Spartan-6 DSP48A1 Project

Verilog Implementation and Design Flow

Mahmoud Ghamry

AUG.2024

Contents

1. **Introduction**

The DSP48A1 block, an evolution of the DSP48A slice found in Extended Spartan-3A FPGAs, stands out as a powerful component in digital signal processing. This versatile slice supports numerous DSP algorithms while utilizing minimal general-purpose FPGA logic, resulting in a design that is low power, high performance, and efficiently utilizes the device.

At its core, the DSP48A1 features an 18-bit input pre-adder, an 18x18-bit two's complement multiplier, and a 48-bit sign-extended adder/subtractor/accumulator. This combination is integral to a wide range of DSP applications. However, a closer examination reveals a wealth of subtle features that enhance its utility, versatility, and speed.

Programmable pipelining of input operands, intermediate products, and accumulator outputs significantly boosts throughput. The 48-bit internal bus facilitates virtually unlimited aggregation of DSP slices, making it an incredibly scalable solution.

A standout feature of the DSP48A1 is its ability to cascade results from one slice to the next without relying on general fabric routing. This capability provides high-performance, low-power post-addition for DSP filter functions of any tap length. Additionally, the cascading of input streams from slice to slice, facilitated by the C input port, allows for the creation of complex 3-input mathematical functions, such as 3-input addition and 2-input multiplication with a single addition.

Moreover, the D input port enables the use of a second argument with the pre-adder, reducing the utilization of DSP48A1 slices in symmetric filters. These features collectively make the DSP48A1 an exceptional arithmetic building block, enhancing the performance and efficiency of digital signal processing tasks.

2. Architecture highlights

Two-input pre-adder/subtracter for efficient implementation of symmetric filters

• 18-bit x 18-bit, two's-complement multiplier with a full-precision 36-bit result, sign extended to 48 bits.

• Two-input, flexible 48-bit post-adder/subtracter with optional registered accumulation feedback.

• Dynamic user-controlled operating modes to adapt DSP48A1 slice functions from clock cycle to clock cycle.

• Cascading 18-bit B bus, supporting input sample propagation.

• Cascading 48-bit P bus, supporting output propagation of partial results.

• Advanced carry management (cascadable, register capable, and routable to the user logic).

• Direct 36-bit multiplier output to the user logic.

• Performance enhancing pipeline options for control and data signals are selectable by configuration bits.

• Input port *C* typically used for multiply-add operation, large two-operand addition, or flexible rounding mode.

• Separate reset and clock enable for control and data registers.

• I/O registers, ensuring maximum clock performance and highest possible sample rates with no area cost.

3. Design Considerations

3.1 DSP48A1 Slice Primitive

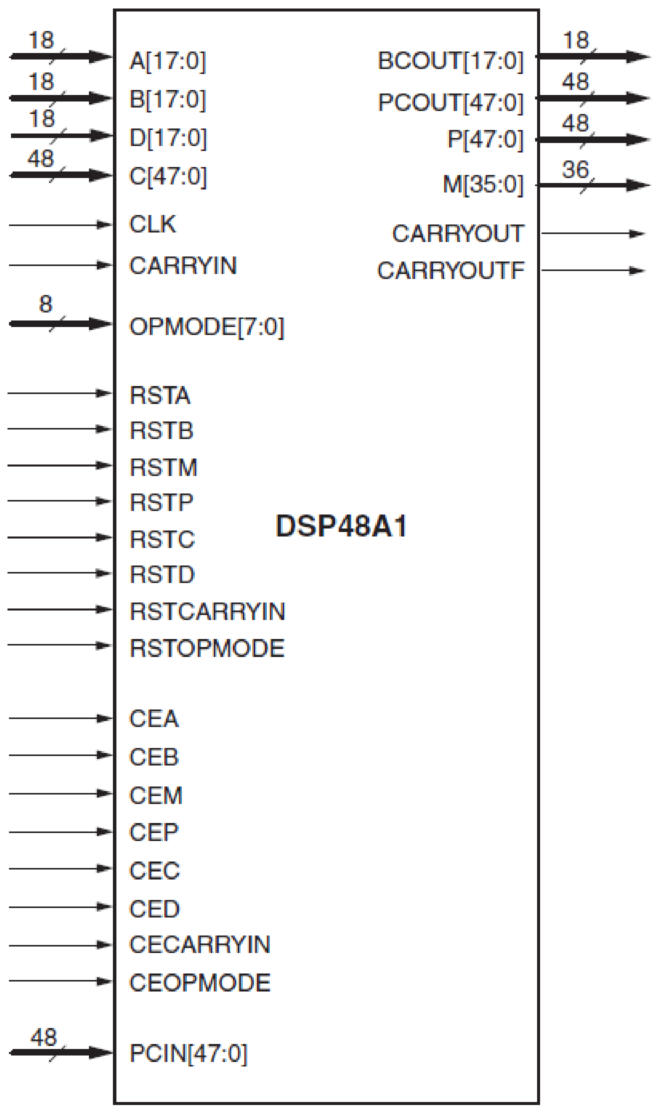

Figure 3-1: **DSP48A1 Slice Primitive**

3.2 Simplified DSP48A1 Slice

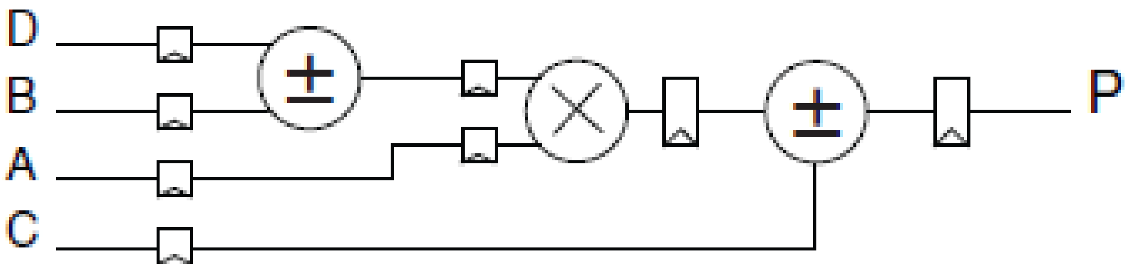

Figure 3-2: Simplified DSP48A1 Slice with Pre-Adder

3.2 DSP48A1 Slice in Detail

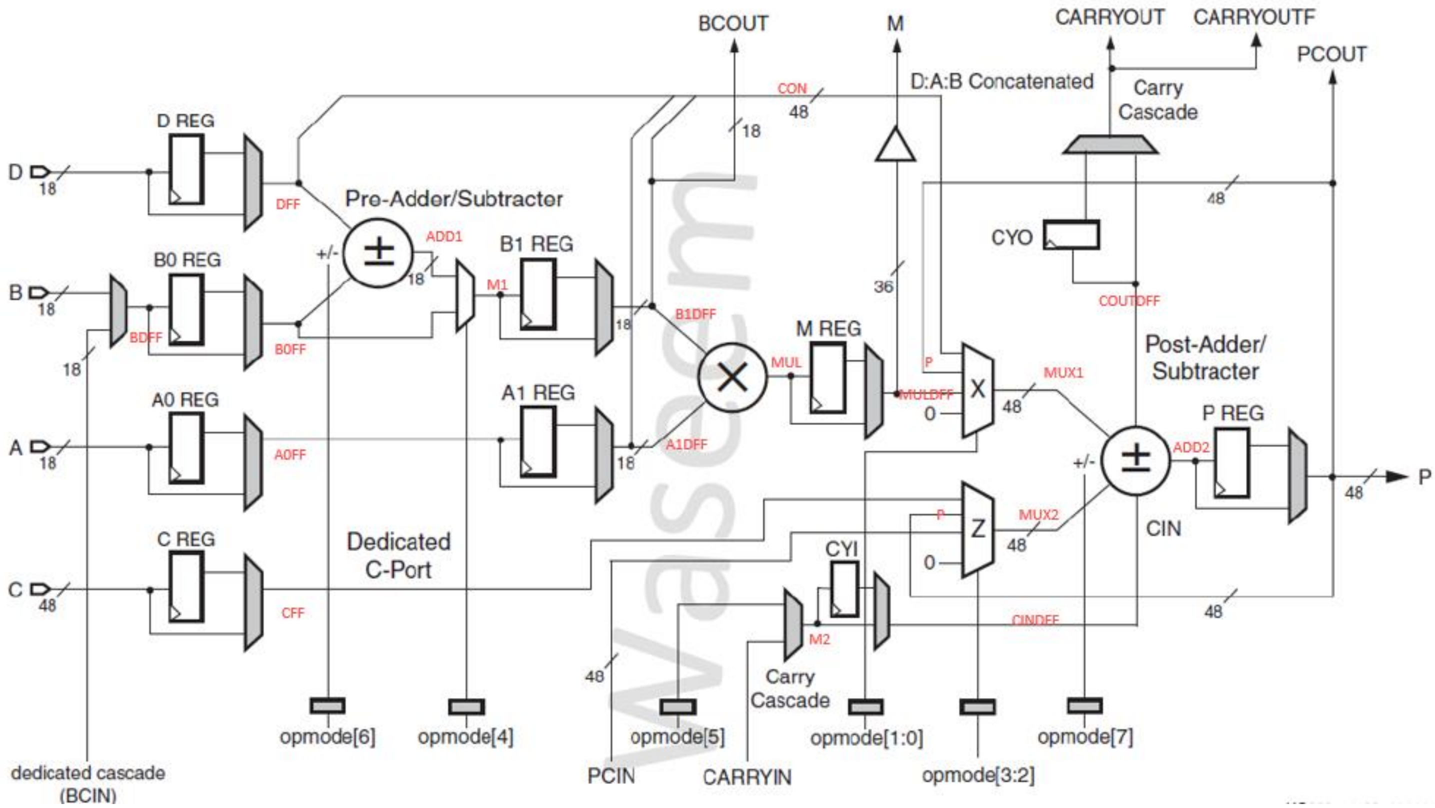

Figure 3-3: DSP48A1 Slice in Detail

4.CODING

4.1 MAIN CODE

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

            );
    //----------------PARAMETERS SECTION -------------
    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 SECTION -------------
    input [17:0]A,B,D,BCIN;
    input [47:0]C,PCIN;
    input [7:0]OPMODE;
    input CLK,CARRYIN,RSTA,RSTB,RSTM,RSTP,RSTC,RSTD,RSTCARRYIN,
          RSTOPMODE,CEA,CEB,CEM,CEP,CEC,CED,CECARRYIN,CEOPMODE;
```

Figure 4-1: main code

```
    //---------------OUTPUTS SECTION -------------
    output [17:0]BCOUT;
    output [47:0]PCOUT,P;
    output [35:0]M;
    output CARRYOUT , CARRYOUTF ;
    //---------------WIRE SECTION -------------
    wire M2,CINDFF,COUTDFF;
    wire [7:0]opmode;
    wire [17:0]DFF,BDFF,B0DFF,ADD1,M1,B1DFF,A0DFF,A1DFF;
    wire [35:0]MUL,MULDFF;
    wire [47:0]CON,MUX_1,MUX_2,CFF,ADD2,z;
    //---------------REG WITH MUX SECTION -------------
    REG_MUX #(8,RSTTYPE,OPMODEREG)OPMODE_OUT(OPMODE,opmode,CLK,RSTOPMODE,CEOPMODE);
    REG_MUX #(18,RSTTYPE,DREG)D_REG(D,DFF,CLK,RSTD,CED);//D,OUT,CLK,RST,ENABLE
    REG_MUX #(18,RSTTYPE,B0REG)B_REG0(BDFF,B0DFF,CLK,RSTB,CEB);
    REG_MUX #(18,RSTTYPE,B1REG)B_REG1(M1,B1DFF,CLK,RSTB,CEB);
    REG_MUX #(18,RSTTYPE,A0REG)A_REG0(A,A0DFF,CLK,RSTA,CEA);
    REG_MUX #(18,RSTTYPE,A1REG)A_REG1(A0DFF,A1DFF,CLK,RSTA,CEA);
    REG_MUX #(48,RSTTYPE,CREG)C_REG(C,CFF,CLK,RSTC,CEC);
    REG_MUX #(36,RSTTYPE,MREG)OUT_MULTIPLY(MUL,MULDFF,CLK,RSTM,CEM);
    REG_MUX #(1,RSTTYPE,CARRYINREG)CYI(M2,CINDFF,CLK,RSTCARRYIN,CECARRYIN);
    REG_MUX #(1,RSTTYPE,CARRYOUTREG)CYO(COUTDFF,CARRYOUT,CLK,RSTCARRYIN,CECARRYIN);
    REG_MUX #(48,RSTTYPE,PREG)P_REG(ADD2,P,CLK,RSTP,CEP);
```

Figure 4-2: main code

```
    //---------------ASSIGN SECTION -------------
    assign BDFF = ( B_INPUT == "DIRECT" ) ? B : BCIN ;
    assign ADD1 = (opmode[6] == 1) ?    DFF - B0DFF : DFF + B0DFF ;
    assign M1 = (opmode[4] == 1) ? B0DFF : ADD1 ;
    assign MUL = B1DFF * A1DFF ;
    assign BCOUT = B1DFF ;
    assign CON = {DFF[11:0],A1DFF,B1DFF};
    assign M = MULDFF ;
    assign M2 = (CARRYINSEL == "OPMODE5" ) ? opmode[5] : CARRYIN  ;
    assign {COUTDFF,ADD2} = (opmode[7] == 0 ) ? MUX_1 + MUX_2 + CINDFF : MUX_1 - MUX_2 - CINDFF ;
    assign z = {12'b0000_0000_0000,MULDFF} ;
    assign CARRYOUTF = CARRYOUT ;
    assign PCOUT = P ;
    //--------------- MUX SECTION ------------
    mux4 #(48)MUX1(CON,P,z,48'b0,opmode[1:0],MUX_1);
    mux4 #(48)MUX2(CFF,P,PCIN,48'b0,opmode[3:2],MUX_2);
endmodule : DSP
```

Figure 4-3: main code

```
module REG_MUX (D,OUT,CLK,RST,ENABLE    );

    parameter SIZE = 18 ;
    parameter RSTTYPE = "SYNC";
    parameter SEL = 1 ;

    input [ SIZE - 1 : 0 ]D;
    input CLK,RST,ENABLE;

    output reg [SIZE-1:0]OUT;

    generate
        if(SEL == 1)begin
        if(RSTTYPE == "ASYNC") begin
            always @(posedge CLK or posedge RST) begin
                if (RST) begin
                    OUT <= 0 ;
                end
                else if(ENABLE) begin
                    OUT <= D ;
                end
            end
        end
```

Figure 4-4: instantiation code

```
        else if(RSTTYPE == "SYNC") begin
            always @(posedge CLK ) begin
                if (RST) begin
                    OUT <= 0 ;
                end
                else if(ENABLE) begin
                    OUT <= D ;
                end
            end
        end
        end
        else if(SEL == 0 )begin
        always @(*) begin
            OUT = D;

            end
        end
    endgenerate
endmodule : REG_MUX
```

Figure 4-5: instantiation code

```
module mux4 (
    A,B,C,D,S,OUT

);
    parameter size = 1 ;
    input [size - 1 : 0 ]A,B,C,D;
    input [ 1 : 0]S;
    output reg [ size - 1 : 0 ]OUT;
    always @(*) begin
        case(S)

        2'b00: OUT = A ;
        2'b01: OUT = B ;
        2'b10: OUT = C ;
        2'b11: OUT = D ;

        endcase
     end
endmodule : mux4
```

Figure 4-6: instantiation code

4.2 TEST BENCH CODE

```
module DSP_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      = "ASYNC";

    reg [17:0]A,B,D,BCIN;
    reg [47:0]C,PCIN;
    reg [7:0]OPMODE;
    reg CLK,CARRYIN,RSTA,RSTB,RSTM,RSTP,RSTC,RSTD,RSTCARRYIN,
        RSTOPMODE,CEA,CEB,CEM,CEP,CEC,CED,CECARRYIN,CEOPMODE;

    wire [17:0]BCOUT;
    wire [47:0]PCOUT,P;
    wire [35:0]M;
    wire CARRYOUT , CARRYOUTF ;

```

*Figure 4-7:*TEST BENCH code

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

```

*Figure 4-8:*TEST BENCH code

```
    initial begin
    CLK=0;

    forever
    #1 CLK=~CLK;
    end

```

*Figure 4-9:*TEST BENCH code

```
    initial begin

        A=0;B=0;D=0;BCIN=0;C=0;PCIN=0;OPMODE=0;
        CARRYIN=0;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;

        @(negedge CLK);

        // Test case 1: P = a
        A = 2; B = 3; D = 6; C = 4; PCIN = 2; BCIN = 1; OPMODE = 8'b00010010;
        @(negedge CLK); @(negedge CLK); @(negedge CLK);@(negedge CLK);

        // Test case 2: P = 4
        A = 2; B = 2; D = 2; C = 2; PCIN = 2; BCIN = 0; OPMODE = 8'b10010010;
        @(negedge CLK); @(negedge CLK);

        // Test case 3: P = 2
        A = 2; B = 4; D = 2; C = 2; PCIN = 2; BCIN = 0; OPMODE = 8'b10011010;
        @(negedge CLK); @(negedge CLK); @(negedge CLK);@(negedge CLK);

        // Test case 4: P = 6
        A = 2; B = 4; D = 3; C = 2; PCIN = 2; BCIN = 0; OPMODE = 8'b10011010;
        @(negedge CLK); @(negedge CLK);

        // Test case 5:
        A = 2; B = 4; D = 2; C = 2; PCIN = 2; BCIN = 0; OPMODE = 8'b00001010;

        @(negedge CLK); @(negedge CLK); @(negedge CLK);@(negedge CLK);

```

*Figure 4-10:*TEST BENCH code

```

        // Test case 6:
        A = 2; B = 24; D = 2; C = 2; PCIN = 12; BCIN = 12; OPMODE = 8'b11010010;
        @(negedge CLK);@(negedge CLK);

        // Test case 7:
        A = 2; B = 4; D = 9; C = 5; PCIN = 2; BCIN = 1; OPMODE = 8'b10010010;
        @(negedge CLK);@(negedge CLK); @(negedge CLK);@(negedge CLK);

        // Test case 8:
        A = 2; B = 4; D = 2; C = 10; PCIN = 5; BCIN = 0; OPMODE = 8'b01010010;

        @(negedge CLK); @(negedge CLK); @(negedge CLK);@(negedge CLK);

        // Test case 9:
        A = 2; B = 4; D = 5; C = 2; PCIN = 8; BCIN = 0; OPMODE = 8'b10010010;
        @(negedge CLK);@(negedge CLK); @(negedge CLK); @(negedge CLK);

        // Test case 10:
        A = 2; B = 3; D = 6; C = 4; PCIN = 2; BCIN = 8; OPMODE = 8'b00111101;
        @(negedge CLK);@(negedge CLK); @(negedge CLK); @(negedge CLK);
        A = 2; B = 3; D = 6; C = 4; PCIN = 2; BCIN = 8; OPMODE = 8'b00111101;

        $stop;
    end
endmodule : DSP_TB
```

*Figure 4-11:*TEST BENCH code

5.RESULT

5.1 Result from Questa Sim -64 2021

5.1.1 Wave form

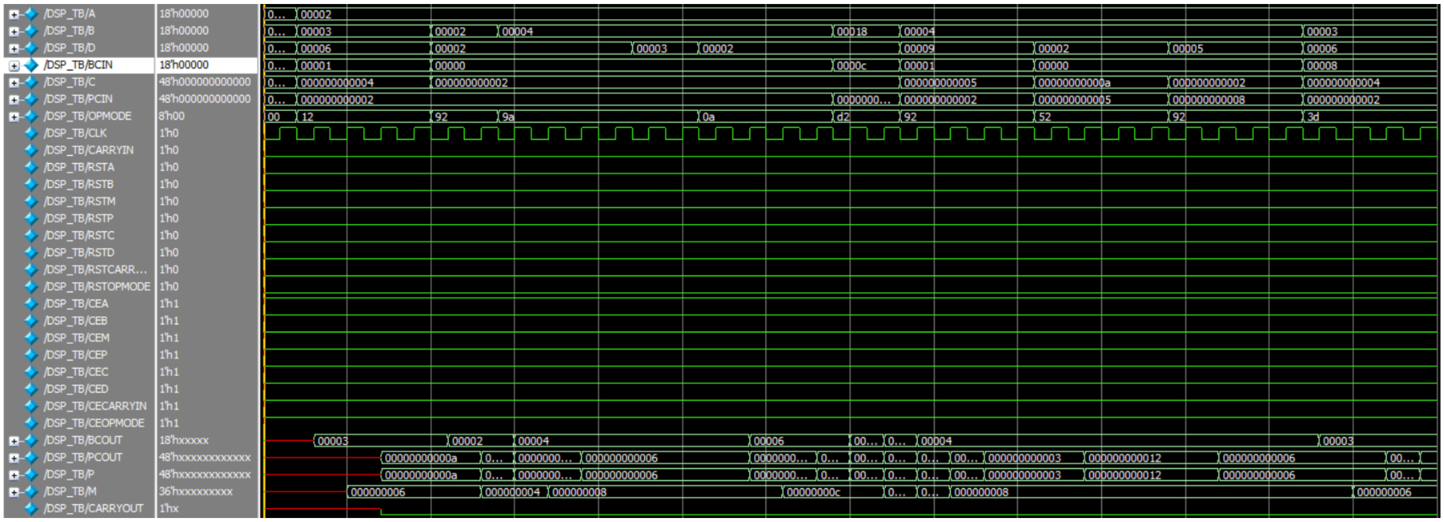

*Figure 5-1:*Wave form

5.2 Schematic using VIVADO 2018

5.2.1 Elaborated Design

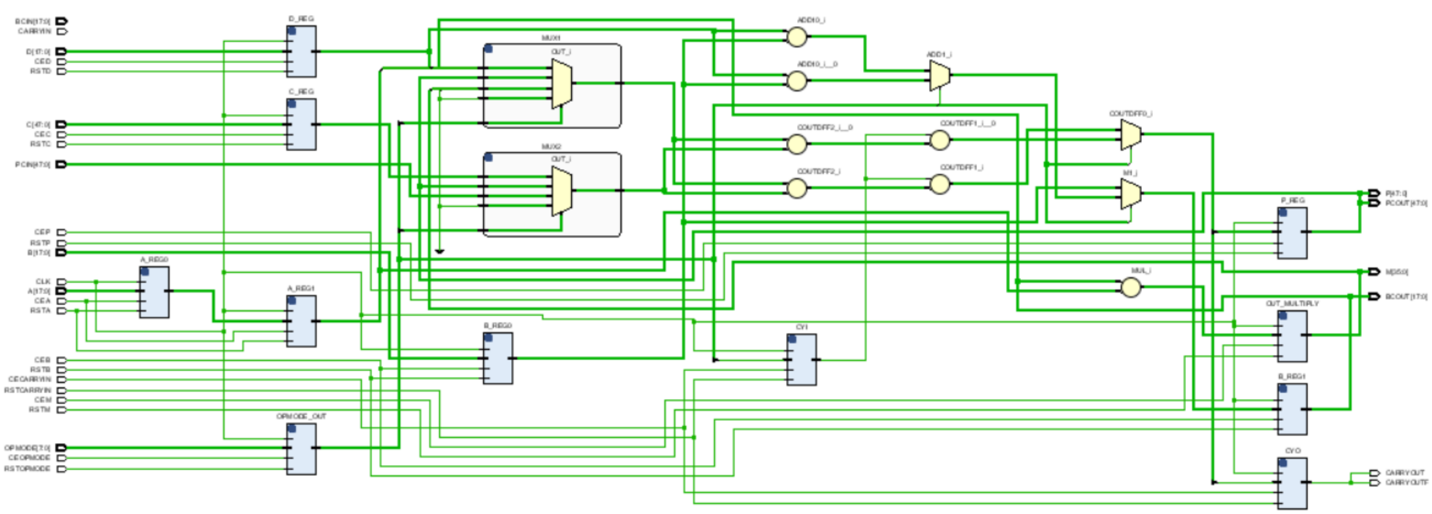

*Figure 5-2:*Elaborated Design

5.2.2 Synthesis Design

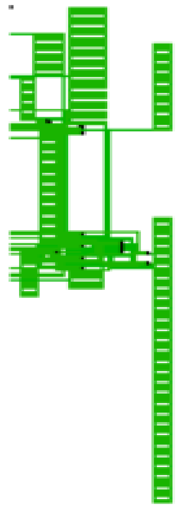

Figure 5-3: Synthesis Design

5.2.3 Implementation Design

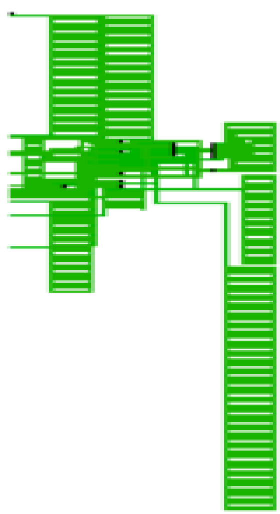

Figure 5-4: Implementation Design

6. REPORTS

6.1 Utilization

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)
DSP	273	179	116	273	50	1	327	1
A_REG1 (REG_MUX__...	0	18	10	0	0	0	0	0
B_REG1 (REG_MUX_...	0	36	13	0	0	0	0	0
C_REG (REG_MUX__...	0	48	17	0	0	0	0	0
CYI (REG_MUX__para...	1	1	1	1	1	0	0	0
CYO (REG_MUX__par...	0	2	2	0	0	0	0	0
D_REG (REG_MUX__...	0	18	10	0	0	0	0	0
OPMODE_OUT (REG_...	272	8	83	272	0	0	0	0
OUT_MULTIPLY (REG...	0	0	0	0	0	1	0	0
P_REG (REG_MUX__...	0	48	12	0	0	0	0	0

Figure 6-1: Utilization Report

6.2 TIME

Setup

Worst Negative Slack (WNS):	4.211 ns
Total Negative Slack (TNS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	125

Hold

Worst Hold Slack (WHS):	0.263 ns
Total Hold Slack (THS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	125

Pulse Width

Worst Pulse Width Slack (WPWS):	4.500 ns
Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	181

All user specified timing constraints are met.

Figure 6-2: Time Report

7. ADDTION FILES

7.1 DO FILE

```
vlib work
vlog DSP.v DSP_TB.v mux4.v REG_MUX.V
vsim -voptargs=+acc work.DSP_TB
add wave *
run -all
#quit -sim
```

Figure 7-1: DO file

7.2 Constraints

```
## 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]
```

Figure 7-2: Constraints_basys

8. Implementation DEVICE IN FPGA

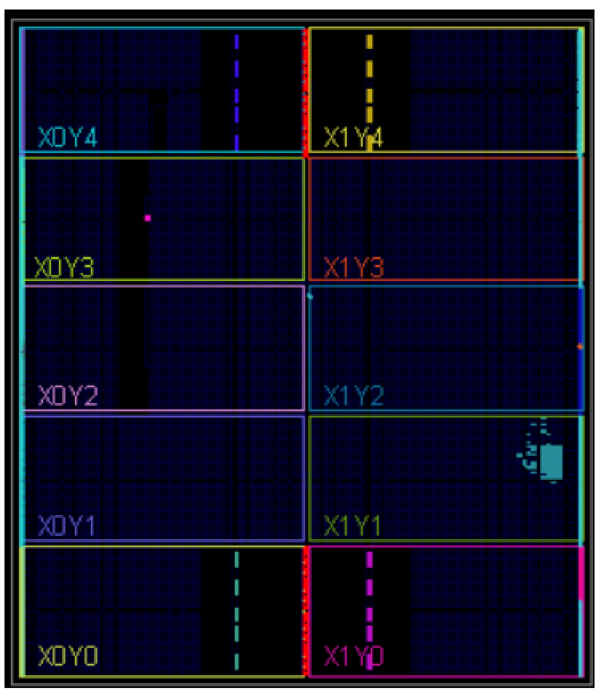

Figure 8: Implementation