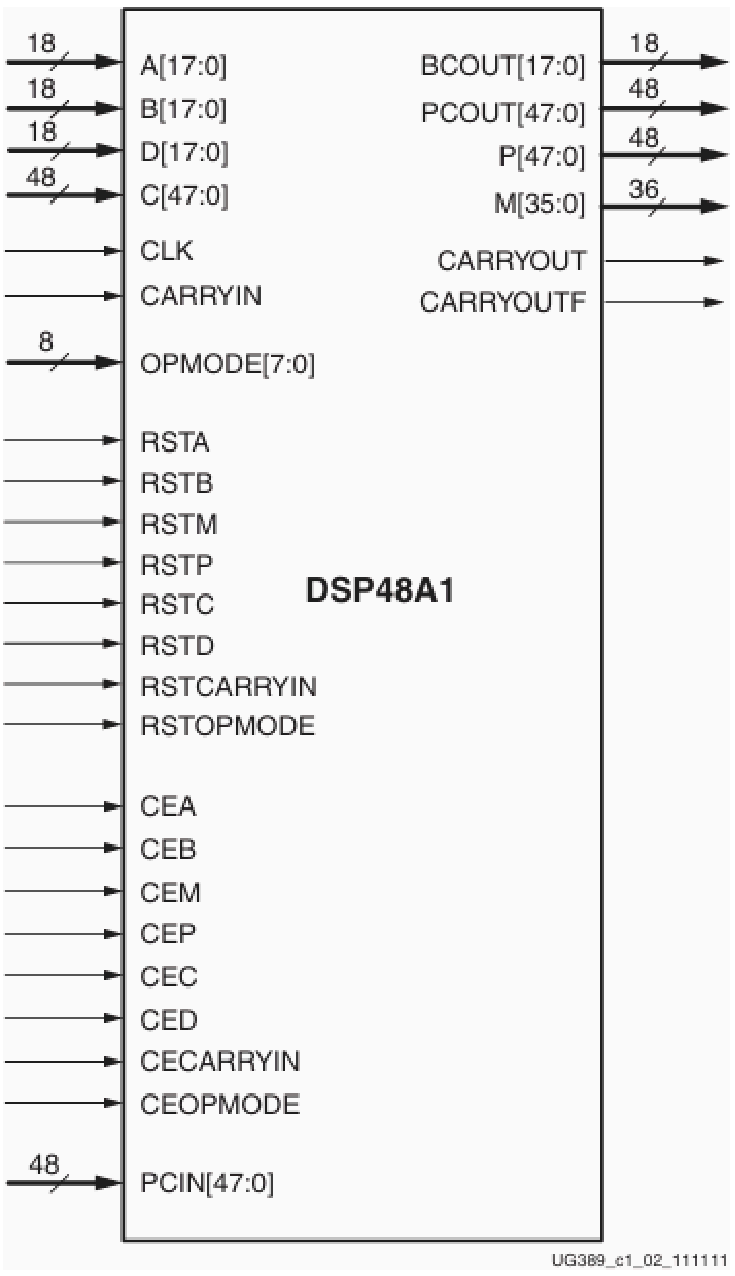
DIGITAL DESIGN PROJECT

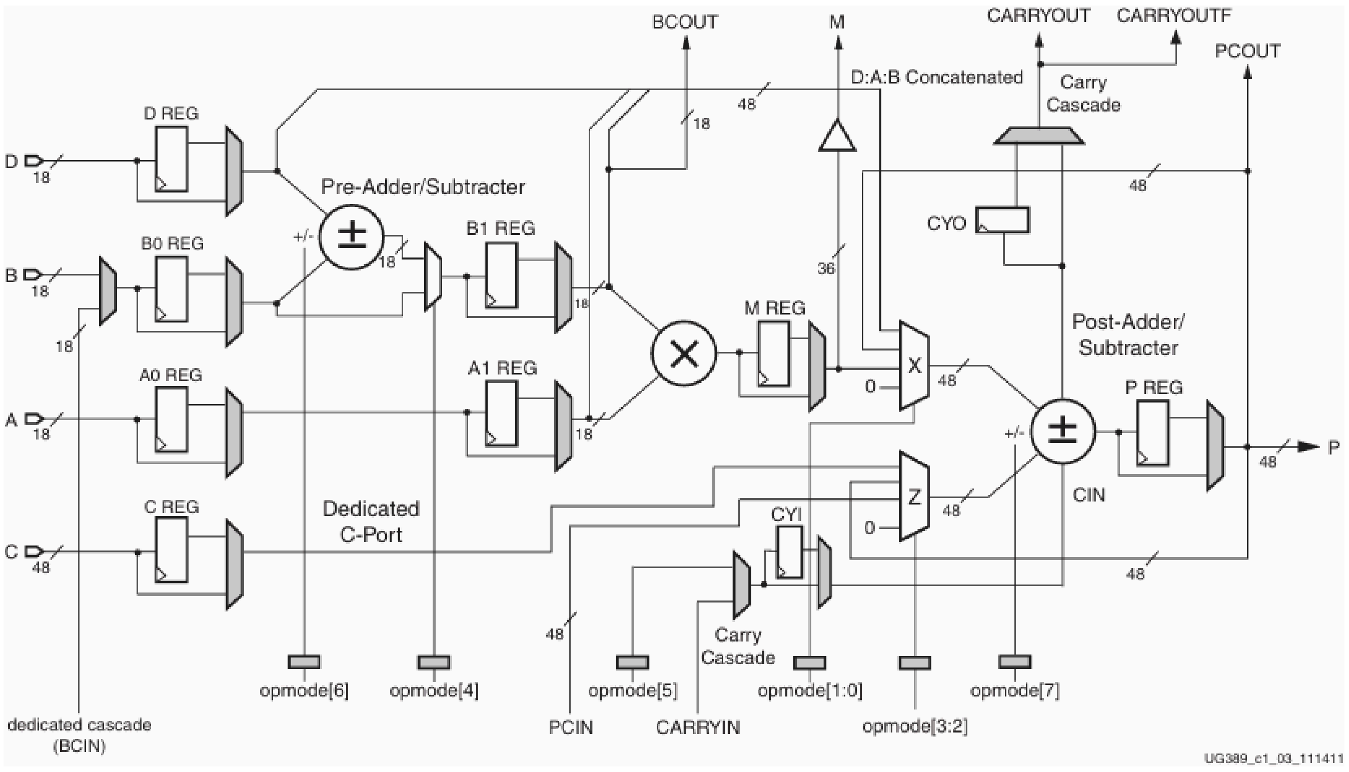
AUG 2025

Spartan6 - DSP48A1

Prepared by : Noureddine Mohamed



A[17:0]
B[17:0]
D[17:0]
C[47:0]
CLK
CARRYIN
OPMODE[7:0]
RSTA
RSTB
RSTM
RSTP
RSTC
RSTD
RSTCARRYIN
RSTOPMODE
CEA
CEB
CEM
CEP
CEC
CED
CECARRYIN
CEOPMODE
PCIN[47:0]
BCOUT[17:0]
PCOUT[47:0]
P[47:0]
M[35:0]
CARRYOUT
CARRYOUTF
DSP48A1
18
18
18
48
8
48
18
48
48
36
UG389_c1_02_111111



BCOUT
M
CARRYOUT
CARRYOUTF
PCOUT
D:A:B Concatenated
Carry Cascade
D REG
D
B0 REG
B
A0 REG
A
C REG
C
Pre-Adder/Subtracter
+/-
B1 REG
A1 REG
M REG
CYO
Post-Adder/ Subtracter
P REG
P
X
Z
CIN
CYI
Dedicated C-Port
Carry Cascade
opmode[6]
opmode[4]
opmode[5]
opmode[1:0]
opmode[3:2]
opmode[7]
dedicated cascade (BCIN)
PCIN
CARRYIN
18
48
36
0
UG389_c1_03_111411

RTL CODE

DSP48A1

```
module DSP48A1 #(
  parameter A0REG = 0,
  parameter A1REG = 1,
  parameter B0REG = 0,
  parameter B1REG = 1,
  parameter CREG = 1,
  parameter DREG = 1,
  parameter MREG = 1,
  parameter PREG = 1,
  parameter CARRYINREG = 1,
  parameter CARRYOUTREG = 1,
  parameter OPMODEREG = 1,
  parameter CARRYINSEL = "OPMODE5",
  parameter B_INPUT = "DIRECT" ,
  parameter RSTTYPE = "SYNC"
)(
  input [17:0] A,
  input [17:0] B,
  input [47:0] C,
  input [17:0] D,
  input    CARRYIN,
  output [35:0] M,
  output [47:0] P,
  output    CARRYOUT,
  output    CARRYOUTF,
  input    CLK,
  input [7:0] OPMODE,
  input    CEA,
  input    CEB,
  input    CEC,
  input    CECARRYIN,
  input    CED,
  input    CEM,
  input    CEOPMODE,
  input    CEP,
  input    RSTA, input    RSTB,
  input    RSTC,
  input    RSTCARRYIN,
  input    RSTD,
  input    RSTM,
  input    RSTOPMODE,
  input    RSTP,
  input [17:0] BCIN,
  input [47:0] PCIN,
  output [17:0] BCOUT,
  output [47:0] PCOUT
);
```

RTL CODE

INTERNAL WIRES

```
  wire [17:0] A0_out, A1_out, A0_reg_out, A1_reg_out;
  wire [17:0] B0_out, B1_out, B_mux_out, B0_reg_out, B1_reg_out;
  wire [47:0] C_out, C_reg_out;
  wire [17:0] D_out, D_reg_out;
  wire [7:0] OPMODE_out, OPMODE_reg_out;
  wire CARRYIN_out, CARRYIN_reg_out, CARRYIN_mux_out;
  wire CARRYOUT_out, CARRYOUT_reg_out;
  wire [35:0] M_out, M_reg_out, Multiplier_out;
  wire [17:0] pre_adder_out, OPMODE4_MUX_out;
  wire [47:0] post_adder_out, Z_mux_out, X_mux_out, P_reg_out,
Concatenate_wires;
```

A PATH

```
 REG #(.WIDTH(18), .RSTTYPE(RSTTYPE)) A0_reg (
    .clk(CLK), .CE(CEA), .rst(RSTA), .data_in(A), .data_out(A0_reg_out)
  );
  MUX #(.WIDTH(18), .NUM_INPUTS(2)) A0_mux (
    .IN0(A), .IN1(A0_reg_out), .IN2(18'b0), .IN3(18'b0), .sel(A0REG), .data_out(A0_out)
  );
  REG #(.WIDTH(18), .RSTTYPE(RSTTYPE)) A1_reg (
    .clk(CLK), .CE(CEA), .rst(RSTA), .data_in(A0_out), .data_out(A1_reg_out)
  );
  MUX #(.WIDTH(18), .NUM_INPUTS(2)) A1_mux (
    .IN0(A0_out), .IN1(A1_reg_out), .IN2(18'b0), .IN3(18'b0), .sel(A1REG), .data_out(A1_out)
  );
```

B PATH

```
MUX #(.WIDTH(18), .NUM_INPUTS(2)) B_mux (
  .IN0(B), .IN1(BCIN), .IN2(18'b0), .IN3(18'b0), .sel(B_INPUT == "CASCADE"), .data_out(B_mux_out)
);
REG #(.WIDTH(18), .RSTTYPE(RSTTYPE)) B0_reg (
  .clk(CLK), .CE(CEB), .rst(RSTB), .data_in(B_mux_out), .data_out(B0_reg_out)
);
MUX #(.WIDTH(18), .NUM_INPUTS(2)) B0_mux (
  .IN0(B_mux_out), .IN1(B0_reg_out), .IN2(18'b0), .IN3(18'b0), .sel(B0REG), .data_out(B0_out)
);
REG #(.WIDTH(18), .RSTTYPE(RSTTYPE)) B1_reg (
  .clk(CLK), .CE(CEB), .rst(RSTB), .data_in(OPMODE4_MUX_out), .data_out(B1_reg_out)
);
MUX #(.WIDTH(18), .NUM_INPUTS(2)) B1_mux (
  .IN0(OPMODE4_MUX_out), .IN1(B1_reg_out), .IN2(18'b0), .IN3(18'b0), .sel(B1REG), .data_out(B1_out)
);
assign BCOUT = B1_out;
```

RTL CODE

C PATH

```
  REG #(.WIDTH(48), .RSTTYPE(RSTTYPE)) C_reg (
    .clk(CLK), .CE(CEC), .rst(RSTC), .data_in(C), .data_out(C_reg_out)
  );
  MUX #(.WIDTH(48), .NUM_INPUTS(2)) C_mux (
    .IN0(C), .IN1(C_reg_out), .IN2(48'b0), .IN3(48'b0), .sel(CREG),
.data_out(C_out)
  );
```

D PATH

```
REG #(.WIDTH(18), .RSTTYPE(RSTTYPE)) D_reg (
    .clk(CLK), .CE(CED), .rst(RSTD), .data_in(D), .data_out(D_reg_out)
  );
  MUX #(.WIDTH(18), .NUM_INPUTS(2)) D_mux (
    .IN0(D), .IN1(D_reg_out), .IN2(18'b0), .IN3(18'b0), .sel(DREG), .data_out(D_out)
  );
```

OPMODE

```
REG #(.WIDTH(8), .RSTTYPE(RSTTYPE)) OPMODE_reg (
   .clk(CLK), .CE(CEOPMODE), .rst(RSTOPMODE), .data_in(OPMODE), .data_out(OPMODE_reg_out)
 );
 MUX #(.WIDTH(8), .NUM_INPUTS(2)) OPMODE_mux (
   .IN0(OPMODE), .IN1(OPMODE_reg_out), .IN2(8'b0), .IN3(8'b0), .sel(OPMODEREG), .data_out(OPMODE_out)
 );
```

CARRY-IN

```
  MUX #(.WIDTH(1), .NUM_INPUTS(2)) CARRYIN_mux (
    .IN0(CARRYIN), .IN1(OPMODE_out[5]), .IN2(1'b0), .IN3(1'b0), .sel(CARRYINSEL == "OPMODE5"),
.data_out(CARRYIN_mux_out)
  );
  REG #(.WIDTH(1), .RSTTYPE(RSTTYPE)) CARRYIN_reg (
    .clk(CLK), .CE(CECARRYIN), .rst(RSTCARRYIN), .data_in(CARRYIN_mux_out),
.data_out(CARRYIN_reg_out)
  );
  MUX #(.WIDTH(1), .NUM_INPUTS(2)) CARRYIN_final_mux (
    .IN0(CARRYIN_mux_out), .IN1(CARRYIN_reg_out), .IN2(1'b0), .IN3(1'b0), .sel(CARRYINREG),
.data_out(CARRYIN_out)
  );
```

PRE-ADDER/SUBTRACTOR

```
assign pre_adder_out = (OPMODE_out[6]) ? (D_out - B0_out) : (D_out + B0_out);
```

OPMODE4 MUX

```
assign OPMODE4_MUX_out = (OPMODE_out[4]) ? pre_adder_out : B0_out;
```

RTL CODE

MULTIPLIER

```
assign Multiplier_out = A1_out * B1_out;
  REG #(.WIDTH(36), .RSTTYPE(RSTTYPE)) M_reg (
    .clk(CLK), .CE(CEM), .rst(RSTM), .data_in(Multiplier_out), .data_out(M_reg_out)
  );
  MUX #(.WIDTH(36), .NUM_INPUTS(2)) M_mux (
    .IN0(Multiplier_out), .IN1(M_reg_out), .IN2(36'b0), .IN3(36'b0), .sel(MREG),
.data_out(M_out)
  );
  assign M = M_out;
```

CONCATENATE WIRES

```
assign Concatenate_wires = {D_out[11:0], A1_out, B1_out};
```

X MUX

```
 MUX #(.WIDTH(48), .NUM_INPUTS(4)) X_mux (
    .IN0(48'b0),
    .IN1({12'b0, M_out}),
    .IN2(P),
    .IN3(Concatenate_wires),
    .sel(OPMODE_out[1:0]), .data_out(X_mux_out)
  );
```

Z MUX

```
MUX #(.WIDTH(48), .NUM_INPUTS(4)) Z_mux (
    .IN0(48'b0),
    .IN1(PCIN),
    .IN2(P),
    .IN3(C_out),
    .sel(OPMODE_out[3:2]), .data_out(Z_mux_out)
  );
```

POST-ADDER/SUBTRACTOR

```
assign {CARRYOUT_out, post_adder_out} = (OPMODE_out[7]) ?
    (Z_mux_out - (X_mux_out + CARRYIN_out)) :
    (Z_mux_out + X_mux_out + CARRYIN_out);
```

CARRYOUT

```
REG #(.WIDTH(1), .RSTTYPE(RSTTYPE)) CARRYOUT_reg (
   .clk(CLK), .CE(CECARRYIN), .rst(RSTCARRYIN), .data_in(CARRYOUT_out), .data_out(CARRYOUT_reg_out)
 );
 MUX #(.WIDTH(1), .NUM_INPUTS(2)) CARRYOUT_mux (
   .IN0(CARRYOUT_out), .IN1(CARRYOUT_reg_out), .IN2(1'b0), .IN3(1'b0), .sel(CARRYOUTREG), .data_out(CARRYOUT)
 );
 assign CARRYOUTF = CARRYOUT;
```

RTL CODE

P OUTPUT

```
  REG #(.WIDTH(48), .RSTTYPE(RSTTYPE)) P_reg (
    .clk(CLK), .CE(CEP), .rst(RSTP), .data_in(post_adder_out), .data_out(P_reg_out)
  );
  MUX #(.WIDTH(48), .NUM_INPUTS(2)) P_mux (
    .IN0(post_adder_out), .IN1(P_reg_out), .IN2(48'b0), .IN3(48'b0), .sel(PREG),
.data_out(P)
  );
  assign PCOUT = P;
```

MODULES

MUX

```
module MUX #(
  parameter WIDTH = 1, // Width of the Mux,
  parameter NUM_INPUTS = 2 // Default Number of Inputs
  )(
  input [WIDTH-1:0] IN0,
  input [WIDTH-1:0] IN1,
  input [WIDTH-1:0] IN2,
  input [WIDTH-1:0] IN3,
  input [$clog2(NUM_INPUTS)-1:0] sel,
  output reg [WIDTH-1:0] data_out
);
  always @(*) begin
    case (sel)
      2'b00: data_out = IN0;
      2'b01: data_out = IN1;
      2'b10: data_out = IN2;
      2'b11: data_out = IN3;
      default: data_out = {WIDTH{1'b0}}; // Default case
    endcase
  end

endmodule
```

REG

```
module REG #(
  parameter WIDTH = 1, // Width of the register,
  parameter RSSTYPE = "SYNC" // Default TYPE of reset
  )(
  input wire clk,CE,
  input wire rst,
  input wire [WIDTH-1:0] data_in,
  output reg [WIDTH-1:0] data_out
);
  generate
  if (RSSTYPE == "ASYNC") begin
      always @(posedge clk or posedge rst) begin
        if (rst)
          data_out <= {WIDTH{1'b0}};
        else if (CE)
          data_out <= data_in;
      end
    end
    else begin // If -> RSTTYPE = "SYNC"
      always @(posedge clk) begin
        if (rst)
          data_out <= {WIDTH{1'b0}};
        else if (CE)
          data_out <= data_in;
    end
  end
  endgenerate

endmodule
```

TESTBENCH CODE

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 CARRYINREG = 1;
  parameter CARRYOUTREG = 1;
  parameter OPMODEREG = 1;
  parameter CARRYINSEL = "OPMODE5";
  parameter B_INPUT = "DIRECT";
  parameter RSTTYPE = "SYNC";
```

DATA PORTS

```
  reg [17:0] A;
  reg [17:0] B;
  reg [47:0] C;
  reg [17:0] D;
  reg    CARRYIN;

  wire [35:0] M;
  wire [47:0] P;
  wire    CARRYOUT;
  wire    CARRYOUTF;

  reg    CLK;
  reg [7:0] OPMODE;
  reg    CEA;
  reg    CEB;
  reg    CEC;
  reg    CECARRYIN;
  reg    CED;
  reg    CEM;
  reg    CEOPMODE;
  reg    CEP;

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

  reg [17:0] BCIN;
  reg [47:0] PCIN;
  wire [17:0] BCOUT;
  wire [47:0] PCOUT;
```

CLOCK GENERATION

```
initial begin
  CLK = 0;
  forever #1 CLK = ~CLK; // 2 time units clock period
 end
```

INSTANTIATE THE DSP48A1 MODULE

```
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)
  ) uut (
    .A(A),
    .B(B),
    .C(C),
    .D(D),
    .CARRYIN(CARRYIN),
    .M(M),
    .P(P),
    .CARRYOUT(CARRYOUT),
    .CARRYOUTF(CARRYOUTF),
    .CLK(CLK),
    .OPMODE(OPMODE),
    .CEA(CEA),
    .CEB(CEB),
    .CEC(CEC),
    .CECARRYIN(CECARRYIN),
    .CED(CED),
    .CEM(CEM),
    .CEOPMODE(CEOPMODE),
    .CEP(CEP),
    .RSTA(RSTA),
    .RSTB(RSTB),
    .RSTC(RSTC),
    .RSTCARRYIN(RSTCARRYIN),
    .RSTD(RSTD),
    .RSTM(RSTM),
    .RSTOPMODE(RSTOPMODE),
    .RSTP(RSTP),
    .BCIN(BCIN),
    .PCIN(PCIN),
    .BCOUT(BCOUT),
    .PCOUT(PCOUT)
  );
```

STIMULUS GENERATION (INITIAL BLOCK)

```
  initial begin
   // Initialize all inputs
   A = 0; B = 0; C = 0; D = 0; CARRYIN = 0; OPMODE = 0; BCIN = 0; PCIN = 0;
   CLK = 0; CEA = 0; CEB = 0; CEC = 0; CECARRYIN = 0; CED = 0; CEM = 0; CEOPMODE = 0; CEP = 0;
   //Verify Reset Operation
   RSTA = 1; RSTB = 1; RSTC = 1; RSTCARRYIN = 1; RSTD = 1; RSTM = 1; RSTOPMODE = 1; RSTP = 1;

   A = $random; B = $random; C = $random; D = $random; CARRYIN = $random;
   OPMODE = $random;
   CEA = $random; CEB = $random; CEC = $random;
   CECARRYIN = $random; CED = $random; CEM = $random; CEOPMODE = $random; CEP = $random;
   @ (negedge CLK);
   if (M == 0 && P == 0 && CARRYOUT == 0 && CARRYOUTF == 0 && BCOUT == 0 && PCOUT ==0) begin
    $display("Reset successful, outputs are zero.");
   end else begin
    $display("Reset failed, outputs are not zero./n M: %h, P: %h, CARRYOUT: %b, CARRYOUTF: %b,
BCOUT: %h, PCOUT: %h", M, P, CARRYOUT, CARRYOUTF, BCOUT, PCOUT);
    $stop;
   end

   // Deassert reset
   RSTA = 0; RSTB = 0; RSTC = 0; RSTCARRYIN = 0; RSTD = 0; RSTM = 0; RSTOPMODE = 0; RSTP = 0;
   CEA = 1; CEB = 1; CEC = 1; CECARRYIN = 1; CED = 1; CEM = 1; CEOPMODE = 1; CEP = 1;
```

VERIFY DSP PATH 1

```
   OPMODE = 8'b11011101;
   A = 20; B = 10; C = 350; D = 25;
   BCIN= $random; PCIN = $random; CARRYIN = $random;
   repeat(4) @(negedge CLK);
   if (BCOUT == 18'hf && M == 36'h12c && P == 48'h32 && PCOUT == 48'h32 && CARRYOUT == 0 &&
CARRYOUTF == 0) begin
     $display("DSP Path 1 successful");
   end else begin
     $display("DSP Path 1 failed");
     $stop;
   end
```

VERIFY DSP PATH 2

```
   OPMODE = 8'b00010000;
   A = 20; B = 10; C = 350; D = 25;
   BCIN= $random; PCIN = $random; CARRYIN = $random;
   repeat(3) @(negedge CLK);
   if (BCOUT == 18'h23 && M == 36'h2bc && P == 0 && PCOUT == 0 && CARRYOUT == 0 && CARRYOUTF ==
0) begin
     $display("DSP Path 2 successful");
   end else begin
     $display("DSP Path 2 failed");
     $stop;
   end
```

VERIFY DSP PATH 3

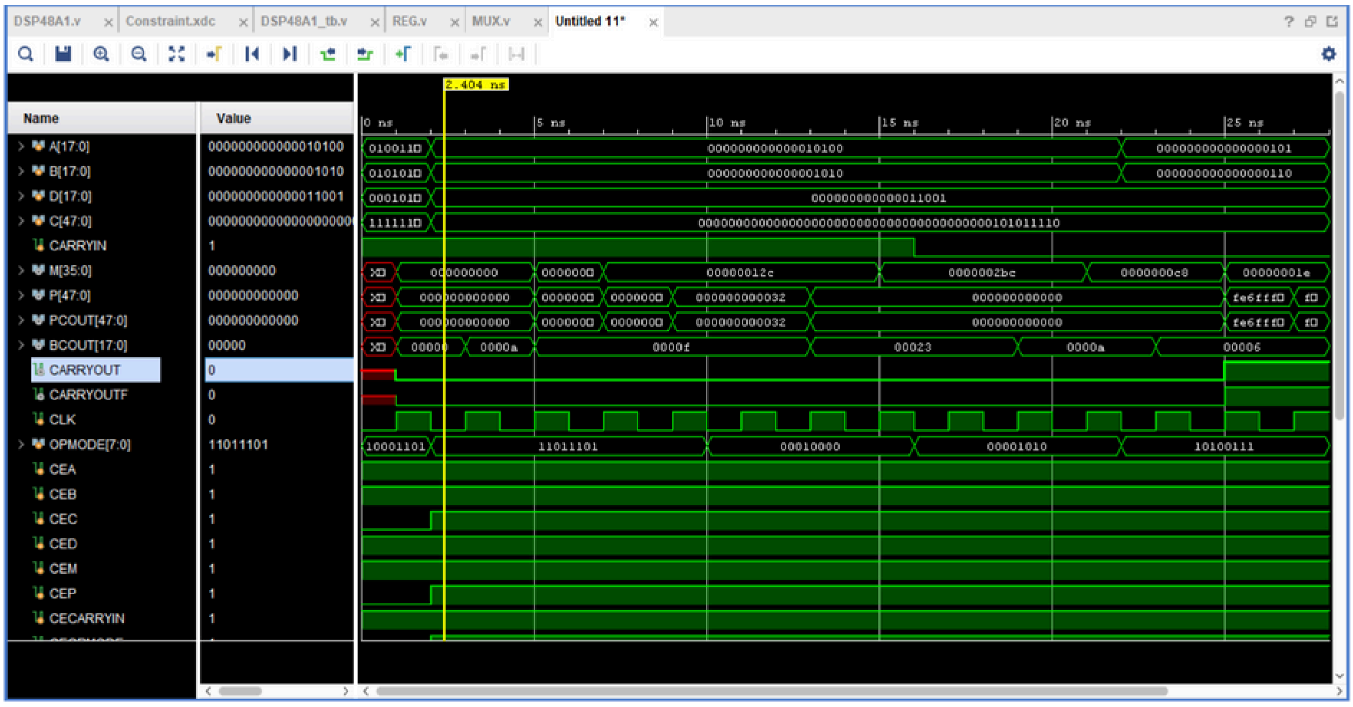
```
   OPMODE = 8'b00001010;
   A = 20; B = 10; C = 350; D = 25;
   BCIN= $random; PCIN = $random; CARRYIN = $random;
   repeat(3) @(negedge CLK);
   if (BCOUT == 18'ha && M == 36'hc8 && P == P && PCOUT == P && CARRYOUT == CARRYOUT &&
CARRYOUTF == CARRYOUTF) begin
     $display("DSP Path 3 successful");
   end else begin
     $display("DSP Path 3 failed");
     $stop;
   end
```

VERIFY DSP PATH 4

```
   OPMODE = 8'b10100111;
   A = 5; B = 6; C = 350; D = 25; PCIN = 3000;
   BCIN= $random; CARRYIN = $random;
   repeat(3) @(negedge CLK);
   if (BCOUT == 18'h6 && M == 36'h1e && P == 48'hfe6fffec0bb1 && PCOUT == 48'hfe6fffec0bb1 &&
CARRYOUT == 1 && CARRYOUTF == 1) begin
     $display("DSP Path 4 successful");
   end else begin
     $display("DSP Path 4 failed");
     $stop;
   end
```

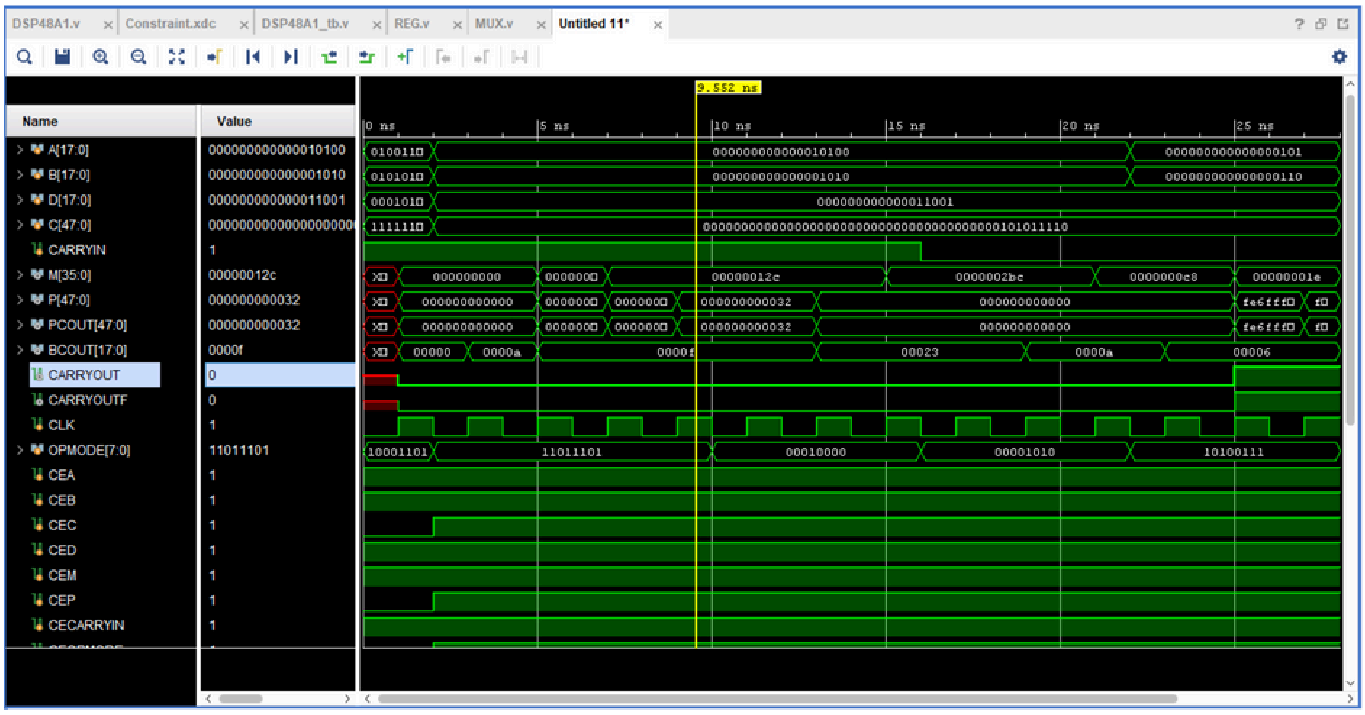
SIMULATION RESULTS

VERIFY RESET



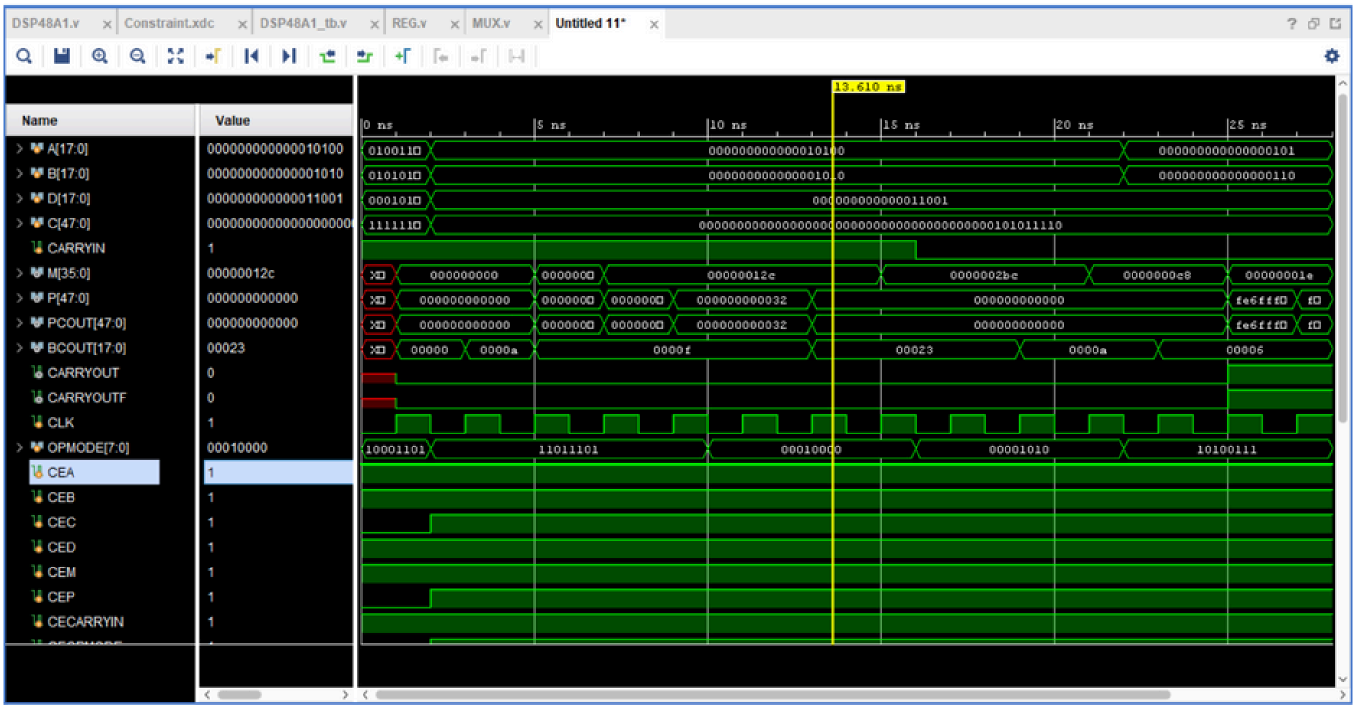
```
Name            Value
A[17:0]         000000000000010100
B[17:0]         000000000000001010
D[17:0]         000000000000011001
C[47:0]         0000000000000000000
CARRYIN         1
M[35:0]         000000000
P[47:0]         000000000000
PCOUT[47:0]     000000000000
BCOUT[17:0]     00000
CARRYOUT        0
CARRYOUTF       0
CLK             0
OPMODE[7:0]     11011101
CEA             1
CEB             1
CEC             1
CED             1
CEM             1
CEP             1
CECARRYIN       1
```

VERIFY DSP PATH 1



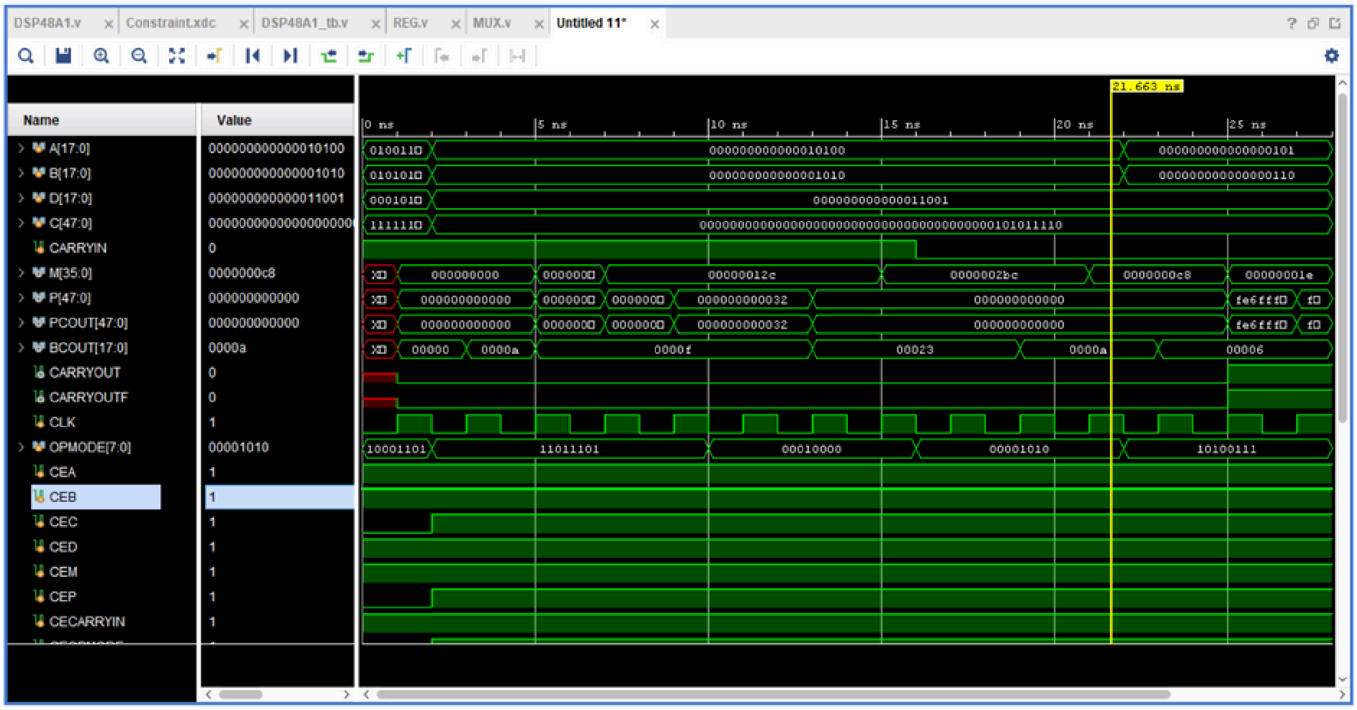
```
Name            Value
A[17:0]         000000000000010100
B[17:0]         000000000000001010
D[17:0]         000000000000011001
C[47:0]         0000000000000000000
CARRYIN         1
M[35:0]         00000012c
P[47:0]         000000000032
PCOUT[47:0]     000000000032
BCOUT[17:0]     0000f
CARRYOUT        0
CARRYOUTF       0
CLK             1
OPMODE[7:0]     11011101
CEA             1
CEB             1
CEC             1
CED             1
CEM             1
CEP             1
CECARRYIN       1
```

VERIFY DSP PATH 2



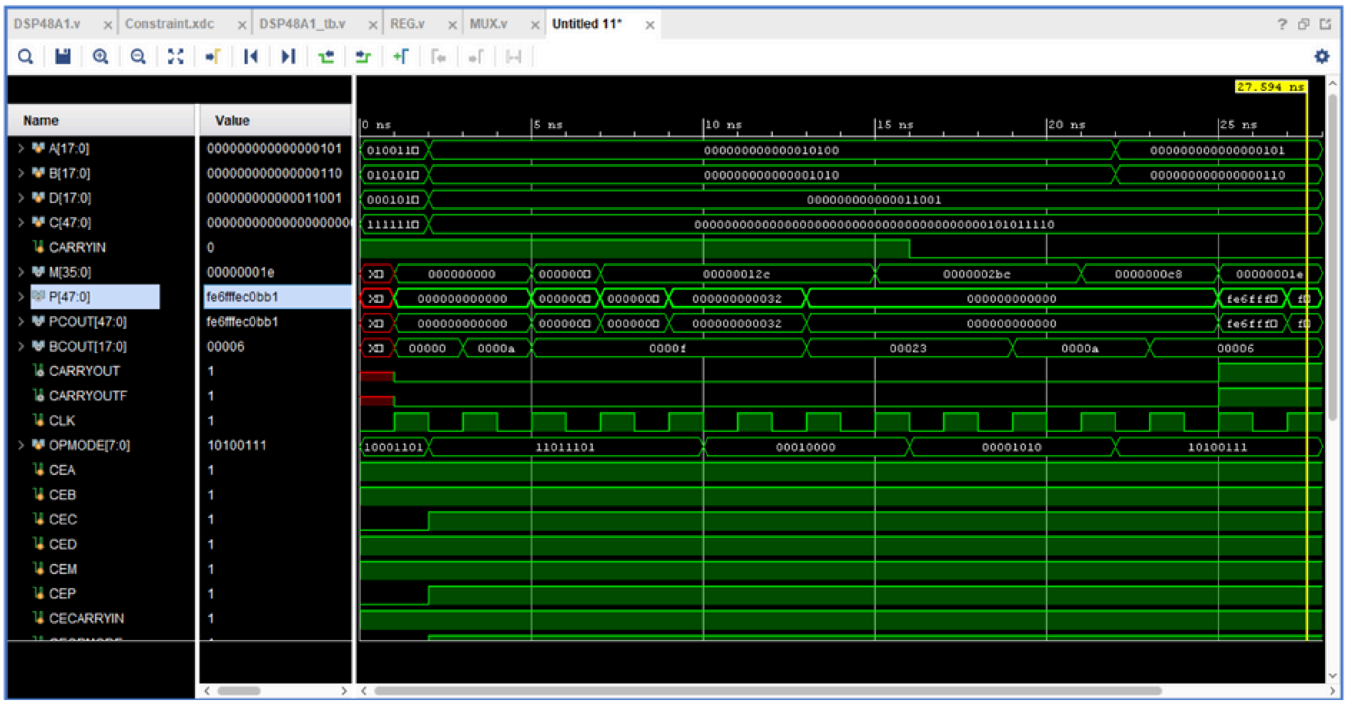
Name	Value
A[17:0]	000000000000010100
B[17:0]	000000000000001010
D[17:0]	000000000000011001
C[47:0]	0000000000000000000
CARRYIN	1
M[35:0]	00000012c
P[47:0]	000000000000
PCOUT[47:0]	000000000000
BCOUT[17:0]	00023
CARRYOUT	0
CARRYOUTF	0
CLK	1
OPMODE[7:0]	00010000
CEA	1
CEB	1
CEC	1
CED	1
CEM	1
CEP	1
CECARRYIN	1

VERIFY DSP PATH 3



Name	Value
A[17:0]	000000000000010100
B[17:0]	000000000000001010
D[17:0]	000000000000011001
C[47:0]	0000000000000000000
CARRYIN	0
M[35:0]	0000000c8
P[47:0]	000000000000
PCOUT[47:0]	000000000000
BCOUT[17:0]	0000a
CARRYOUT	0
CARRYOUTF	0
CLK	1
OPMODE[7:0]	00001010
CEA	1
CEB	1
CEC	1
CED	1
CEM	1
CEP	1
CECARRYIN	1

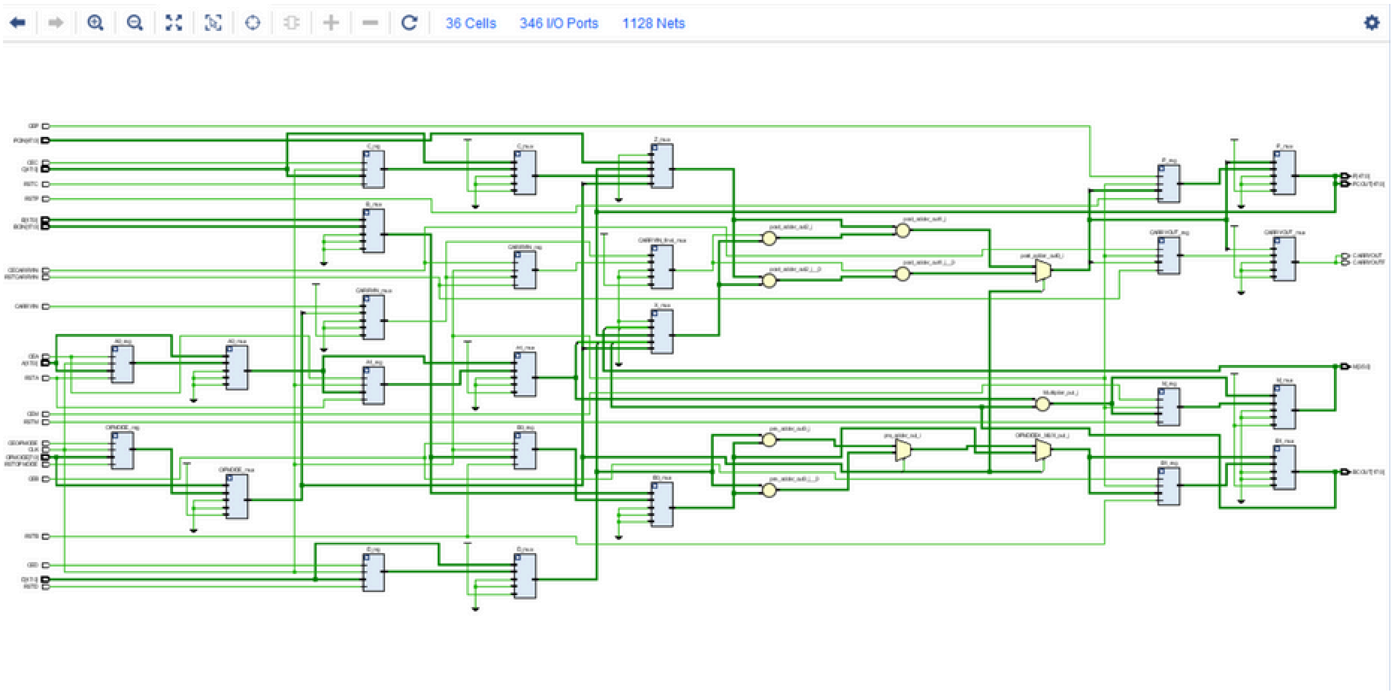
VERIFY DSP PATH 4



Name	Value
A[17:0]	000000000000000101
B[17:0]	000000000000000110
D[17:0]	000000000000011001
C[47:0]	0000000000000000000
CARRYIN	0
M[35:0]	00000001e
P[47:0]	fe6fffec0bb1
PCOUT[47:0]	fe6fffec0bb1
BCOUT[17:0]	00006
CARRYOUT	1
CARRYOUTF	1
CLK	1
OPMODE[7:0]	10100111
CEA	1
CEB	1
CEC	1
CED	1
CEM	1
CEP	1
CECARRYIN	1

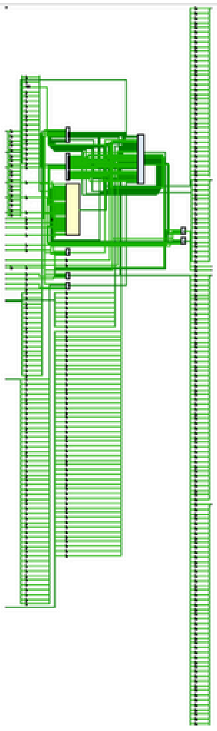
ELABORATION

SCHEMATIC

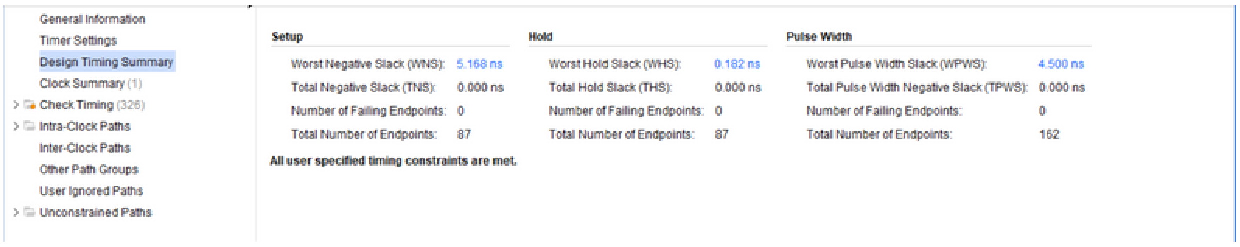


SYNTHESIS

SCHEMATIC



TIMING REPORT

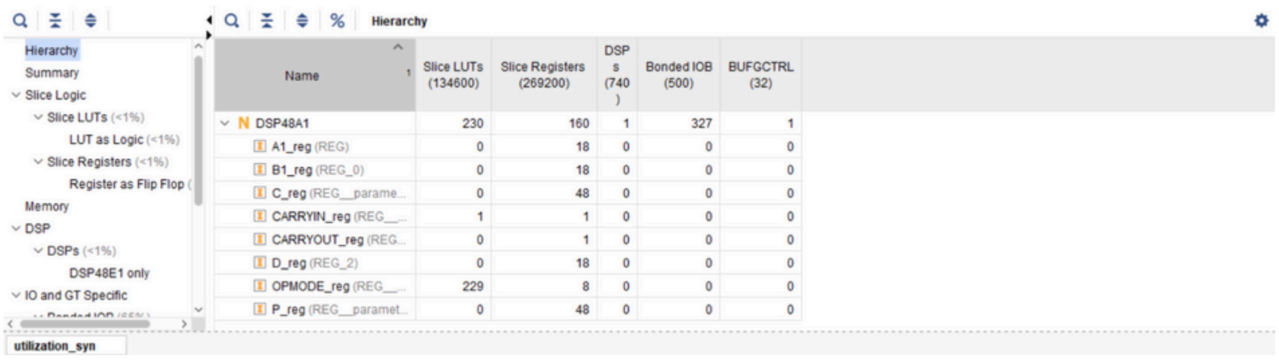


General Information
Timer Settings
Design Timing Summary
Clock Summary (1)
Check Timing (326)
Intra-Clock Paths
Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths

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:	87	Total Number of Endpoints:	87	Total Number of Endpoints:	162

All user specified timing constraints are met.

UTILIZATION REPORT



Hierarchy
Summary
Slice Logic
Slice LUTs (<1%)
LUT as Logic (<1%)
Slice Registers (<1%)
Register as Flip Flop (
Memory
DSP
DSPs (<1%)
DSP48E1 only
IO and GT Specific

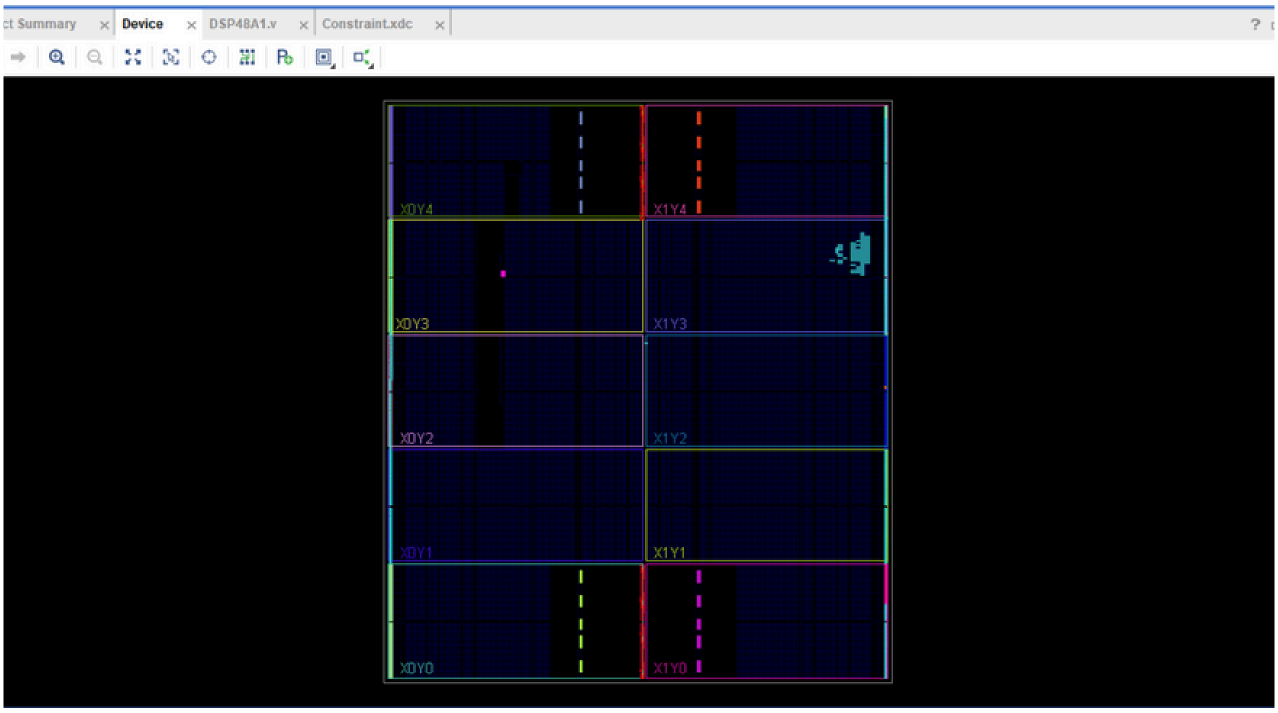
Hierarchy

Name	Slice LUTs (134600)	Slice Registers (269200)	DSPs (740)	Bonded IOB (500)	BUFGCTRL (32)
DSP48A1	230	160	1	327	1
A1_reg (REG)	0	18	0	0	0
B1_reg (REG_0)	0	18	0	0	0
C_reg (REG__parame...	0	48	0	0	0
CARRYIN_reg (REG__...	1	1	0	0	0
CARRYOUT_reg (REG...	0	1	0	0	0
D_reg (REG_2)	0	18	0	0	0
OPMODE_reg (REG__...	229	8	0	0	0
P_reg (REG__paramet...	0	48	0	0	0

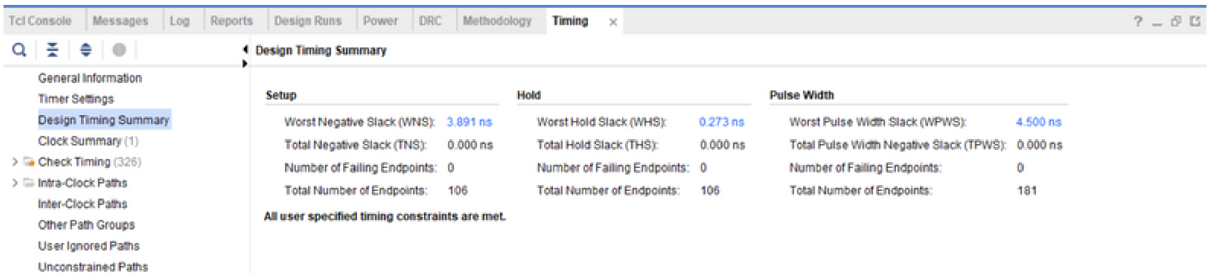
utilization_syn

IMPLEMENTATION

SCHEMATIC



TIMING REPORT

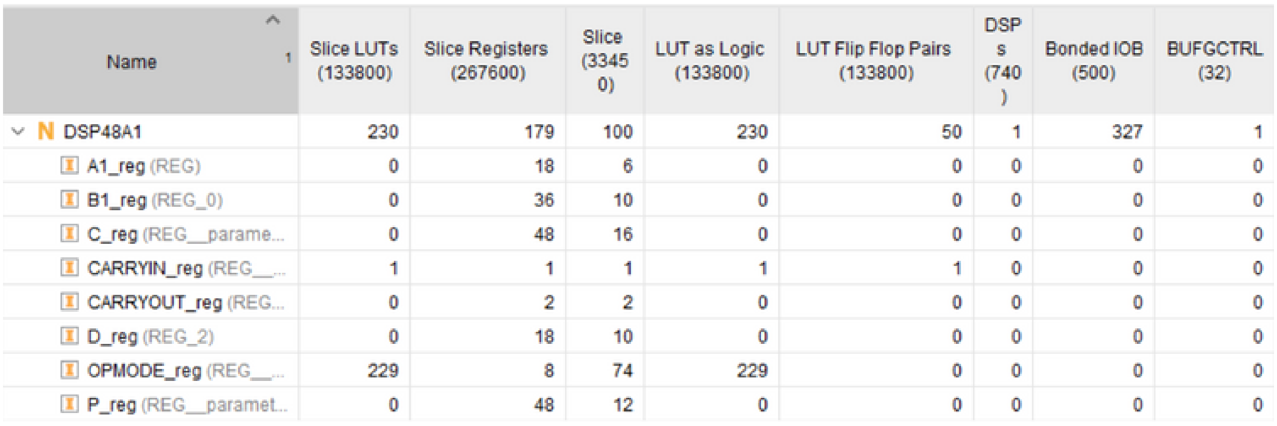


Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.891 ns	Worst Hold Slack (WHS):	0.273 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:	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)
DSP48A1	230	179	100	230	50	1	327	1
A1_reg (REG)	0	18	6	0	0	0	0	0
B1_reg (REG_0)	0	36	10	0	0	0	0	0
C_reg (REG__parame...	0	48	16	0	0	0	0	0
CARRYIN_reg (REG__...	1	1	1	1	1	0	0	0
CARRYOUT_reg (REG...	0	2	2	0	0	0	0	0
D_reg (REG_2)	0	18	10	0	0	0	0	0
OPMODE_reg (REG__...	229	8	74	229	0	0	0	0
P_reg (REG__paramet...	0	48	12	0	0	0	0	0