Spartan6 DSP48A1

FPGA Flow

Prepared by : Yousef Elbadry

1. Spartan-6 DSP48A1 Overview

1.1. Description

A flexible, high-performance DSP block designed for digital signal processing applications is the Spartan-6 FPGA family's DSP48A1 slice. It is incorporated into the FPGA fabric to improve performance and computational efficiency.

1.2. Configuration and Usage

The DSP48A1 block can be configured using Xilinx's design tools (e.g., Vivado) to match the specific needs of the application. It offers flexibility in terms of input and output data width and supports various mathematical functions.

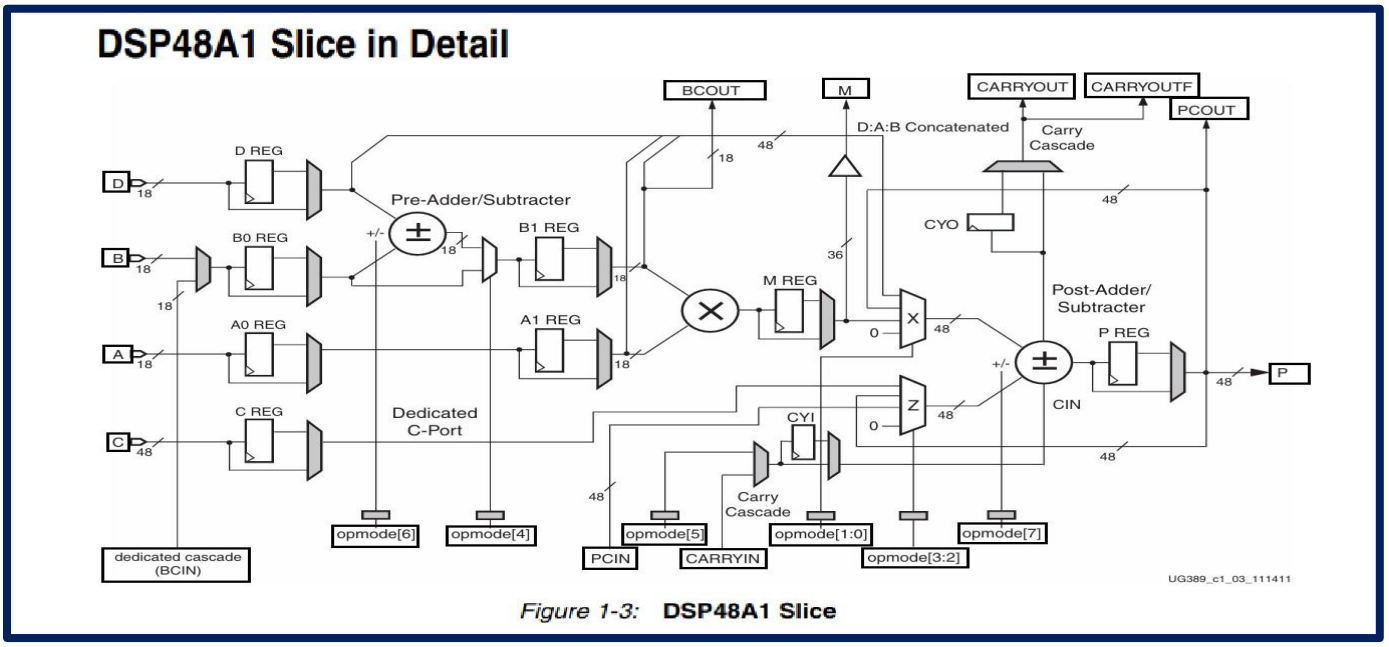


Figure 1-3: **DSP48A1 Slice**

2. RTL Code

```
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 [17:0]D,
        input [47:0]C,
        input clk,CARRYIN,
        input [7:0]OPMODE,
        input [17:0]BCIN,
        input RSTA,RSTB,RSTM,RSTP,RSTC,RSTD,RSTCARRYIN,RSTOPMODE, //active high resets
                CEA,CEB,CEM,CEP,CEC,CED,CECARRYIN,CEOPMODE, //clock enable
        input [47:0]PCIN,
        output [17:0]BCOUT,
        output [47:0]PCOUT,
        output reg [47:0]P,
        output [35:0]M,
        output reg CARRYOUT,
        output CARRYOUTF
        );

reg [7:0] OPMODE_out,WOPMODE;
reg [17:0] WA0 , A0_out, WA1, A1_out, WB0_in, WB0, B0_out, WB1_in, B1_out, WB1, WD ,D_out ,WPAS;
reg [47:0] WC , C_out, WP_in, P_out, Wx, Wz;
reg [35:0] WM, M_out;
reg WCYI, WCYI_in , CYI_out, CYO_out, WCYO_in;

wire [35:0] WM_in;
wire [47:0] Wconc;


assign CARRYOUTF = CARRYOUT ;
assign BCOUT = WB1;
assign PCOUT = P;
assign M =~(~WM);
assign WM_in = WA1 * WB1;
assign Wconc ={ WD[11:0] , WA1[17:0] , WB1[17:0] } ;


//**************************************RST IS SYNCHRONUS*************************************************
if (RSTTYPE=="SYNC")
begin
//************************OP MODE REG**************************
always @(posedge clk)
begin
        if (RSTOPMODE)
                OPMODE_out<=0;
        else if(CEOPMODE)
                OPMODE_out<=OPMODE;
end
//************************A0 REG*******************************
always @(posedge clk)
begin
        if (RSTA)
                A0_out<=0;
        else if(CEA)
                A0_out<=A;
end
```

```
//*************************B0 REG*********************************
always @(posedge clk)
begin
        if (RSTB)
                B0_out<=0;
        else if(CEB)
                B0_out<=WB0_in;
end
//*************************C REG**********************************
always @(posedge clk)
begin
        if (RSTC)
                C_out<=0;
        else if(CEC)
                C_out<=C;
end
//*************************D REG**********************************
always @(posedge clk)
begin
        if (RSTD)
                D_out<=0;
        else if(CED)
                D_out<=D;
end
//*************************B1 REG**********************************
always @(posedge clk)
begin
        if (RSTB)
                B1_out<=0;
        else if(CEB)
                B1_out<=WB1_in;
end
//*************************A1 REG**********************************
always @(posedge clk)
begin
        if (RSTA)
                A1_out<=0;
        else if(CEA)
                A1_out<=WA0;
end
//*************************M REG**********************************
always @(posedge clk)
begin
        if (RSTM)
                M_out<=0;
        else if(CEM)
                M_out<=WM_in;
end
//*************************CYI REG**********************************
always @(posedge clk)
begin
        if (RSTCARRYIN)
                CYI_out<=0;
        else if(CECARRYIN)
                CYI_out<=WCYI_in;
end
//*************************CYO REG**********************************
always @(posedge clk)
begin
        if (RSTCARRYIN)
                CYO_out<=0;
        else if(CECARRYIN)
                CYO_out<=WCYO_in;
end
//*************************P REG**********************************
always @(posedge clk)
```

```
begin
        if (RSTP)
                P_out<=0;
        else if(CEP)
                P_out<=WP_in;
end

end

//*******************************RST IS ASYNCHRONUS*****************************************************************
else if (RSTTYPE=="ASYNC")
begin
//************************OP MODE REG**************************
always @(posedge clk or posedge RSTOPMODE)
begin
        if (RSTOPMODE)
                OPMODE_out<=0;
        else if(CEOPMODE)
                OPMODE_out<=OPMODE;
end
//************************A0 REG*******************************
always @(posedge clk or posedge RSTA)
begin
        if (RSTA)
                A0_out<=0;
        else if(CEA)
                A0_out<=A;
end
//************************B0 REG*******************************
always @(posedge clk or posedge RSTB)
begin
        if (RSTB)
                B0_out<=0;
        else if(CEB)
                B0_out<=WB0_in;
end
//************************C REG*********************************
always @(posedge clk or posedge RSTC)
begin
        if (RSTC)
                C_out<=0;
        else if(CEC)
                C_out<=C;
end
//************************D REG*********************************
always @(posedge clk or posedge RSTD)
begin
        if (RSTD)
                D_out<=0;
        else if(CED)
                D_out<=D;
end
//************************B1 REG*********************************
always @(posedge clk or posedge RSTB)
begin
        if (RSTB)
                B1_out<=0;
        else if(CEB)
                B1_out<=WB1_in;
end
//************************A1 REG*********************************
always @(posedge clk or posedge RSTA)
begin
        if (RSTA)
                A1_out<=0;
        else if(CEA)
```

```
                A1_out<=WA0;
end
//************************M REG************************************
always @(posedge clk or posedge RSTM)
begin
        if (RSTM)
                M_out<=0;
        else if(CEM)
                M_out<=WM_in;
end
//************************CYI REG**********************************
always @(posedge clk or posedge RSTCARRYIN)
begin
        if (RSTCARRYIN)
                CYI_out<=0;
        else if(CECARRYIN)
                CYI_out<=WCYI_in;
end
//************************CYO REG**********************************
always @(posedge clk or posedge RSTCARRYIN)
begin
        if (RSTCARRYIN)
                CYO_out<=0;
        else if(CECARRYIN)
                CYO_out<=WCYO_in;
end
//***************************P REG************************************
always @(posedge clk or posedge RSTP)
begin
        if (RSTP)
                P_out<=0;
        else if(CEP)
                P_out<=WP_in;
end

end

//*******************************COMBINATIONAL ALWAYS BLOCKS FOR MUXES*********************************
//************************OP MODE REG**************************
always @(*)
begin
        if (OPMODEREG)
                WOPMODE = OPMODE_out;
        else
                WOPMODE = OPMODE;
end
//************************A0 REG*******************************
always @(*)
begin
        if (A0REG)
                WA0 = A0_out;
        else
                WA0 = A;
end
//************************B0 REG*******************************
always @(*)
begin
        if (B_INPUT=="DIRECT")
                WB0_in=B;
        else
                WB0_in=BCIN;

        if (B0REG)
                WB0 = B0_out;
        else
                WB0 = WB0_in;
```

```
end
//************************C REG**********************************
always @(*)
begin
        if (CREG)
                WC = C_out;
        else
                WC = C;
end
//************************D REG**********************************
always @(*)
begin
        if (DREG)
                WD = D_out;
        else
                WD = D;
end
//*********************Pre adder subtractor*************************
always @(*)
begin
        if(WOPMODE[6]==0)
                WPAS = WD + WB0;
        else
                WPAS = WD - WB0;
end
//************************B1 REG**********************************
always @(*)
begin
        if(WOPMODE[4]==0)
                WB1_in=WB0;
        else
                WB1_in=WPAS;

        if (B1REG)
                WB1 = B1_out;
        else
                WB1 = WB1_in;
end
//************************A1 REG**********************************
always @(*)
begin
        if (A1REG)
                WA1 = A1_out;
        else
                WA1 = WA0;
end
//************************M REG**********************************
always @(*)
begin
        if (MREG)
                WM = M_out;
        else
                WM = WM_in;
end
//************************X MUX**********************************
always @(*)
begin
        case(WOPMODE[1:0])
        2'b00:Wx=0;
        2'b01:Wx={12'b000000000000,WM};
        2'b10:Wx=P;
        2'b11:Wx=Wconc;
        endcase
end
//************************Z MUX**********************************
always @(*)
```

```
begin
        case(WOPMODE[3:2])
        2'b00:Wz=0;
        2'b01:Wz=PCIN;
        2'b10:Wz=P;
        2'b11:Wz=WC;
        endcase
end
//************************CYI REG***************************************
always @(*)
begin
        if(CARRYINSEL=="OPMODE5")
                WCYI_in=WOPMODE[5];
        else
                WCYI_in=CARRYIN;

        if (CARRYINREG)
                WCYI = CYI_out;
        else
                WCYI = WCYI_in;
end
//************************CYO REG***************************************
always @(*)
begin
        if (CARRYOUTREG)
                CARRYOUT = CYO_out;
        else
                CARRYOUT = WCYO_in;
end
//**********************POST adder subtractor*****************************
always @(*)
begin
        if(WOPMODE[7]==0)
                {WCYO_in,WP_in} = Wz + Wx + WCYI ;
        else
                {WCYO_in,WP_in} = Wz - ( Wx + WCYI );
end
//**************************P REG*************************************
always @(*)
begin
        if (PREG)
                P = P_out;
        else
                P = WP_in;
end

endmodule
```

3. Testbench Code

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

//inputs and outputs
reg [17:0]A;
reg [17:0]B;
reg [17:0]D;
reg [47:0]C;
reg clk,CARRYIN;
reg [7:0]OPMODE;
reg [17:0]BCIN;
reg RSTA,RSTB,RSTM,RSTP,RSTC,RSTD,RSTCARRYIN,RSTOPMODE, //active high resets
    CEA,CEB,CEM,CEP,CEC,CED,CECARRYIN,CEOPMODE; //clock enable
reg [47:0]PCIN;
wire [17:0]BCOUT;
wire [47:0]PCOUT;
wire [47:0]P;
wire [35:0]M;
wire CARRYOUT,CARRYOUTF;

// Instantiate the DUT
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 (
  .A(A),
  .B(B),
  .D(D),
  .C(C),
  .clk(clk),
  .CARRYIN(CARRYIN),
  .OPMODE(OPMODE),
  .BCIN(BCIN),
  .RSTA(RSTA),
  .RSTB(RSTB),
  .RSTM(RSTM),
  .RSTP(RSTP),
  .RSTC(RSTC),
  .RSTD(RSTD),
  .RSTCARRYIN(RSTCARRYIN),
  .RSTOPMODE(RSTOPMODE),
  .CEA(CEA),
  .CEB(CEB),
  .CEM(CEM),
  .CEP(CEP),
  .CEC(CEC),
  .CED(CED),
```

```
   .CECARRYIN(CECARRYIN),
   .CEOPMODE(CEOPMODE),
   .PCIN(PCIN),
   .BCOUT(BCOUT),
   .PCOUT(PCOUT),
   .P(P),
   .M(M),
   .CARRYOUT(CARRYOUT),
   .CARRYOUTF(CARRYOUTF)
);

// Clock generation
initial begin
   clk = 0;
   forever #5 clk = ~clk;
end

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

   // Reset the design
   RSTA = 1; RSTB = 1; RSTM = 1; RSTP = 1; RSTC = 1; RSTD = 1; RSTCARRYIN = 1; RSTOPMODE = 1;

   A=18'd50; B=18'd20; C=48'd70; D=18'd90; PCIN=48'd150; CARRYIN=1; BCIN=18'd250;
   repeat(4) @(negedge clk);

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

   //*****************************************************************************
   //test add -- add Xmux paths (00) & Zmux paths (00) with CIN(opmode[5]) =1
   OPMODE=8'b00110000;
   repeat(2) @(negedge clk);

   //test add -- add Xmux paths (00) & Zmux paths (01) with CIN(opmode[5]) =1
   OPMODE=8'b00110100;
   repeat(2) @(negedge clk);

   //test add -- add Xmux paths (00) & Zmux paths (10) with CIN(opmode[5]) =0
   OPMODE=8'b00101000;
   repeat(1) @(negedge clk);

   //test add -- add Xmux paths (00) & Zmux paths (11) with CIN(opmode[5]) =1
   OPMODE=8'b00111100;
   repeat(3) @(negedge clk);
   //*****************************************************************************
   //test add -- add Xmux paths (00) & Zmux paths (11) with CIN(opmode[5]) =1
   OPMODE=8'b10111100;
   repeat(3) @(negedge clk);
   //*****************************************************************************
   //test add -- add Xmux paths (01) & Zmux paths (00) with CIN(opmode[5]) =1
   OPMODE=8'b00110001;
   repeat(4) @(negedge clk);

   //test add -- add Xmux paths (01) & Zmux paths (01) with CIN(opmode[5]) =1
   OPMODE=8'b00110101;
   repeat(4) @(negedge clk);

   //test add -- add Xmux paths (01) & Zmux paths (10) with CIN(opmode[5]) =1
   OPMODE=8'b00111001;
   repeat(4) @(negedge clk);

   //test add -- add Xmux paths (01) & Zmux paths (11) with CIN(opmode[5]) =1
```

```
OPMODE=8'b00111101;
repeat(4) @(negedge clk);
//*************************************************************************
//test sub -- add Xmux paths (01) & Zmux paths (00) with CIN(opmode[5]) =1
OPMODE=8'b01110001;
repeat(4) @(negedge clk);

//test sub -- add Xmux paths (01) & Zmux paths (01) with CIN(opmode[5]) =1
OPMODE=8'b01110101;
repeat(4) @(negedge clk);

//test sub -- add Xmux paths (01) & Zmux paths (10) with CIN(opmode[5]) =1
OPMODE=8'b01111001;
repeat(4) @(negedge clk);

//test sub -- add Xmux paths (01) & Zmux paths (11) with CIN(opmode[5]) =1
OPMODE=8'b01111101;
repeat(4) @(negedge clk);
//*************************************************************************
//test add -- add Xmux paths (10) & Zmux paths (00) with CIN(opmode[5]) =1
OPMODE=8'b00110010;
repeat(2) @(negedge clk);

//test add -- add Xmux paths (10) & Zmux paths (01) with CIN(opmode[5]) =1
OPMODE=8'b00110110;
repeat(2) @(negedge clk);

//test add -- add Xmux paths (10) & Zmux paths (10) with CIN(opmode[5]) =1
OPMODE=8'b00111010;
repeat(2) @(negedge clk);

//test add -- add Xmux paths (10) & Zmux paths (11) with CIN(opmode[5]) =1
OPMODE=8'b00111110;
repeat(2) @(negedge clk);
 //*************************************************************************
//test add -- sub Xmux paths (10) & Zmux paths (00) with CIN(opmode[5]) =1
OPMODE=8'b10110010;
repeat(2) @(negedge clk);

//test add -- sub Xmux paths (10) & Zmux paths (01) with CIN(opmode[5]) =1
OPMODE=8'b10110110;
repeat(2) @(negedge clk);

//test add -- sub Xmux paths (10) & Zmux paths (10) with CIN(opmode[5]) =1
OPMODE=8'b10111010;
repeat(2) @(negedge clk);

//test add -- sub Xmux paths (10) & Zmux paths (11) with CIN(opmode[5]) =1
OPMODE=8'b10111110;
repeat(2) @(negedge clk);
//*************************************************************************
//test add -- add Xmux paths (11) & Zmux paths (00) with CIN(opmode[5]) =1
OPMODE=8'b00110011;
repeat(3) @(negedge clk);

//test add -- add Xmux paths (11) & Zmux paths (01) with CIN(opmode[5]) =1
OPMODE=8'b00110111;
repeat(3) @(negedge clk);

//test add -- add Xmux paths (11) & Zmux paths (10) with CIN(opmode[5]) =1
OPMODE=8'b00111011;
repeat(3) @(negedge clk);

//test add -- add Xmux paths (11) & Zmux paths (11) with CIN(opmode[5]) =1
OPMODE=8'b00111111;
repeat(3) @(negedge clk);
```

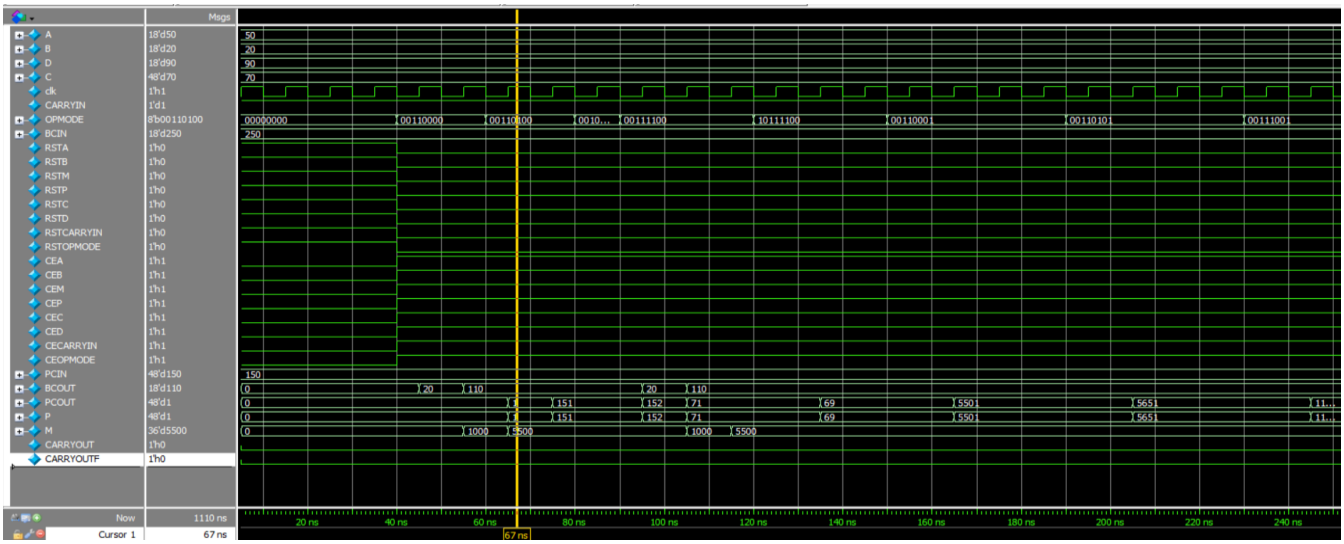
```
  //*************************************************************************
  //test add -- sub Xmux paths (11) & Zmux paths (00) with CIN(opmode[5]) =1
  OPMODE=8'b10110011;
  repeat(3) @(negedge clk);

  //test add -- sub Xmux paths (11) & Zmux paths (01) with CIN(opmode[5]) =1
  OPMODE=8'b10110111;
  repeat(3) @(negedge clk);

  //test add -- sub Xmux paths (11) & Zmux paths (10) with CIN(opmode[5]) =1
  OPMODE=8'b10111011;
  repeat(3) @(negedge clk);

  //test add -- sub Xmux paths (11) & Zmux paths (11) with CIN(opmode[5]) =1
  OPMODE=8'b10111111;
  repeat(3) @(negedge clk);
  //*************************************************************************
  //test sub -- add Xmux paths (11) & Zmux paths (00) with CIN(opmode[5]) =1
  OPMODE=8'b01110011;
  repeat(3) @(negedge clk);

  //test sub -- add Xmux paths (11) & Zmux paths (01) with CIN(opmode[5]) =1
  OPMODE=8'b01110111;
  repeat(3) @(negedge clk);

  //test sub -- add Xmux paths (11) & Zmux paths (10) with CIN(opmode[5]) =1
  OPMODE=8'b01111011;
  repeat(3) @(negedge clk);

  //test sub -- add Xmux paths (11) & Zmux paths (11) with CIN(opmode[5]) =1
  OPMODE=8'b01111111;
  repeat(3) @(negedge clk);
  //*************************************************************************
  //test sub -- sub Xmux paths (11) & Zmux paths (00) with CIN(opmode[5]) =1
  OPMODE=8'b11110011;
  repeat(3) @(negedge clk);

  //test sub -- sub Xmux paths (11) & Zmux paths (01) with CIN(opmode[5]) =1
  OPMODE=8'b11110111;
  repeat(3) @(negedge clk);

  //test sub -- sub Xmux paths (11) & Zmux paths (10) with CIN(opmode[5]) =1
  OPMODE=8'b11111011;
  repeat(3) @(negedge clk);

  //test sub -- sub Xmux paths (11) & Zmux paths (11) with CIN(opmode[5]) =1
  OPMODE=8'b11111111;
  repeat(3) @(negedge clk);

$stop;
end
endmodule
```

4. Do File

```
vlib work
vlog DSP48A1.v DSP48A1_tb.v
vsim -voptargs=+acc work.DSP48A1_tb
add wave *
run -all
#quit -sim
```

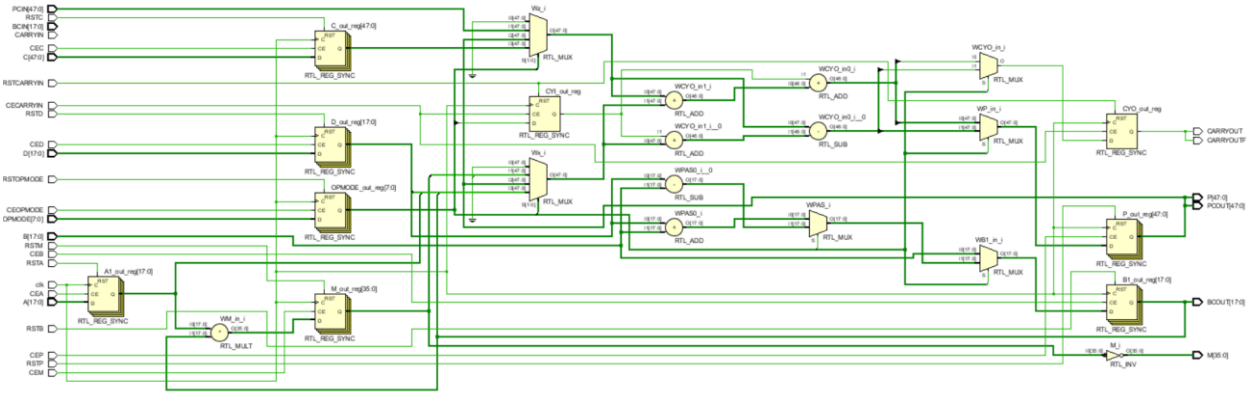
5. Waveform

A=18'd50; B=18'd20; C=48'd70; D=18'd90; PCIN=48'd150; CARRYIN=1; BCIN=18'd250;

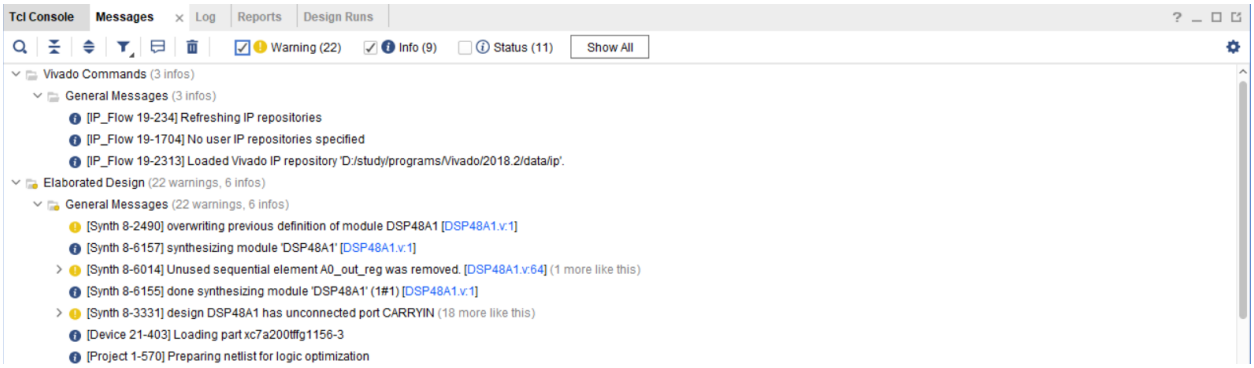


6. Elaboration

6.1 Schematic

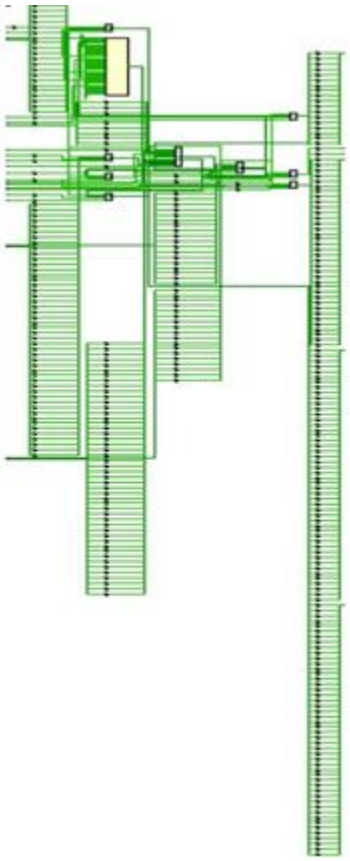


6.2 Message tab

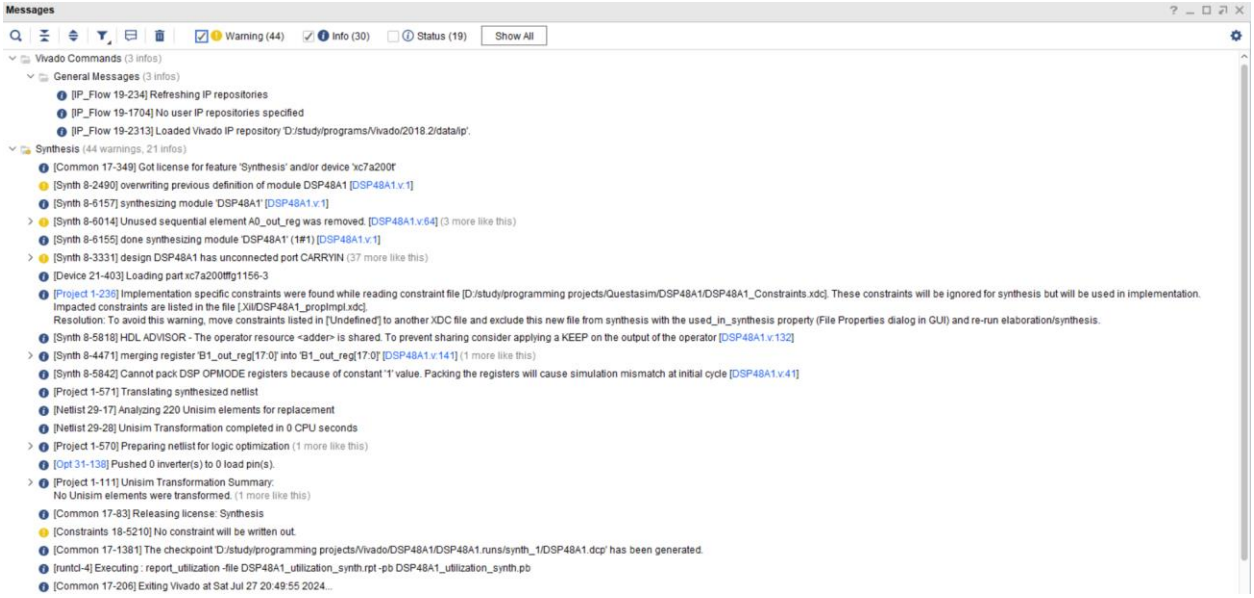


7. Synthesis

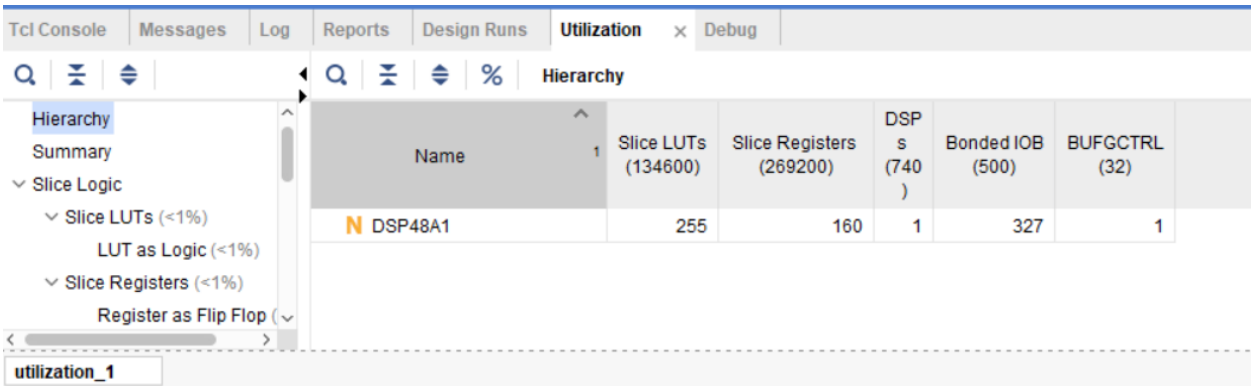
7.1 Schematic

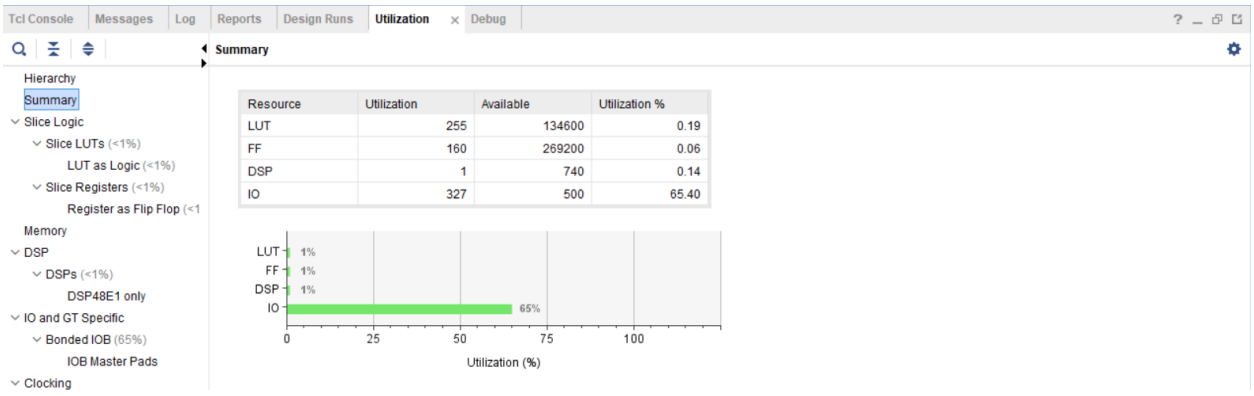


7.2 Message tab

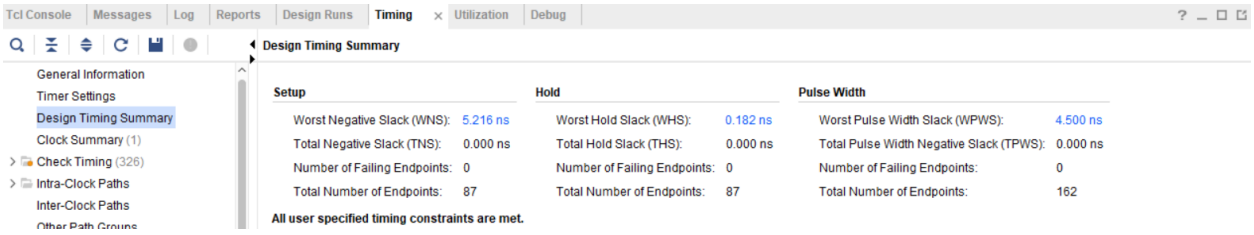


7.3 Utilization report



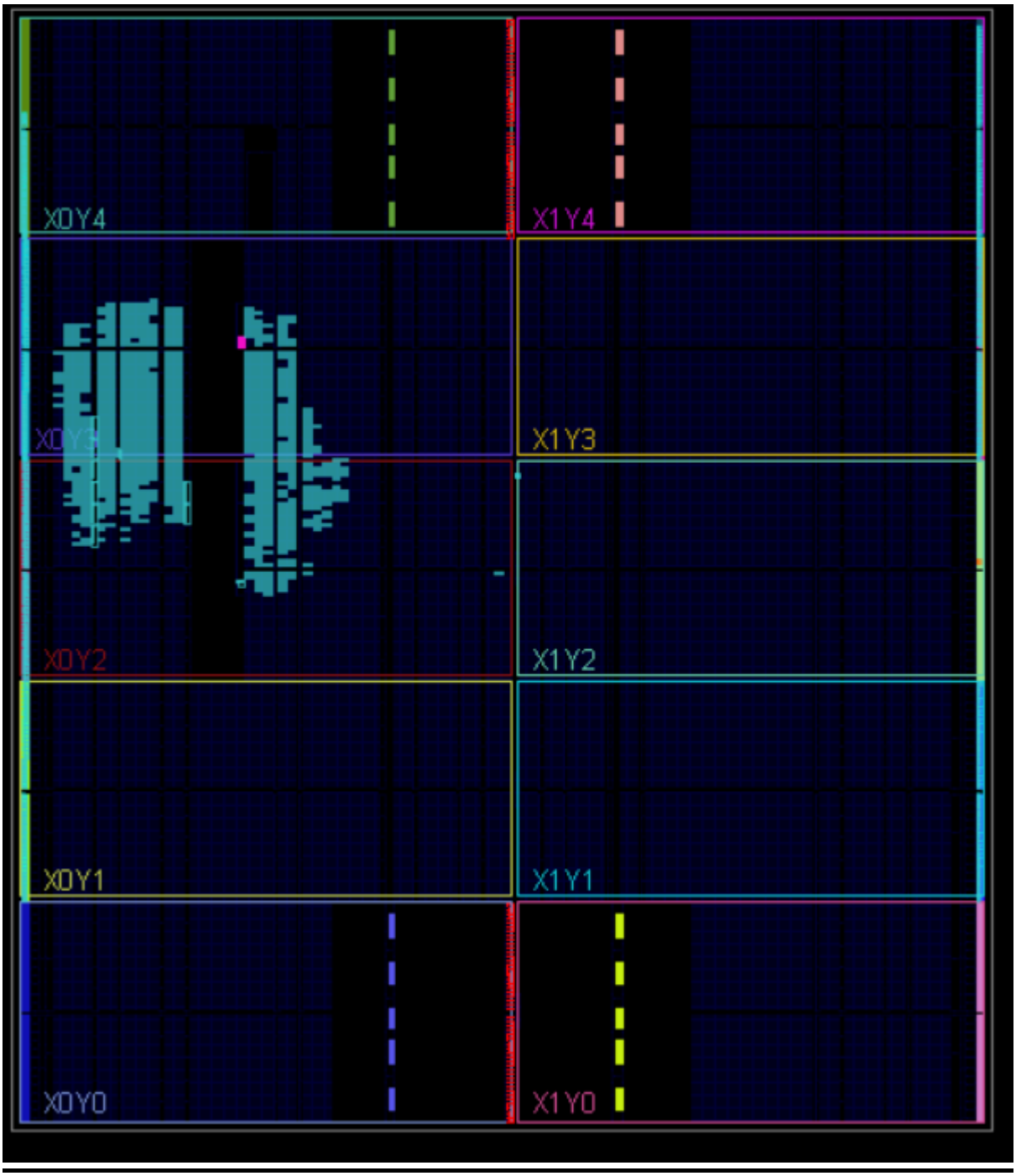


7.4 Timing report summary

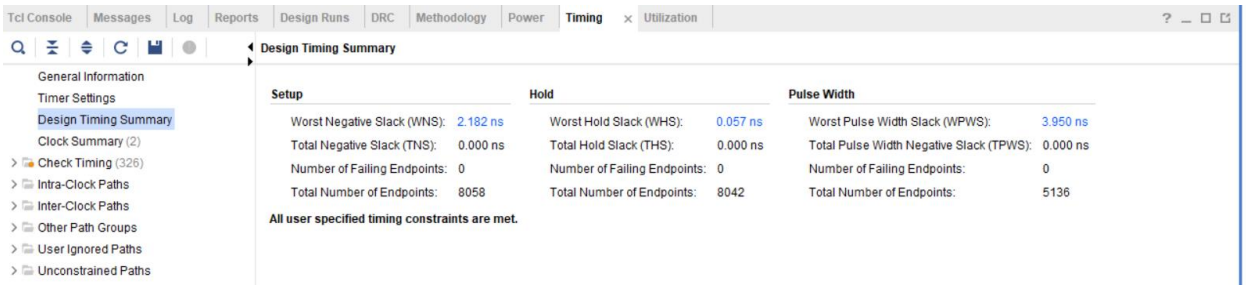


8. Implementation

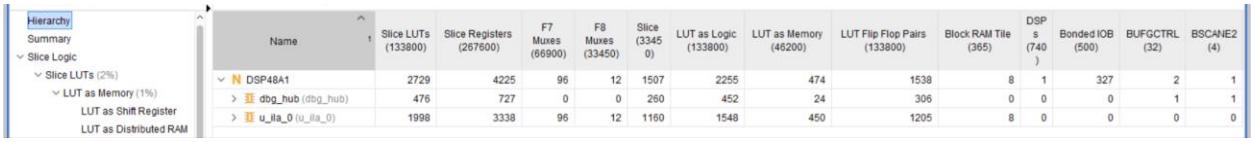
8.1 Device

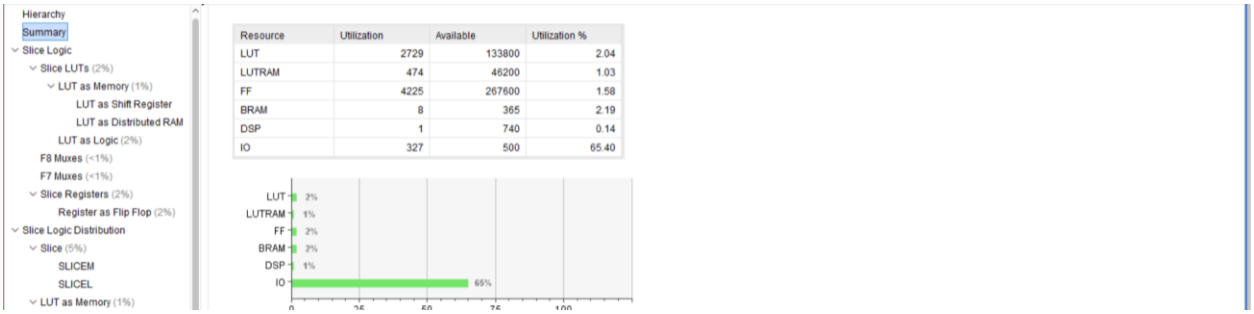


8.2 Timing report summary



8.3 Utilization report





8.4 Message tab