage (peline_st...	0	2	1	0	0	0	0	0
DREG_stage (peline_...	0	18	9	0	0	0	0	0
MUX_X (Mux1_4)	47	0	12	47	0	0	0	0
MUX_Z (Mux1_4_3)	48	0	19	48	0	0	0	0
OPMODEREG_stage (...	67	8	49	67	0	0	0	0
Post_add_substract (P...	72	0	25	72	0	0	0	0
PREG_stage (peline_...	0	48	13	0	0	0	0	0

9.4.2. Summary



Summary

Resource	Utilization	Available	Utilization %
LUT	254	133800	0.19
FF	179	267600	0.07
DSP	1	740	0.14
IO	327	500	65.40