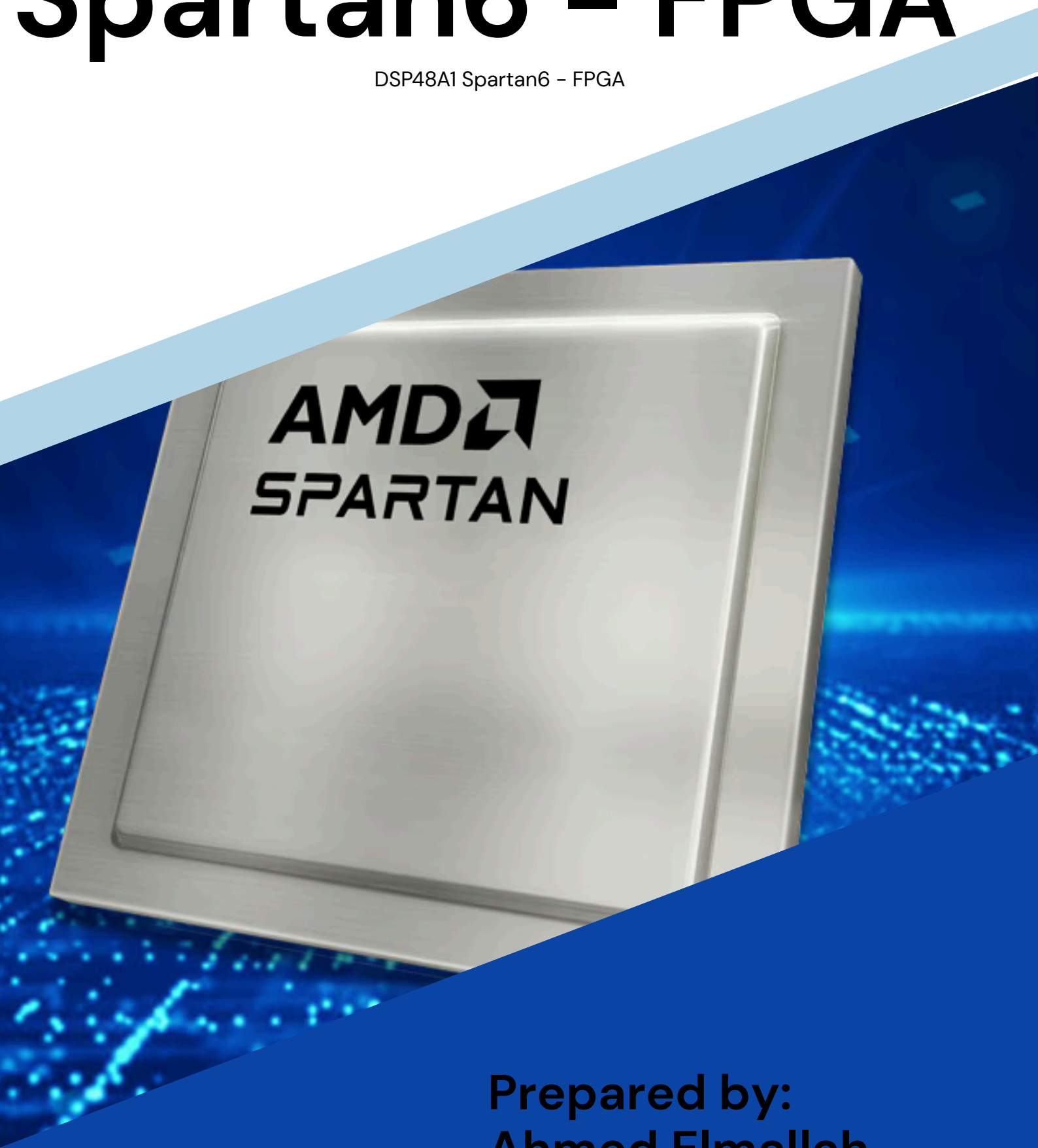
DSP48A1 Spartan6 – FPGA

DSP48A1 Spartan6 - FPGA



Prepared by:
Ahmed Elmallah

Table of Contents

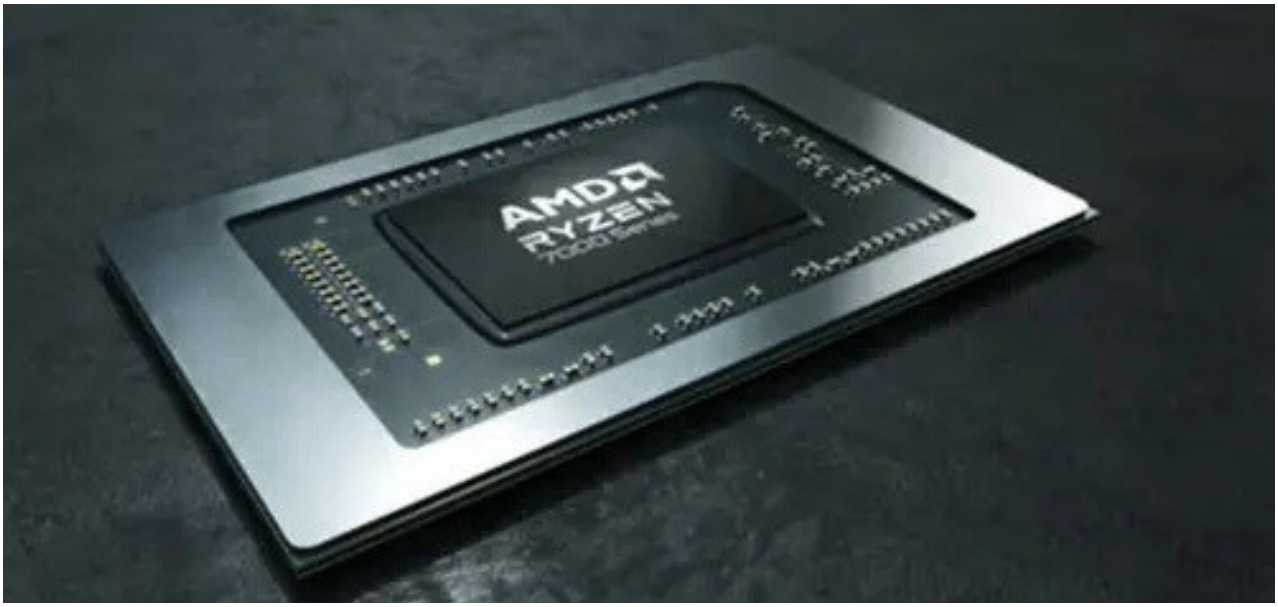
Introduction 03
DSP Architecture 04
RTL Code 05
Testbench Code 08
Do File 10
Questa 11
QuestaLint 12
Constraints File 12
Vivado 12
Implemenation; 16

Introduction

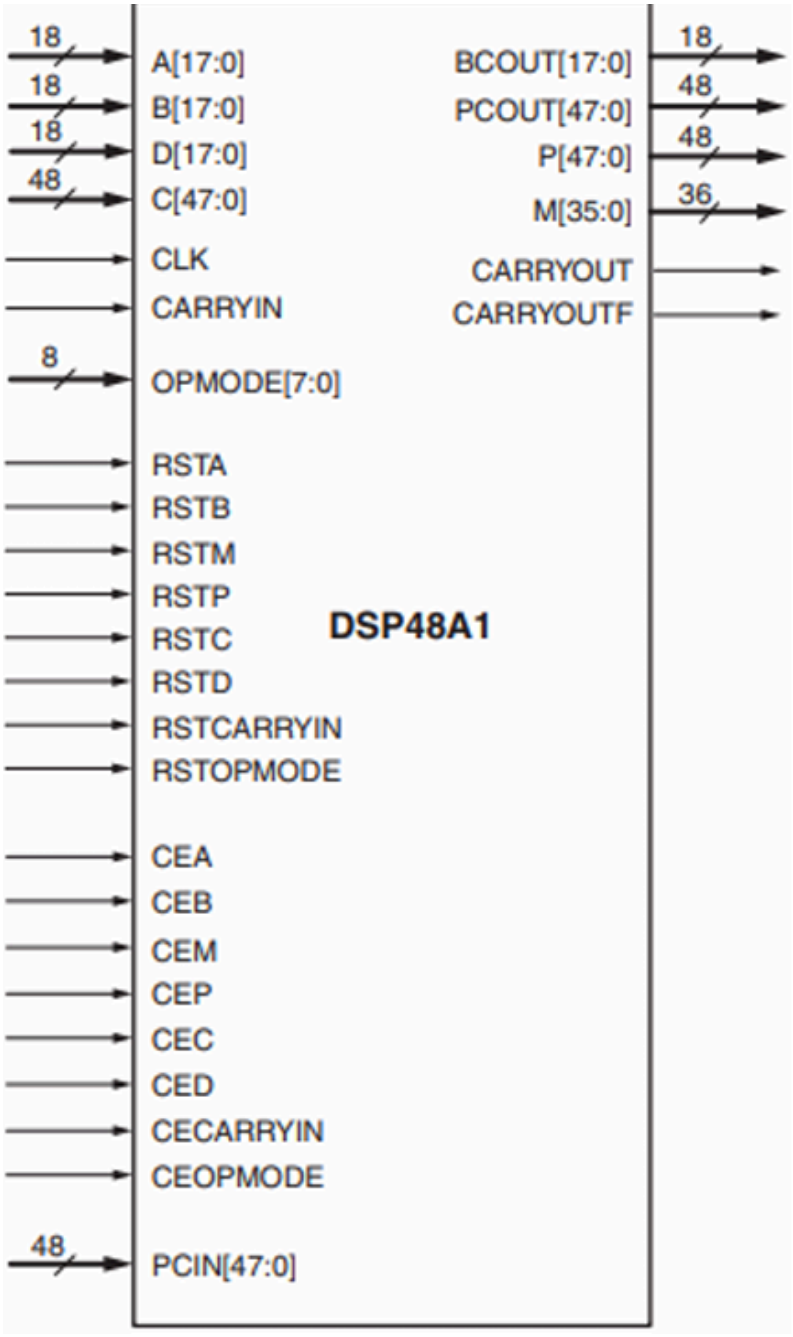
This project focuses on the design, implementation, and verification of the DSP48A1 slice — a critical arithmetic building block within Xilinx Spartan-6 FPGAs. The DSP48A1 slice is widely used in digital signal processing (DSP), image processing, and high-performance computing applications due to its ability to efficiently perform multiply-accumulate, addition, subtraction, and logical operations.
In this work, a complete hardware model of the DSP48A1 slice was developed using VHDL/Verilog, including all supporting components and a comprehensive verification flow. The design effort covers every stage from RTL development to FPGA implementation:

- A configurable model module for instantiating DSP48A1 functionality.
- A full RTL implementation of the DSP48A1 slice in VHDL/Verilog.
- A robust testbench to validate functionality across multiple scenarios.
- Static analysis using QuestaLint and automated simulation on QuestaSim.
- FPGA synthesis and timing analysis on Xilinx Vivado, including constraints and utilization reports.

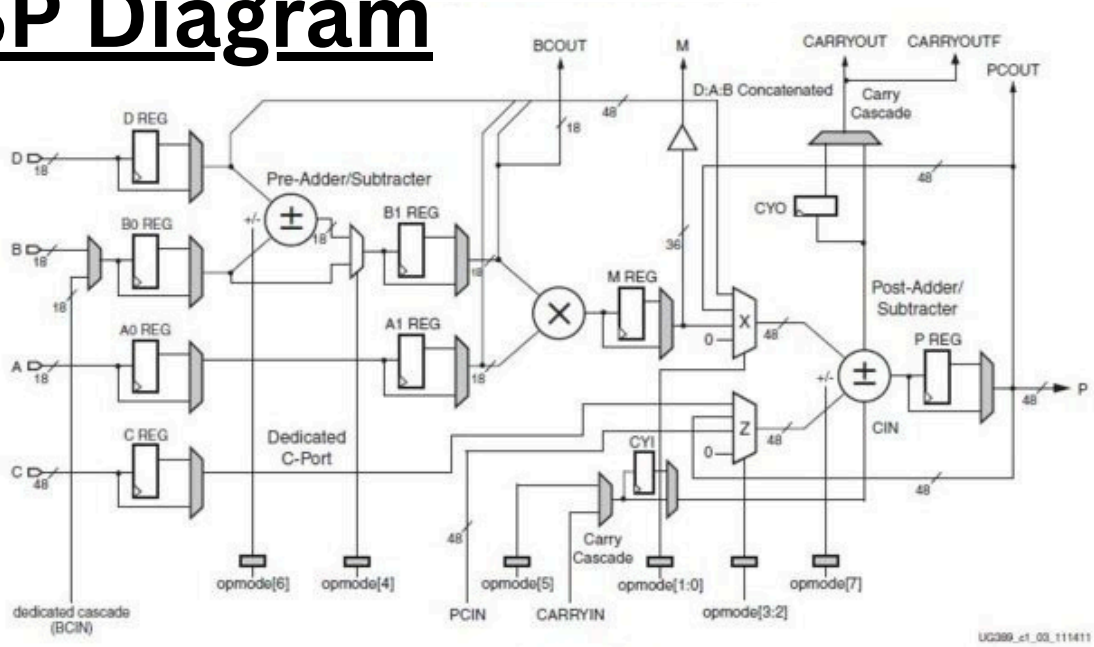
This end-to-end flow demonstrates how a complex FPGA building block can be designed, validated, and optimized for hardware, offering insight into both the architectural features of the DSP48A1 slice and best practices in FPGA development.



DSP Architecture



DSP Diagram



RTL

1.PIPELINE module

```
module MUX (in,out,en,clk,rst);
    // Parameters
    parameter RSTTYPE = "SYNC";
    parameter sel     = 1;
    parameter size    = 18;

    // Ports
    input  [size-1:0] in;
    input   rst, clk, en;
    output [size-1:0] out;

    // Internal register
    reg [size-1:0] in_reg;

    // Registering logic
    generate
        if (RSTTYPE == "SYNC") begin
            // Synchronous reset
            always @(posedge clk) begin
                if (rst)
                    in_reg <= 0;
                else if (en)
                    in_reg <= in;
            end
        end else begin
            // Asynchronous reset
            always @(posedge clk or posedge rst) begin
                if (rst)
                    in_reg <= 0;
                else if (en)
                    in_reg <= in;
            end
        end
    endgenerate

    // Output selection (MUX)
    assign out =(sel)? in_reg:in;
endmodule // MUX

```

2.DSP48A1 Code

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

// Parameters
parameter A0REG = 0;
parameter A1REG = 1;
parameter B0REG = 0;
parameter B1REG = 1;
parameter CREG = 1, DREG = 1, MREG = 1, PREG = 1, CARRTINREG = 1, CARRYOUTREG = 1, OPMODEREG = 1;
parameter CARRYINSEL = "OPMODE5";
parameter B_INPUT = "DIRECT";
parameter RSTTYPE = "SYNC";

// Ports
input  [17:0] A, B, BCIN;
input  [47:0] C;
output CARRYOUT, CARRYOUTF;
input CARRYIN;
input  [17:0] D;
output [35:0] M;
output [47:0] P;
input clk;
input [7:0] OPMODE;
input CEA, CEB, CEC, CECARRYIN, CED, CEM, CEOPMODE;
input CEP, RSTA, RSTB, RSTC, RSTCARRYIN, RSTD, RSTM, RSTOPMODE, RSTP;
output [17:0] BCOUT;
input  [47:0] PCIN;
output [47:0]  PCOUT;

// internal signals
wire [7:0] OPMODE_REG;
wire [17:0] B_cascade;
wire [17:0] a0_reg, b0_reg;
wire [47:0] c_reg;
wire [17:0] d_reg;
wire [17:0] a1_reg, b1_reg;
wire [17:0] pre_adder_sub;
wire [17:0] mux1;
wire [35:0] mul_out;
wire [35:0] mul_reg;
wire carry_cascade;
wire carryin_reg;
reg [47:0] x_reg, z_reg;
wire [47:0] post_add_sub;

```

```
// Registers instantiation


MUX #(.RSTTYPE(RSTTYPE), .sel(OPMODEREG), .size(8)) opmode_reg (.in(OPMODE), .out(OPMODE_REG), .en(CEOPMODE), .clk(clk), .rst(RSTOPMODE));
assign B_cascade = (B_INPUT == "DIRECT") ? B : (B_INPUT == "DIRECT") ? BCIN : 17'b0;

MUX #(.RSTTYPE(RSTTYPE), .sel(A0REG), .size(18)) A0_REG (.in(A), .out(a0_reg), .en(CEA), .clk(clk), .rst(RSTA));
MUX #(.RSTTYPE(RSTTYPE), .sel(B0REG), .size(18)) B0_REG (.in(B_cascade), .out(b0_reg), .en(CEB), .clk(clk), .rst(RSTB));
MUX #(.RSTTYPE(RSTTYPE), .sel(CREG),  .size(48)) C_REG  (.in(C), .out(c_reg), .en(CEC), .clk(clk), .rst(RSTC));
MUX #(.RSTTYPE(RSTTYPE), .sel(DREG),  .size(18)) D_REG  (.in(D), .out(d_reg), .en(CED), .clk(clk), .rst(RSTD));

MUX #(.RSTTYPE(RSTTYPE), .sel(A1REG), .size(18)) A1_REG (.in(a0_reg), .out(a1_reg), .en(CEA), .clk(clk), .rst(RSTA));

// Pre-adder logic
assign pre_adder_sub = (OPMODE_REG[6] == 1'b0) ? b0_reg + d_reg : b0_reg - d_reg;
assign mux1 = (OPMODE_REG[4] == 1'b0) ? b0_reg : pre_adder_sub;

// B1 register
MUX #(.RSTTYPE(RSTTYPE), .sel(B1REG), .size(18)) B1_REG (.in(mux1), .out(b1_reg), .en(CEB), .clk(clk), .rst(RSTB));
assign BCOUT = b1_reg;

// Multiplier logic
assign mul_out = a1_reg * b1_reg;
MUX #(.RSTTYPE(RSTTYPE), .sel(MREG), .size(36)) Mul_reg (.in(mul_out), .out(mul_reg), .en(CEM), .clk(clk), .rst(RSTM));
assign M = mul_reg;

// Carryin logic
assign carry_cascade = (CARRYINSEL == "OPMODE5") ? OPMODE[5] : (CARRYINSEL == "OPMODE5") ? CARRYIN : 1'b0;
MUX #(.RSTTYPE(RSTTYPE), .sel(CARRTINREG), .size(1)) CARRYIN_REG (.in(carry_cascade), .out(carryin_reg), .en(CECARRYIN), .clk(clk), .rst(RSTCARRYIN));

// X mux
always @(*) begin
    case (OPMODE_REG[1:0])
        2'b00: x_reg = 48'b0;
        2'b01: x_reg = {12'b0, mul_reg};
        2'b10: x_reg = P;
        2'b11: x_reg = {d_reg[11:0], a1_reg, b1_reg};
        default: x_reg = 48'b0;
    endcase
end

// Z mux
always @(*) begin
    case (OPMODE_REG[3:2])
        2'b00: z_reg = 48'b0;
        2'b01: z_reg = PCIN;
        2'b10: z_reg = P;
        2'b11: z_reg = c_reg;
        default: z_reg = 48'b0;
    endcase
end

// Add/Sub logic
assign {carryout_in, post_add_sub} = (OPMODE_REG[7] == 1'b0) ? z_reg + x_reg + carryin_reg : z_reg - (x_reg + carryin_reg);

// Carryout logic
MUX #(.RSTTYPE(RSTTYPE), .sel(CARRYOUTREG), .size(1)) CARRYOUT_REG (.in(carryout_in), .out(CARRYOUT), .en(CECARRYIN), .clk(clk), .rst(RSTCARRYIN));
assign CARRYOUTF = CARRYOUT;

// P register
MUX #(.RSTTYPE(RSTTYPE), .sel(PREG), .size(48)) P_REG (.in(post_add_sub), .out(P), .en(CEP), .clk(clk), .rst(RSTP));
assign PCOUT = P;




endmodule // DSP

```

Testbench Code

```
module DSP_tb ();
    // Parameters
    parameter A0REG = 0;
    parameter A1REG = 1;
    parameter B0REG = 0;
    parameter B1REG = 1;
    parameter CREG = 1, DREG = 1, MREG = 1, PREG = 1, CARRTINREG = 1, CARRYOUTREG = 1, OPMODEREG = 1;
    parameter CARRYINSEL = "OPMODE5";
    parameter B_INPUT = "DIRECT";
    parameter RSTTYPE = "SYNC";

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

    // Instantiate the Unit Under Test (UUT)
    DSP #(
        .A0REG(A0REG),
        .A1REG(A1REG),
        .B0REG(B0REG),
        .B1REG(B1REG),
        .CREG(CREG),
        .DREG(DREG),
        .MREG(MREG),
        .PREG(PREG),
        .CARRTINREG(CARRTINREG),
        .CARRYOUTREG(CARRYOUTREG),
        .OPMODEREG(OPMODEREG),
        .CARRYINSEL(CARRYINSEL),
        .B_INPUT(B_INPUT),
        .RSTTYPE(RSTTYPE)
    ) uut (.*);

    // Clock generation
    initial begin
        clk = 0;
        forever #5 clk = ~clk; // Toggle clock every 5 time units
    end

```

```
// ++++++++++++++++++++++++++++ RST TEST ++++++++++++++++++++++++++++
    initial begin
        // active reset
        RSTA = 1; RSTB = 1; RSTC = 1; RSTCARRYIN = 1; RSTD = 1; RSTM = 1; RSTOPMODE = 1; RSTP = 1;
        // Initialize control signals to 1
        CEA = 1; CEB = 1; CEC = 1; CECARRYIN = 1; CED = 1; CEM = 1; CEOPMODE = 1; CEP = 1;
        // Initialize inputs to 1
        A = 1; B = 1; C = 1; BCIN = 1; CARRYIN = 1; D = 1; OPMODE = 1; PCIN = 1;
        @(negedge clk);
        // deactivate reset
        RSTA = 0; RSTB = 0; RSTC = 0; RSTCARRYIN = 0; RSTD = 0; RSTM = 0; RSTOPMODE = 0; RSTP = 0;
        if (P !== 0 || M !== 0 || PCOUT !== 0 || BCOUT !== 0 || CARRYOUT !== 0 || CARRYOUTF !== 0) begin
            $display("Test for RST Failed");
            $stop;
        end

        // ++++++++++++++++++++++++ opmode [0] test ++++++++++++++++++++
        OPMODE = 8'b00000001;
        A = 18'd6; B = 18'd14; C = 48'd43; BCIN = 18'd34; CARRYIN = 1; D = 18'd12;
        repeat (4) @(negedge clk);
        if (P !== (A*B) || M !== (A*B) || PCOUT !== (A*B) || BCOUT !== B || CARRYOUT !== 0 || CARRYOUTF !== 0) beg
            $display("Test Failed for OPMODE[0] fail");
            $stop;
        end

        // ++++++++++++++++++++++ opmode [1] test ++++++++++++++++++++
        OPMODE = 8'b00000010;
        repeat (4) @(negedge clk);
        if (P !== P || M !== (A*B) || PCOUT !== P || BCOUT !== B || CARRYOUT !== 0 || CARRYOUTF !== 0) begin
            $display("Test Failed for OPMODE[1] fail");
            $stop;
        end

        // ++++++++++++++++++++++ opmode [2] test ++++++++++++++++++++
        OPMODE = 8'b00000100;
        repeat (4) @(negedge clk);
        if (P !== PCIN || M !== (A*B) || PCOUT !== PCIN || BCOUT !== B || CARRYOUT !== 0 || CARRYOUTF !== 0) begi
            $display("Test Failed for OPMODE[2] fail");
            $stop;
        end

        // ++++++++++++++++++++++ opmode [3] test++++++++++++++++++++++
        OPMODE = 8'b00001000;
        repeat (4) @(negedge clk);
        if (P !== P || M !== (A*B) || PCOUT !== P || BCOUT !== B || CARRYOUT !== 0 || CARRYOUTF !== 0) begin
            $display("Test Failed for OPMODE[3] fail");
            $stop;
        end

        // ++++++++++++++++++++++ opmode [4] test ++++++++++++++++++++++
        OPMODE = 8'b00010000;
        repeat (4) @(negedge clk);
        if (P !== 0 || M !== (B+D)*A || PCOUT !== 0 || BCOUT !== (B+D) || CARRYOUT !== 0 || CARRYOUTF !== 0) begi
            $display("Test Failed for OPMODE[4] fail");
            $stop;
        end

        // ++++++++++++++++++++++ opmode [5] test ++++++++++++++++++++++
        OPMODE = 8'b00100000;
        repeat (4) @(negedge clk);
        if (P !== 1 || M !== B*A || PCOUT !== 1 || BCOUT !== B || CARRYOUT !== 0 || CARRYOUTF !== 0) begin
            $display("Test Failed for OPMODE[5] fail");
            $stop;
        end

        // ++++++++++++++++++++++ opmode [6] test ++++++++++++++++++++++
        OPMODE = 8'b01000000;
        repeat (4) @(negedge clk);
        if (P !== 0 || M !== B*A || PCOUT !== 0 || BCOUT !== B || CARRYOUT !== 0 || CARRYOUTF !== 0) begin
            $display("Test Failed for OPMODE[6] fail");
            $stop;
        end

        // ++++++++++++++++++++++ opmode [7] test ++++++++++++++++++++++
        OPMODE = 8'b10000000;
        repeat (4) @(negedge clk);
        if (P !== 0 || M !== A*B || PCOUT !== 0 || BCOUT !== B || CARRYOUT !== 0 || CARRYOUTF !== 0) begin
            $display("Test Failed for OPMODE[7] fail");
            $stop;
        end

```

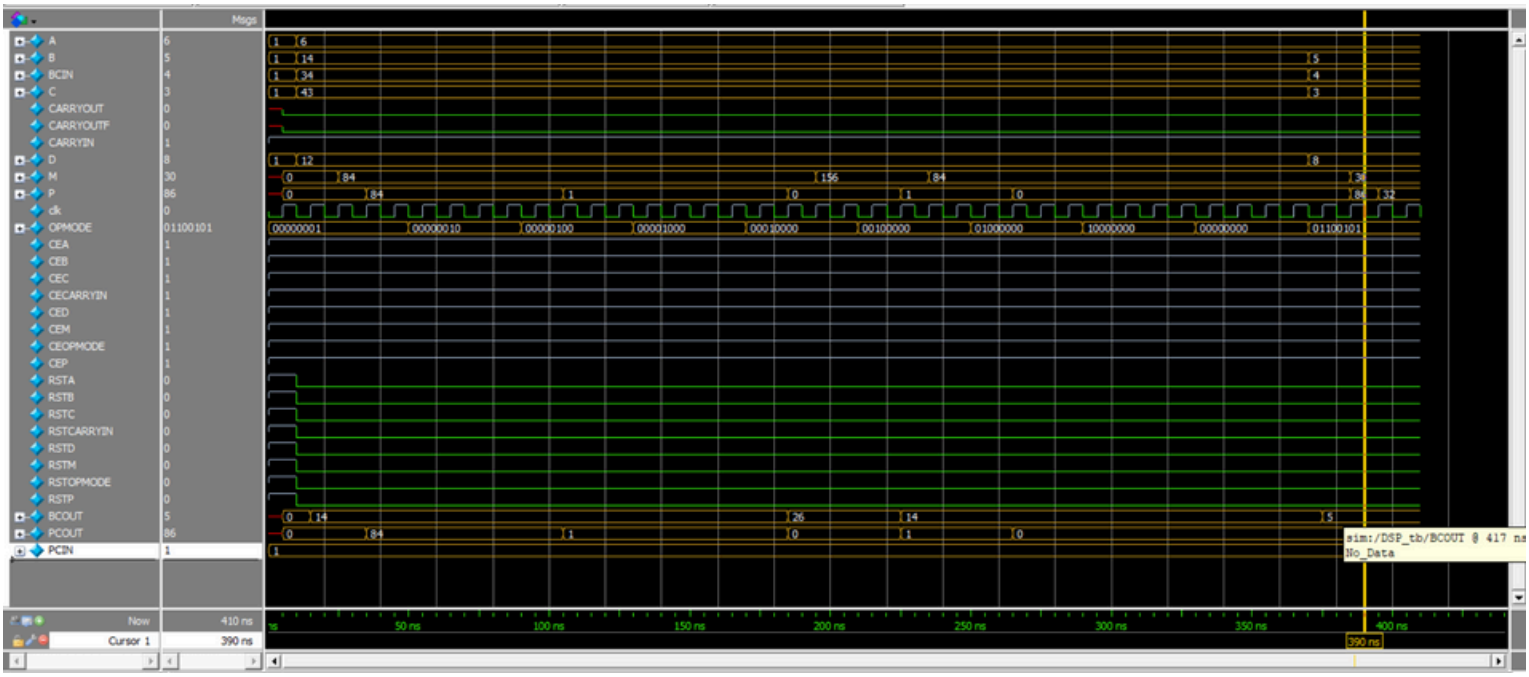
```
      // +++++++++++++++++++++ wrong opmode test ++++++++++++++++++++++++++
        OPMODE = 8'b00000000;
        repeat (4) @(negedge clk);
        if (P !== 0 || M !== A*B || PCOUT !== 0 || BCOUT !== B || CARRYOUT !== 0 || CARRYOUTF !== 0) begin
            $display("Test Failed for wrong OPMODE fail");
            $stop;
        end

        // +++++++++++++++++++++ random opmode test ++++++++++++++++++++++++
        OPMODE = 8'b01100101;
        A = 18'd6; B = 18'd5; C = 48'd3; D = 18'd8; BCIN = 18'd4; CARRYIN = 1'b1;
        repeat (4) @(negedge clk);
        if (P !== (PCIN + (A*B) + 1) || M !== (A*B) || PCOUT !== (PCIN + (A*B)) + 1 || BCOUT !== B || CARRYOUT !== 0 || CARRYOUTF !== 0) begin
            $display("Test Failed for random OPMODE fail: DUT");
            $stop;
        end

        $display("+++++++++++++++++++++");
        $display("All tests passed 👌");
        $display("+++++++++++++++++++++");
        $stop;
    end

    initial begin
        $monitor("%0t : A=%0d, B=%0d, C=%0d, D=%0d, OPMODE=%0b, P=%0d, M=%0d, PCOUT=%0d, BCOUT=%0d, CARRYOUT=%0b, CARRYOUTF=%0b",
            $time, A, B, C, D, OPMODE, P, M, PCOUT, BCOUT, CARRYOUT, CARRYOUTF);
    end
endmodule
```

Do File

```
vlib work
vlog DSP.v MUX.v DSP_tb.v
vsim -voptargs=+acc work.tb
add wave *
run -all
#quit -sim
```

Questa

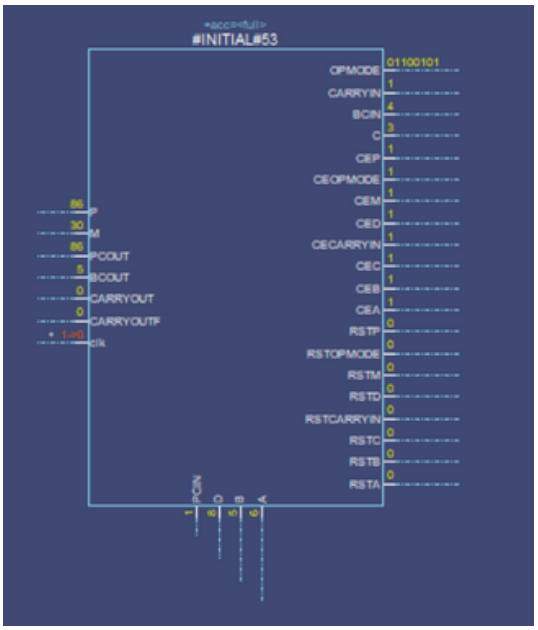
1.Wave Form



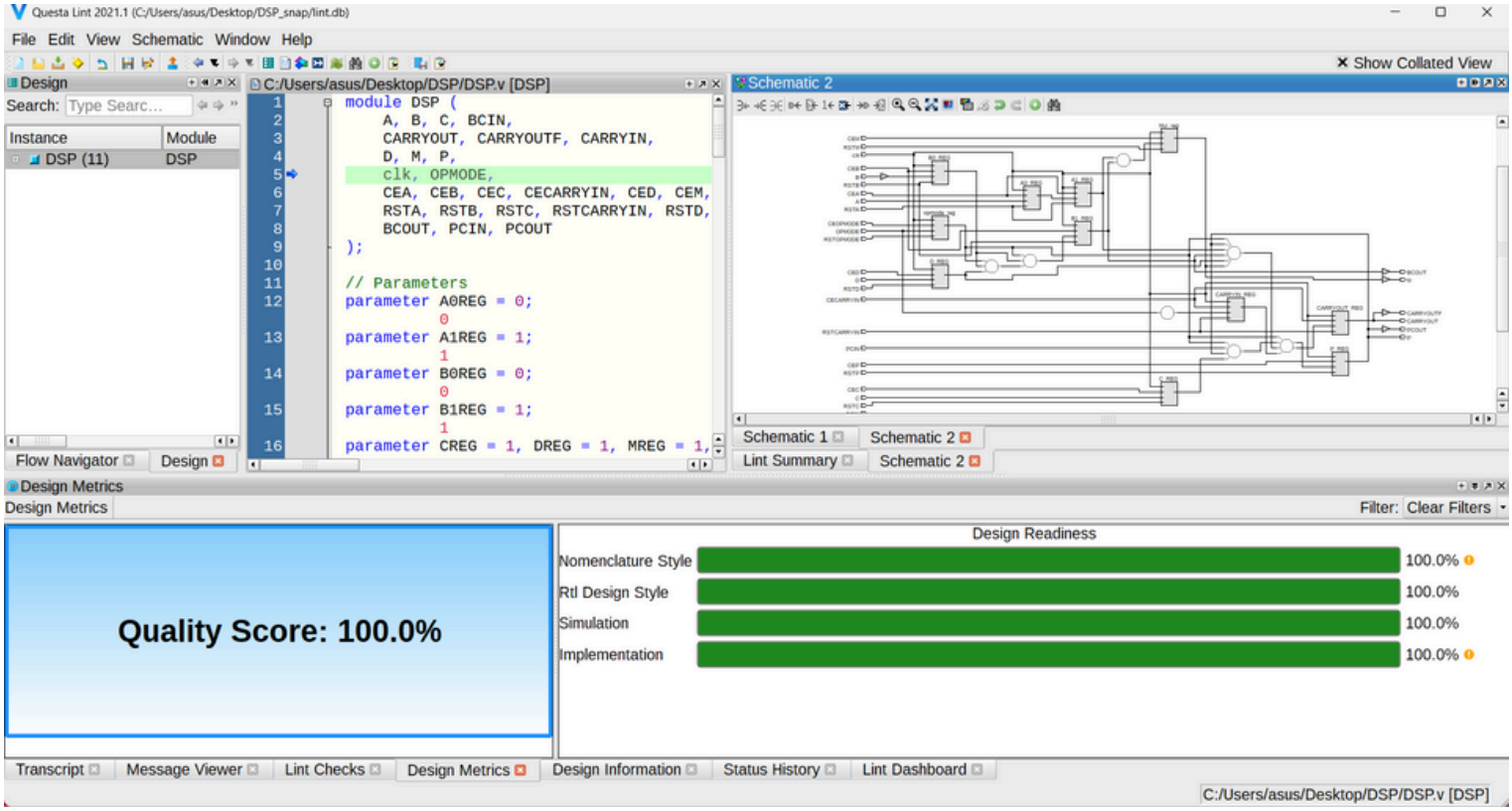
2.Transcript

```
# 0 : A=1, B=1, C=1, D=1, OPMODE=1, P=x, M=x, PCOUT=x, BCOUT=x, CARRYOUT=x, CARRYOUTF=x
# 5 : A=1, B=1, C=1, D=1, OPMODE=1, P=0, M=0, PCOUT=0, BCOUT=0, CARRYOUT=0, CARRYOUTF=0
# 10 : A=6, B=14, C=43, D=12, OPMODE=1, P=0, M=0, PCOUT=0, BCOUT=0, CARRYOUT=0, CARRYOUTF=0
# 15 : A=6, B=14, C=43, D=12, OPMODE=1, P=0, M=0, PCOUT=0, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 25 : A=6, B=14, C=43, D=12, OPMODE=1, P=0, M=84, PCOUT=0, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 35 : A=6, B=14, C=43, D=12, OPMODE=1, P=84, M=84, PCOUT=84, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 50 : A=6, B=14, C=43, D=12, OPMODE=10, P=84, M=84, PCOUT=84, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 90 : A=6, B=14, C=43, D=12, OPMODE=100, P=84, M=84, PCOUT=84, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 105 : A=6, B=14, C=43, D=12, OPMODE=100, P=1, M=84, PCOUT=1, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 130 : A=6, B=14, C=43, D=12, OPMODE=1000, P=1, M=84, PCOUT=1, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 170 : A=6, B=14, C=43, D=12, OPMODE=10000, P=1, M=84, PCOUT=1, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 185 : A=6, B=14, C=43, D=12, OPMODE=10000, P=0, M=84, PCOUT=0, BCOUT=26, CARRYOUT=0, CARRYOUTF=0
# 195 : A=6, B=14, C=43, D=12, OPMODE=10000, P=0, M=156, PCOUT=0, BCOUT=26, CARRYOUT=0, CARRYOUTF=0
# 210 : A=6, B=14, C=43, D=12, OPMODE=100000, P=0, M=156, PCOUT=0, BCOUT=26, CARRYOUT=0, CARRYOUTF=0
# 225 : A=6, B=14, C=43, D=12, OPMODE=100000, P=1, M=156, PCOUT=1, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 235 : A=6, B=14, C=43, D=12, OPMODE=100000, P=1, M=84, PCOUT=1, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 250 : A=6, B=14, C=43, D=12, OPMODE=1000000, P=1, M=84, PCOUT=1, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 265 : A=6, B=14, C=43, D=12, OPMODE=1000000, P=0, M=84, PCOUT=0, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 290 : A=6, B=14, C=43, D=12, OPMODE=10000000, P=0, M=84, PCOUT=0, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 330 : A=6, B=14, C=43, D=12, OPMODE=0, P=0, M=84, PCOUT=0, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 370 : A=6, B=5, C=3, D=8, OPMODE=1100101, P=0, M=84, PCOUT=0, BCOUT=14, CARRYOUT=0, CARRYOUTF=0
# 375 : A=6, B=5, C=3, D=8, OPMODE=1100101, P=0, M=84, PCOUT=0, BCOUT=5, CARRYOUT=0, CARRYOUTF=0
# 385 : A=6, B=5, C=3, D=8, OPMODE=1100101, P=86, M=30, PCOUT=86, BCOUT=5, CARRYOUT=0, CARRYOUTF=0
# 395 : A=6, B=5, C=3, D=8, OPMODE=1100101, P=32, M=30, PCOUT=32, BCOUT=5, CARRYOUT=0, CARRYOUTF=0
# ++++++++++++++++++++++
# All tests passed6
# ++++++++++++++++++++++
# ** Note: $stop    : C:/Users/asus/Desktop/DSP/DSP_tb.v(153)
```

3.Data Flow



Questa Lint

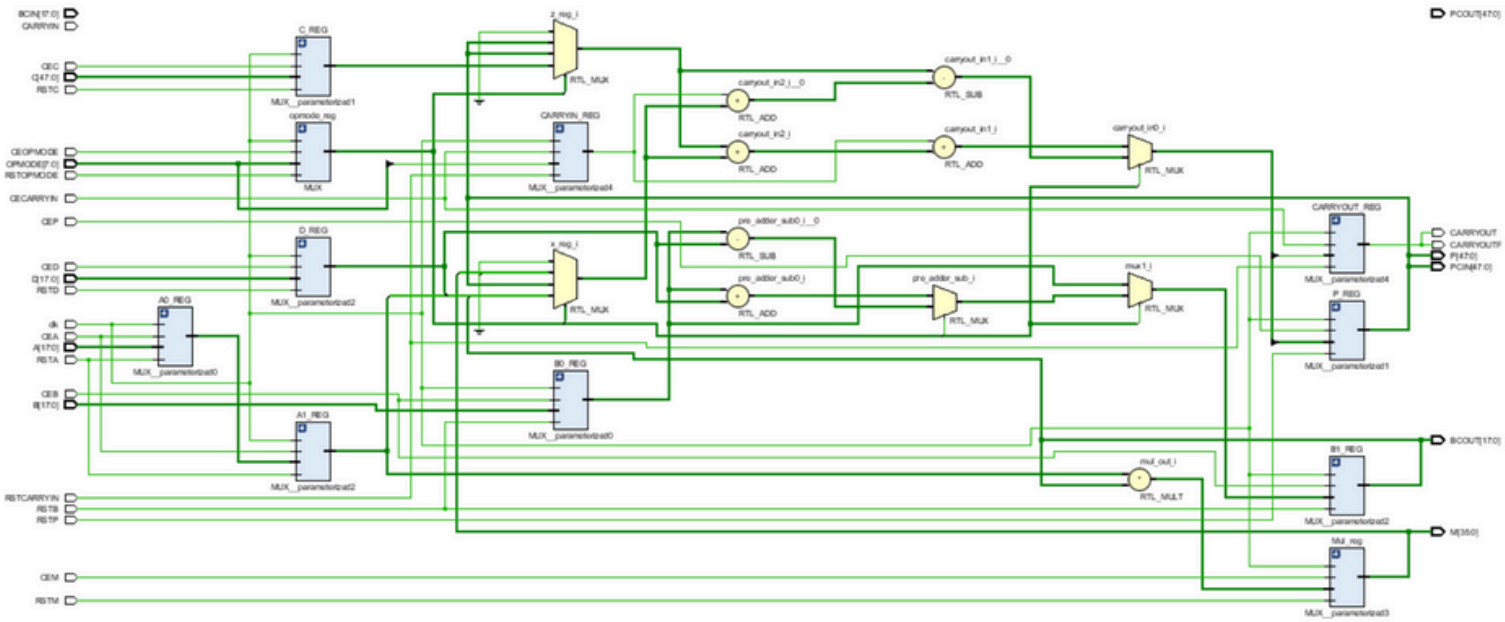


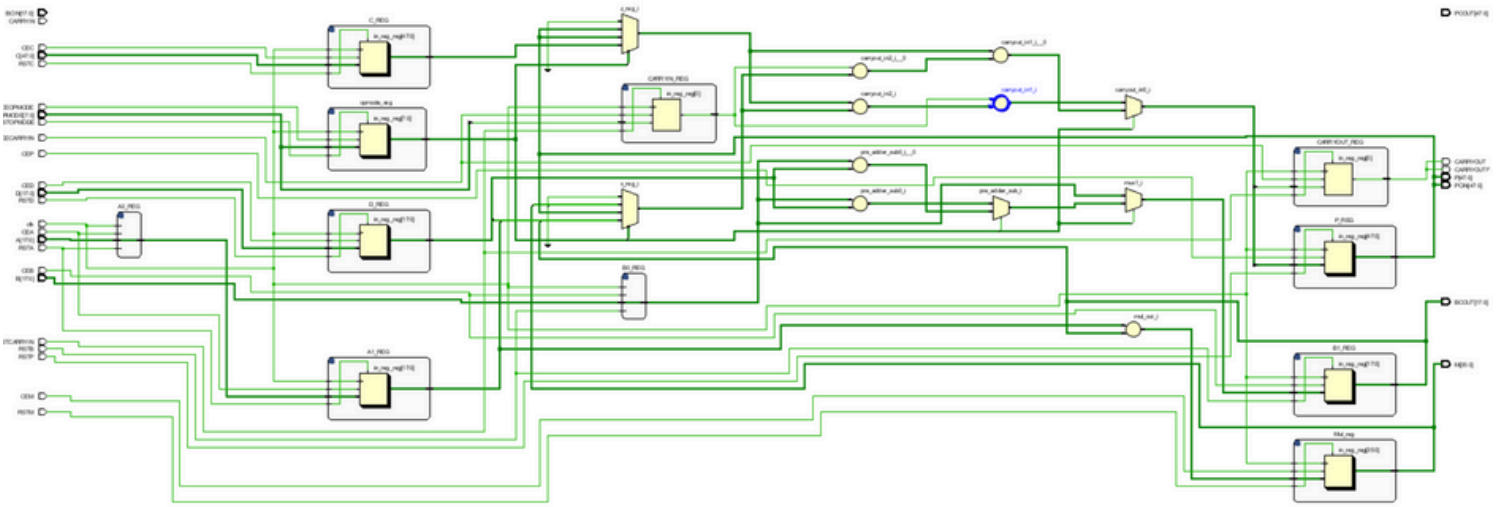
Constraints File

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

## Clock signal
set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
```

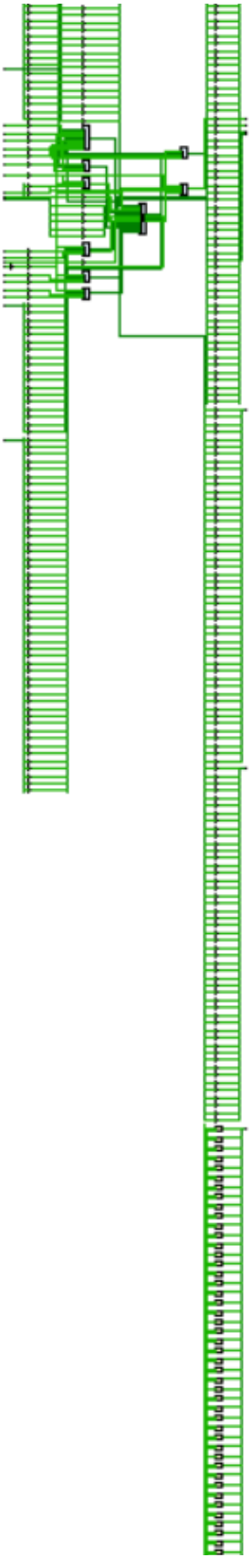
Vivado

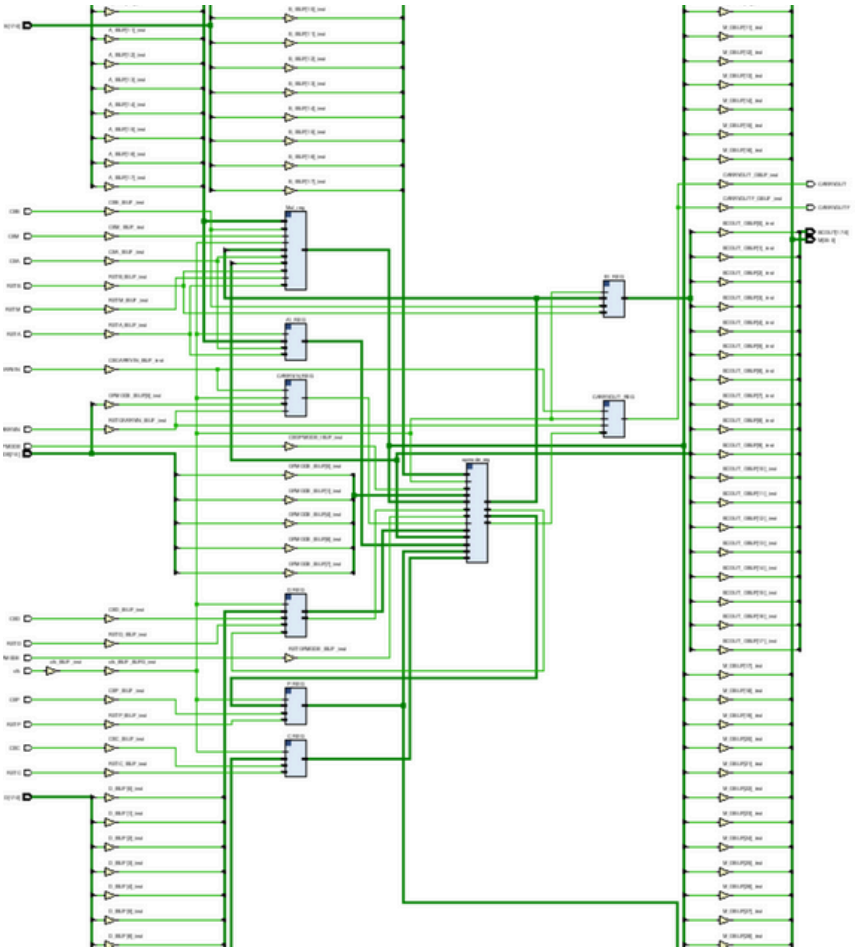
1.Elaborated Design



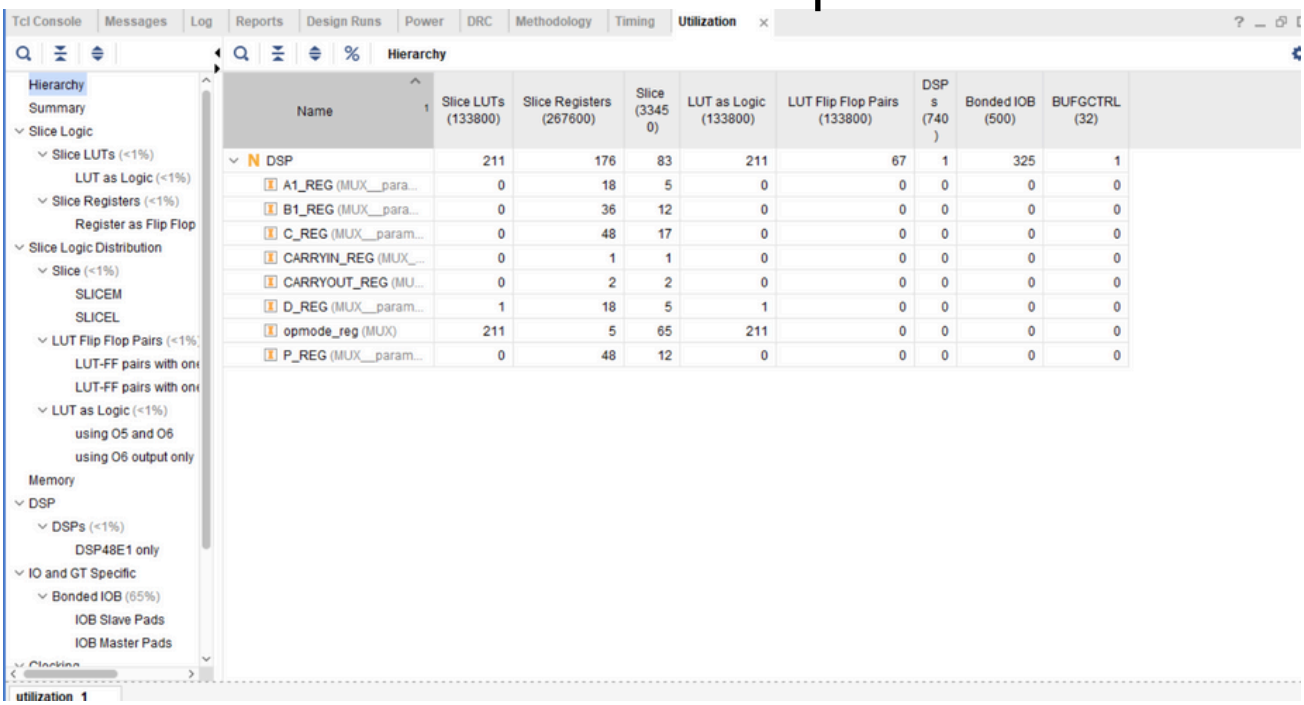


2.Synthesis





4.Utilization Report

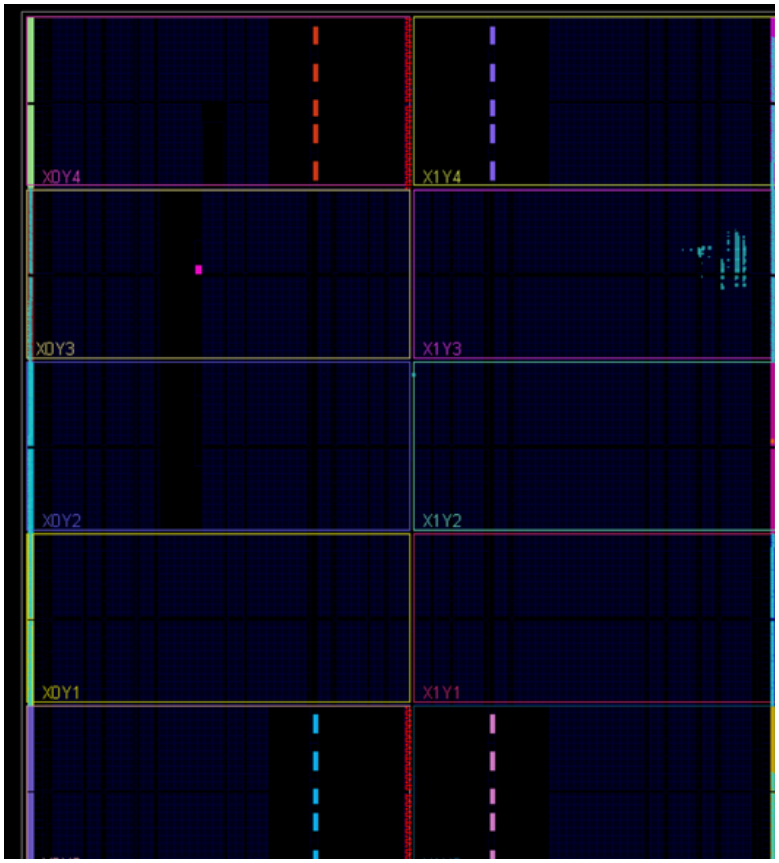


Name	Slice LUTs (133800)	Slice Registers (267600)	Slice (33450)	LUT as Logic (133800)	LUT Flip Flop Pairs (133800)	DSPs (740)	Bonded IOB (500)	BUFGCTRL (32)
N DSP	211	176	83	211	67	1	325	1
A1_REG (MUX__para...	0	18	5	0	0	0	0	0
B1_REG (MUX__para...	0	36	12	0	0	0	0	0
C_REG (MUX__param...	0	48	17	0	0	0	0	0
CARRYIN_REG (MUX_...	0	1	1	0	0	0	0	0
CARRYOUT_REG (MU...	0	2	2	0	0	0	0	0
D_REG (MUX__param...	1	18	5	1	0	0	0	0
opmode_reg (MUX)	211	5	65	211	0	0	0	0
P_REG (MUX__param...	0	48	12	0	0	0	0	0

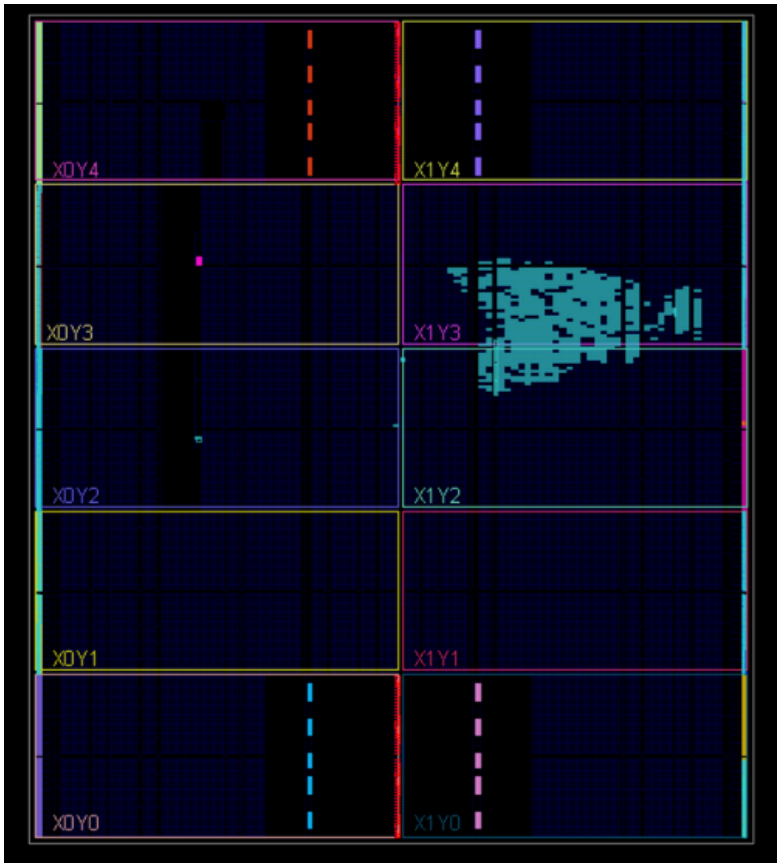
5.Timing Summary Report

Implementation

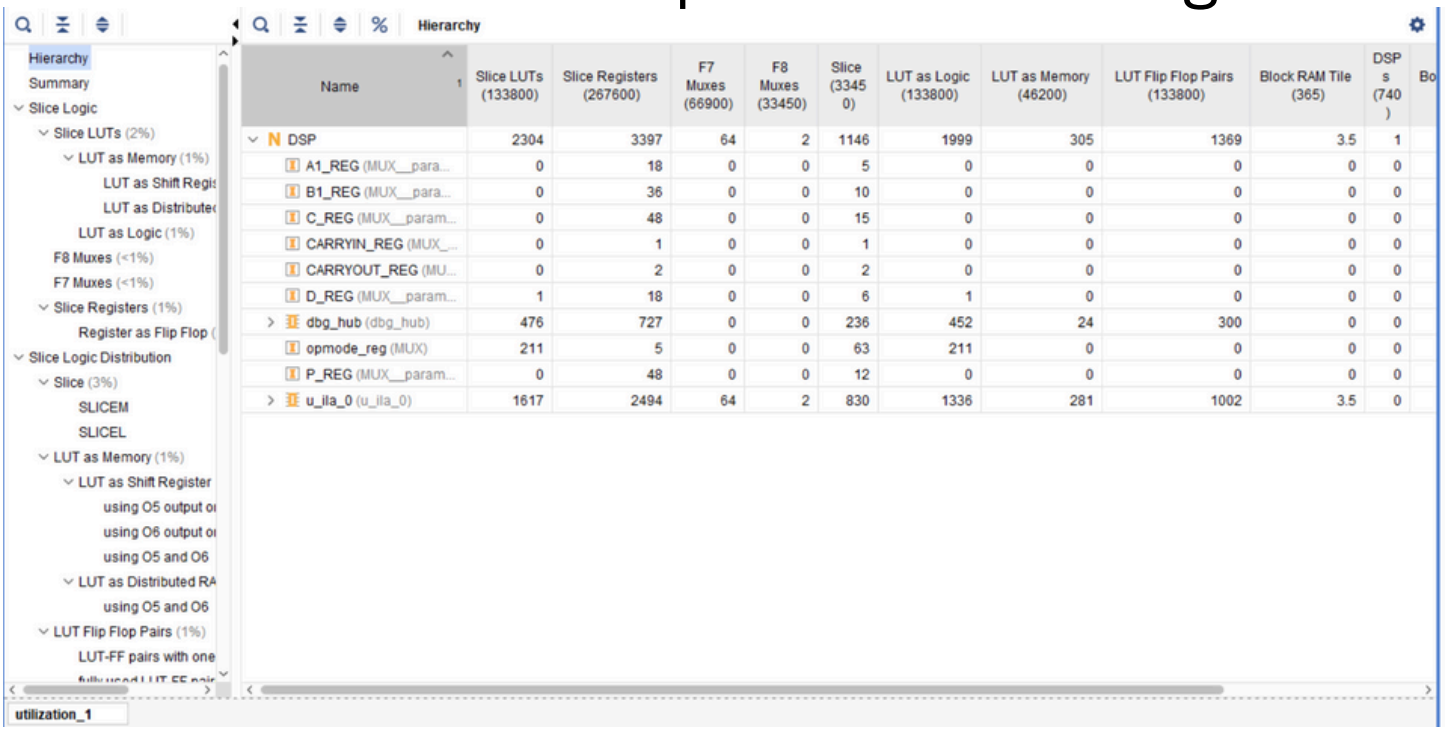
1.Before Debug



2.After Debug



3.Utilization Report After Debug

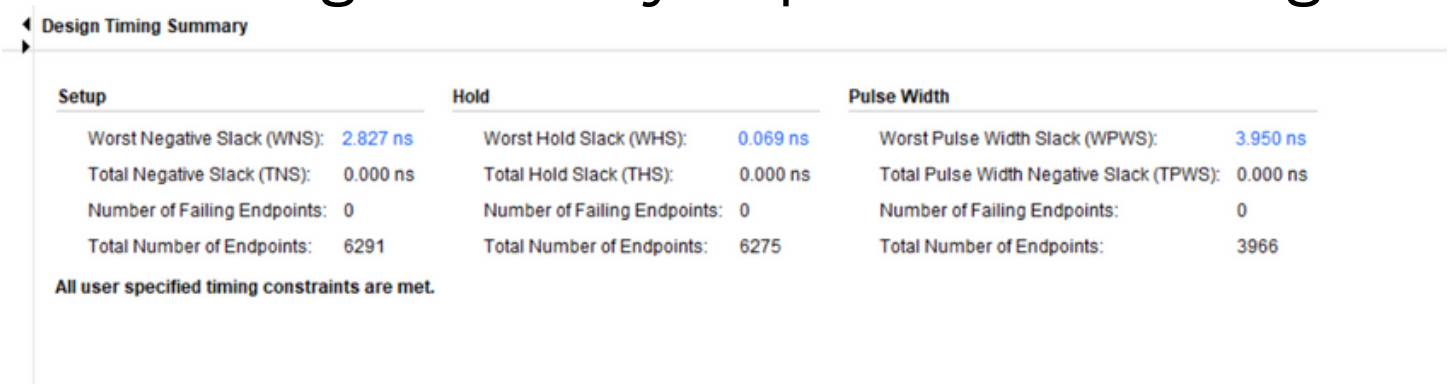


Hierarchy
Summary
Slice Logic
Slice LUTs (2%)
LUT as Memory (1%)
LUT as Shift Regis
LUT as Distribute
LUT as Logic (1%)
F8 Muxes (<1%)
F7 Muxes (<1%)
Slice Registers (1%)
Register as Flip Flop
Slice Logic Distribution
Slice (3%)
SLICEM
SLICEL
LUT as Memory (1%)
LUT as Shift Register
using O5 output o
using O6 output o
using O5 and O6
LUT as Distributed RA
using O5 and O6
LUT Flip Flop Pairs (1%)
LUT-FF pairs with one
utilization_1

Hierarchy

Name	Slice LUTs (133800)	Slice Registers (267600)	F7 Muxes (66900)	F8 Muxes (33450)	Slice (33450)	LUT as Logic (133800)	LUT as Memory (46200)	LUT Flip Flop Pairs (133800)	Block RAM Tile (365)	DSPs (740)	Bo
DSP	2304	3397	64	2	1146	1999	305	1369	3.5	1	
A1_REG (MUX__para...	0	18	0	0	5	0	0	0	0	0	
B1_REG (MUX__para...	0	36	0	0	10	0	0	0	0	0	
C_REG (MUX__param...	0	48	0	0	15	0	0	0	0	0	
CARRYIN_REG (MUX_...	0	1	0	0	1	0	0	0	0	0	
CARRYOUT_REG (MU...	0	2	0	0	2	0	0	0	0	0	
D_REG (MUX__param...	1	18	0	0	6	1	0	0	0	0	
dbg_hub (dbg_hub)	476	727	0	0	236	452	24	300	0	0	
opmode_reg (MUX)	211	5	0	0	63	211	0	0	0	0	
P_REG (MUX__param...	0	48	0	0	12	0	0	0	0	0	
u_ila_0 (u_ila_0)	1617	2494	64	2	830	1336	281	1002	3.5	0	

4.Timing Summary Report After Debug



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.827 ns	Worst Hold Slack (WHS):	0.069 ns	Worst Pulse Width Slack (WPWS):	3.950 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:	6291	Total Number of Endpoints:	6275	Total Number of Endpoints:	3966

All user specified timing constraints are met.

My Problems