Project Proposal ↗

Spartan6 - DSP48A1

Prepared by :

Marwan Mohamed

Table of Contents

1-Introduction

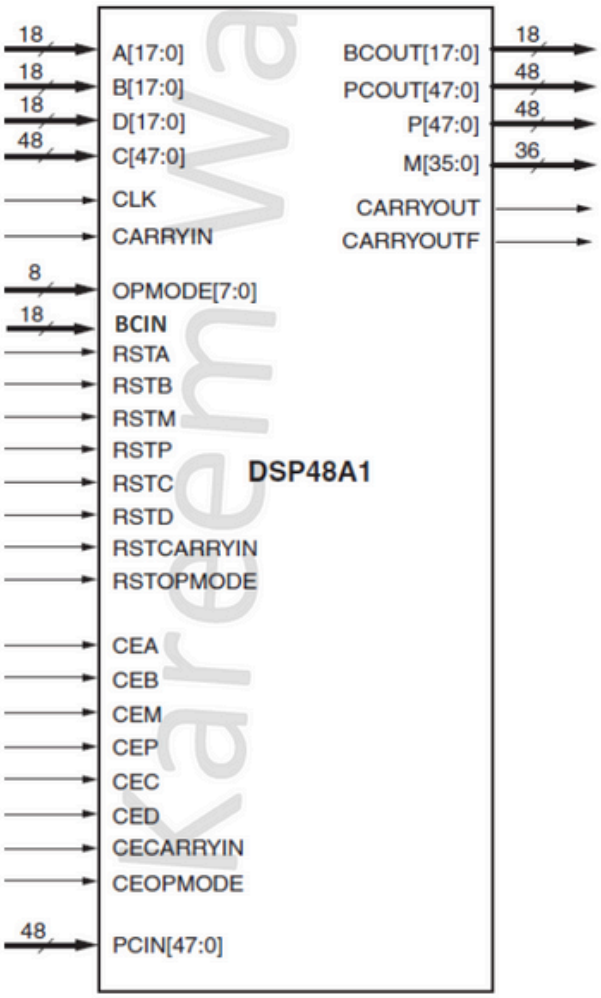
This project focuses on the design and verification of the DSP48A1 slice within Spartan-6 FPGA devices. The DSP48A1 is a powerful arithmetic block optimized for multiply-accumulate operations, making it highly suitable for math-intensive applications such as DSP, image, and signal processing.

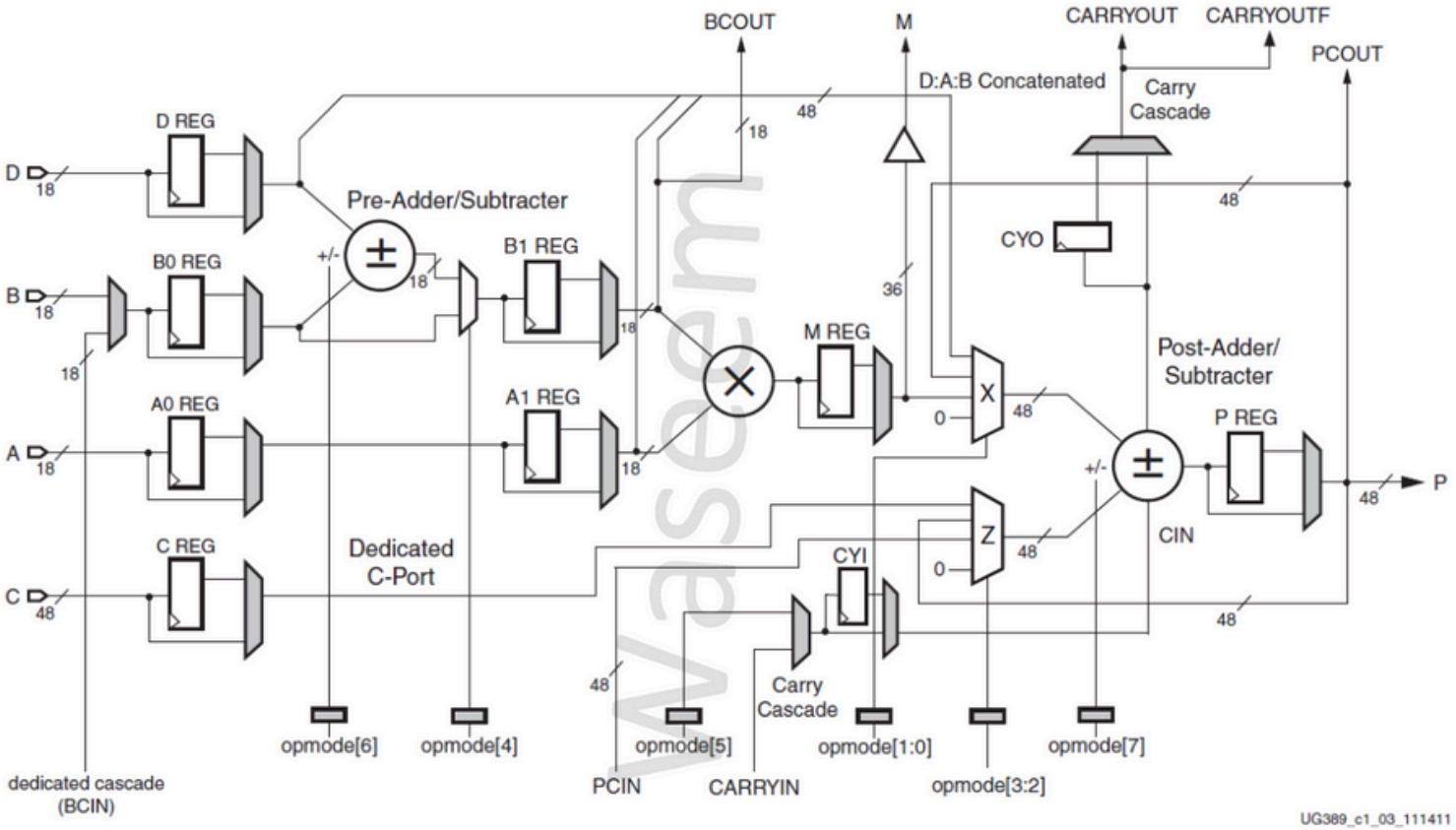
The implementation involved:

- Developing parameterized RTL code with support for pipeline registers, carry handling, and control signals.
- Building a directed testbench in QuestaSim with waveform analysis and do-file automation for efficient verification.
- Running the complete FPGA design flow in Vivado: elaboration, synthesis, implementation, and timing analysis.
- Generating utilization and timing reports to ensure no critical warnings or errors.

This project demonstrates practical experience in RTL design, FPGA toolchains, and verification workflows, bridging theory with real-world hardware implementation.

2-DSP BLOCK


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


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

3-RTL CODE:

MUX :

```
module MUX (X, rst, cen, clk, out);
parameter sel = 0;
parameter size = 18;
parameter rsttype = "SYNC";
input [size - 1 : 0] X;
input rst, clk, cen;
output reg [size - 1 : 0] out;

reg [size - 1 : 0 ] X_reg;

generate
  if (rsttype == "SYNC")
    begin
      always @ (posedge clk)
        begin
          if (rst)
            X_reg <= 0;
          else if (cen)
            X_reg <= X;
        end
    end
  else
    begin
      always @ (posedge clk, posedge rst)
        begin
          if (rst)
            X_reg <= 0;
          else if (cen)
            X_reg <= X;
        end
    end
endgenerate

always @ (*)
  begin
    if (sel)
      out = X_reg;
    else
      out = X;
  end
endmodule
```

RTL CODE:

DSP :

```
module dsp (A,B,C,D,PCIN,BCIN,CARRYIN,CLK,OPMODE,CEA,CEB,CEC,CED,CEM,CEP,CEOPMODE,CECARRYIN,RSTA,
            RSTB,RSTC,RSTD,RSTP,RSTM,RSTCARRYIN,RSTOPMODE,BCOUT,PCOUT,M,P,CARRYOUT,CARRYOUTF);
parameter A0REG = 0; // If equal (0) --> No register, else --> Registered
parameter A1REG = 1; // If equal (0) --> No register, else --> Registered
parameter B0REG = 0; // If equal (0) --> No register, else --> Registered
parameter B1REG = 1; // If equal (0) --> No register, else --> Registered
parameter CREG =1;   // If equal (0) --> No register, else --> Registered
parameter DREG =1;   // If equal (0) --> No register, else --> Registered
parameter MREG =1;   // If equal (0) --> No register, else --> Registered
parameter PREG =1;   // If equal (0) --> No register, else --> Registered
parameter CARRYINREG =1; // If equal (0) --> No register, else --> Registered
parameter CARRYOUTREG =1; // If equal (0) --> No register, else --> Registered
parameter OPMODEREG =1;   // If equal (0) --> No register, else --> Registered
parameter CARRYINSEL = "OPMODE5";  // Selects between CARRYIN and opcode[5]
parameter BINPUT = "DIRECT";      // Selects between direct (B) input and cascaded input (BCIN)
parameter RSTTYPE = "SYNC";      // Selects whether all registers have a synchronous or asynchronous reset

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

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

wire [17 : 0] D_OUT, B_OUT, A_OUT;
wire [47 : 0] C_OUT;
wire [7 : 0] OPMODE_OUT;
wire [17 : 0] b_out, PRE_ADD_SUB_OUT, SEL1, b1_reg_out, a1_reg_out;
wire [35 : 0] MUL_OUT, M_OUT;
wire CARRYCASCADE, CIN_OUT, CARRYOUT_IN;
wire [47 : 0] POST_ADD_SUB_OUT;
reg [47 : 0] X_OUT, Z_OUT;
```

```
assign b_out = (BINPUT == "DIRECT") ? B : ((BINPUT == "CASCADE") ? BCIN : 18'b0);

MUX #(.sel(DREG), .size(18), .rsttype(RSTTYPE)) d (.X(D), .rst(RSTD), .cen(CED), .clk(CLK), .out(D_OUT)); // D instantiation
MUX #(.sel(B0REG), .size(18), .rsttype(RSTTYPE)) b (.X(b_out), .rst(RSTB), .cen(CEB), .clk(CLK), .out(B_OUT)); // B0 instantiation
MUX #(.sel(A0REG), .size(18), .rsttype(RSTTYPE)) a (.X(A), .rst(RSTA), .cen(CEA), .clk(CLK), .out(A_OUT)); // A0 instantiation
MUX #(.sel(CREG), .size(48), .rsttype(RSTTYPE)) c (.X(C), .rst(RSTC), .cen(CEC), .clk(CLK), .out(C_OUT)); // C instantiation
MUX #(.sel(OPMODEREG), .size(8), .rsttype(RSTTYPE)) opmode (.X(OPMODE), .rst(RSTOPMODE), .cen(CEOPMODE), .clk(CLK), .out(OPMODE_OUT)); // C

assign PRE_ADD_SUB_OUT = (OPMODE_OUT[6] == 1'b0) ? (D_OUT + B_OUT) : (D_OUT - B_OUT);
assign SEL1 = (OPMODE_OUT[4] == 1'b0) ? (B_OUT) : (PRE_ADD_SUB_OUT);

MUX #(.sel(B1REG), .size(18), .rsttype(RSTTYPE)) b1reg (.X(SEL1), .rst(RSTB), .cen(CEB), .clk(CLK), .out(b1_reg_out)); // B1 instantiation
MUX #(.sel(A1REG), .size(18), .rsttype(RSTTYPE)) a1reg (.X(A_OUT), .rst(RSTA), .cen(CEA), .clk(CLK), .out(a1_reg_out)); // A1 instantiation

assign MUL_OUT = a1_reg_out * b1_reg_out;
assign BCOUT = b1_reg_out;

MUX #(.sel(MREG), .size(36), .rsttype(RSTTYPE)) m (.X(MUL_OUT), .rst(RSTM), .cen(CEM), .clk(CLK), .out(M_OUT)); // M instantiation
assign CARRYCASCADE = (CARRYINSEL == "OPMODE5") ? (OPMODE_OUT[5]) : ((CARRYINSEL == "CARRYIN") ? (CARRYIN) : (1'b0));
MUX #(.sel(CARRYINREG), .size(1), .rsttype(RSTTYPE)) carryin (.X(CARRYCASCADE), .rst(RSTCARRYIN), .cen(CECARRYIN), .clk(CLK), .out(CIN_OUT));
assign M = M_OUT;

always @ (*)
  begin
    case (OPMODE_OUT[1 : 0])
      2'b00 : X_OUT = 48'b0;
      2'b01 : X_OUT = M_OUT;
      2'b10 : X_OUT = P;
      2'b11 : X_OUT = {D_OUT[11 : 0], a1_reg_out, b1_reg_out};
      default : X_OUT = 48'b0;
    endcase
  end
```

```
always @ (*)
  begin
    case (OPMODE_OUT[3 : 2])
      2'b00 : Z_OUT = 48'b0;
      2'b01 : Z_OUT = PCIN;
      2'b10 : Z_OUT = P;
      2'b11 : Z_OUT = C_OUT;
      default : Z_OUT = 48'b0;
    endcase
  end
assign {CARRYOUT_IN ,POST_ADD_SUB_OUT} = (OPMODE_OUT[7] == 1'b0) ? (Z_OUT + X_OUT + CIN_OUT) : (Z_OUT - (X_OUT + CIN_OUT));
MUX #(.sel(CARRYOUTREG), .size(1), .rsttype(RSTTYPE)) carryout (.X(CARRYOUT_IN), .rst(RSTCARRYIN), .cen(CECARRYIN), .clk(CLK), .out(CARRYOUT));
assign CARRYOUTF = CARRYOUT;

MUX #(.sel(PREG), .size(48), .rsttype(RSTTYPE)) p (.X(POST_ADD_SUB_OUT), .rst(RSTP), .cen(CEP), .clk(CLK), .out(P)); // P instantiation
assign PCOUT = P;

endmodule
```

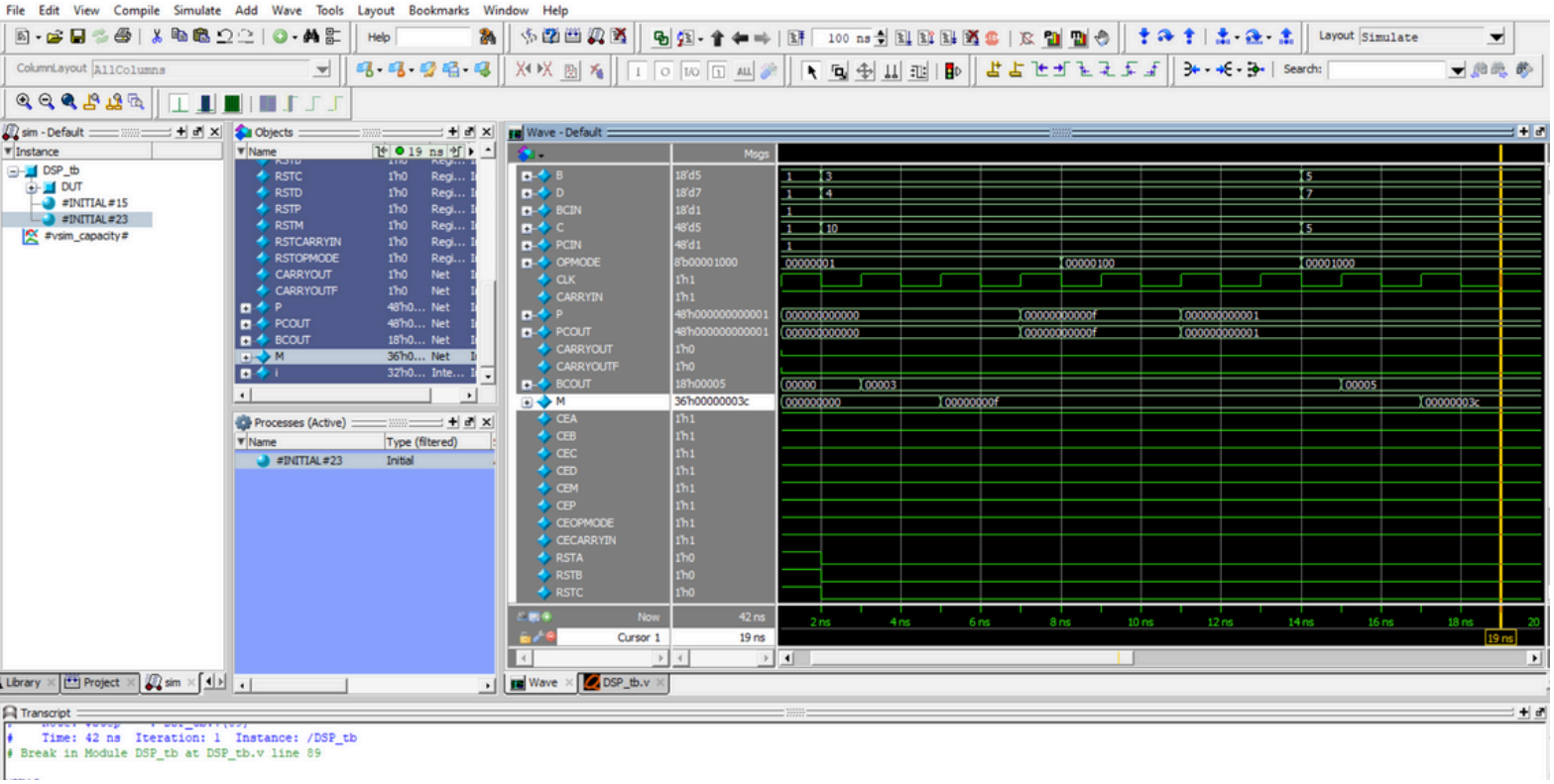
4-Testbench Code:

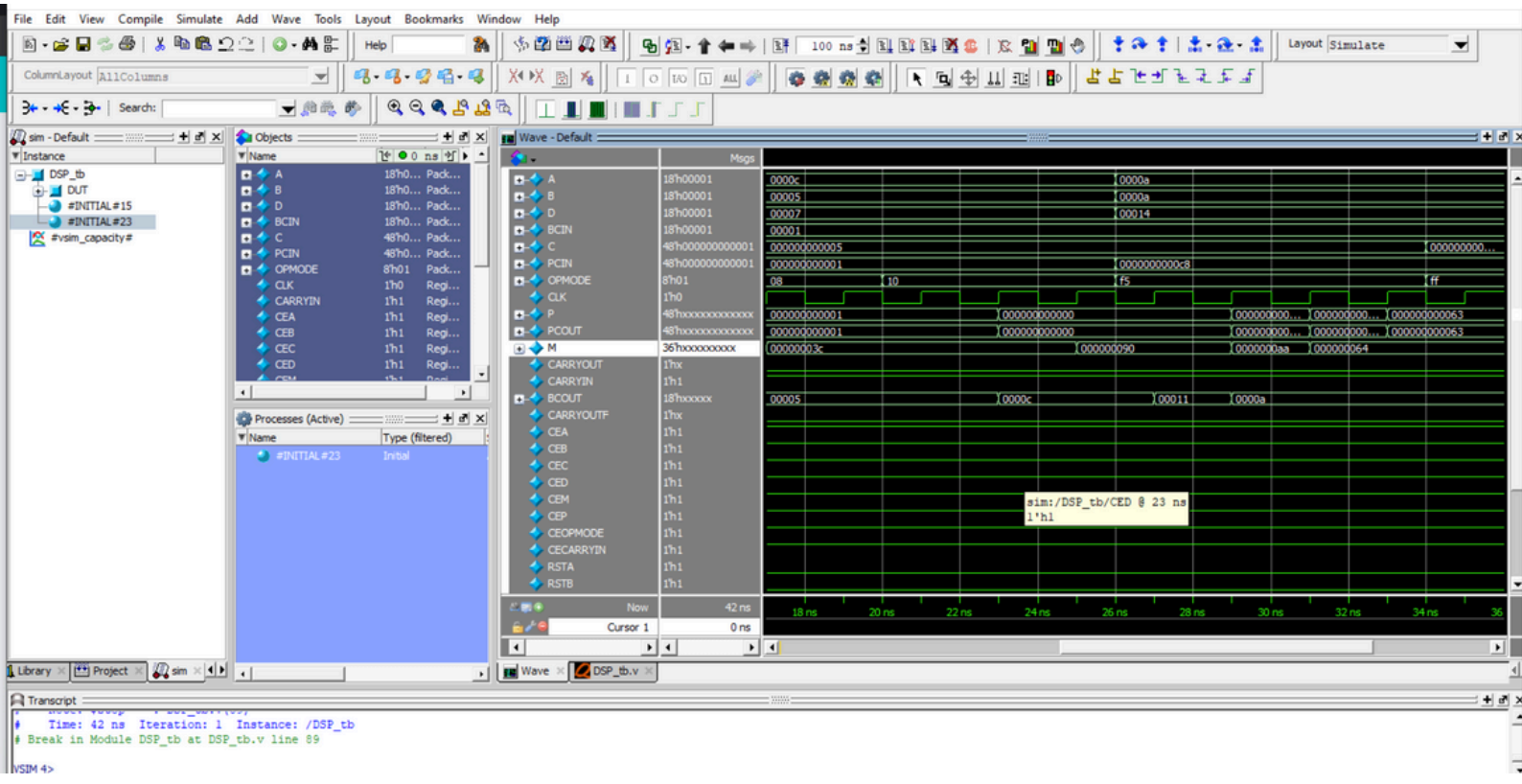
```
module DSP_tb ();
reg [17:0] A,B,D,BCIN;
reg [47:0] C,PCIN;
reg [7:0] OPMODE;
reg CLK,CARRYIN;
reg CEA,CEB,CEC,CED,CEM,CEP,CEOPMODE,CECARRYIN;
reg RSTA,RSTB,RSTC,RSTD,RSTP,RSTM,RSTCARRYIN,RSTOPMODE;

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

dsp DUT(.*);
initial begin
  CLK =0;
  forever
#1 CLK = ~ CLK;
end

integer i = 0;

initial begin
  // reset check
  RSTA = 1'b1; RSTB = 1'b1; RSTM = 1'b1; RSTP = 1'b1; RSTC = 1'b1; RSTD = 1'b1; RSTCARRYIN = 1'b1; RSTOPMODE = 1'b1;
  CEA = 1'b1; CEB = 1'b1; CEM = 1'b1; CEP = 1'b1; CEC = 1'b1; CED = 1'b1; CECARRYIN = 1'b1; CEOPMODE = 1'b1;
  PCIN = 48'b1; BCIN = 18'b1; OPMODE = 8'b1; CARRYIN = 1'b1;
  A = 18'b1; B = 18'b1; D = 18'b1; C = 48'b1;
  @ (negedge CLK)
  RSTA = 1'b0; RSTB = 1'b0; RSTM = 1'b0; RSTP = 1'b0; RSTC = 1'b0; RSTD = 1'b0; RSTCARRYIN = 1'b0; RSTOPMODE = 1'b0;
  if ((BCOUT !== 0) || (PCOUT !== 0) || (P !== 0) || (M !== 0) || (CARRYOUT !== 0) || (CARRYOUTF !== 0))
    begin
      $display ("Error - Incorrect output");
      $stop;
    end
```

```
initial begin
  if ((BCOUT !== 0) || (PCOUT !== 0) || (P !== 0) || (M !== 0) || (CARRYOUT !== 0) || (CARRYOUTF !== 0))
  // opmode =  8'b00000001
  A = 18'd5; B = 18'd3; C = 18'd10; D = 18'd4;
  OPMODE = 8'b00000001;
  repeat (3) @ (negedge CLK);
  if ((PCOUT !== (B * A)) || (P !== (B * A)) || (M !== (B * A)) || (CARRYOUT !== 0) || (CARRYOUTF !== 0) || (BCOUT !== B))
    begin
      $display ("Error - Incorrect output");
      $stop;
    end
  // opmode = 8'b00000100
  OPMODE = 8'b00000100;
  repeat (3) @(negedge CLK);
  if ((PCOUT !== PCIN) || (P !== PCIN) || (M !== (B * A)) || (CARRYOUT !== 0) || (CARRYOUTF !== 0) || (BCOUT !== B))
    begin
      $display ("Error - Incorrect output");
      $stop;
    end
    // OPMODE = 8'b00001000
    A = 18'd12; B = 18'd5; C = 18'd5; D = 18'd7;
    OPMODE = 8'b00001000;
    repeat (3) @(negedge CLK);
    if ((PCOUT !== P) || (P !== P) || (M !== (B * A)) || (CARRYOUT !== 0) || (CARRYOUTF !== 0) || (BCOUT !== B))
    begin
      $display ("Error - Incorrect output");
      $stop;
    end
    // OPMODE = 8'b00010000
    OPMODE = 8'b00010000;
    repeat (3) @(negedge CLK);
    if ((PCOUT !== 0) || (P !== 0) || (M !== ((B + D)*A)) || (CARRYOUT !== 0) || (CARRYOUTF !== 0) || (BCOUT !== (B + D)))
    begin
      $display ("Error - Incorrect output");
      $stop;
    end
```

```
  // OPMODE = 8'b11110101
  A = 18'd10; B = 18'd10; C = 18'd5; D = 18'd20; PCIN = 48'd200;
  OPMODE = 8'b11110101;
  repeat (4) @(negedge CLK);
  if ((P !== (PCIN-(((D - B) * A) + OPMODE[5]))) || (PCOUT !== P) || (M !== ((D - B) * A)) || (CARRYOUT !== 0) || (CARRYOUTF !== 0) || (
  begin
    $display ("Error - Incorrect output");
    $stop;
  end

  // OPMODE = 8'b11111111
  A = 18'd10; B = 18'd10; C = 18'd300; D = 18'd20; PCIN = 48'd200;
  OPMODE = 8'b11111111;
  repeat (4) @(negedge CLK);
  if ((P !== (C - ( {D[11 : 0], A, B}+ OPMODE[5]))) || (PCOUT !== P) || (M !== ((D - B) * A)) || (BCOUT !== (D - B)))
  begin
    $display ("Error - Incorrect output");
    $stop;
  end
  $stop;
end
initial begin
  $monitor ("%t :OPMODE = %0d , A = %0d , B = %0d , C = %0d , D = %0d , P = %0d ,
             PCOUT = %0d , M = %0d , BCOUT = %0d , CARRYOUT = %0d , CARRYOUTF = %0d",
              $time, OPMODE, A, B, C, D, P , PCOUT, M, BCOUT, CARRYOUT, CARRYOUTF);
end
endmodule //DSP_tb
```

5-Waveform Snippet:



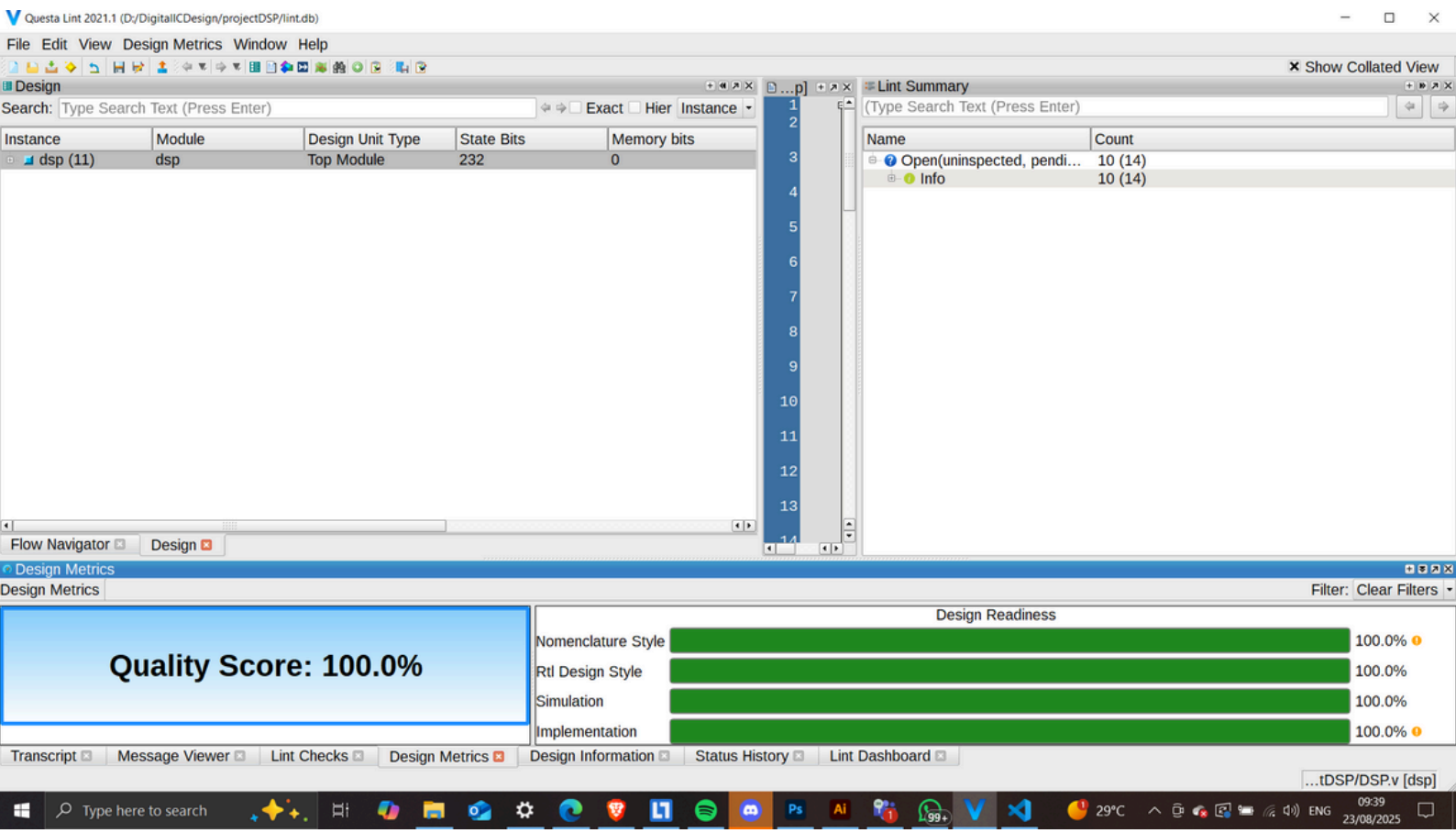


6-Do File :

run.do

```
1    vlib work
2    vlog DSP.v DSP_tb.v MUX.v
3    vsim -voptargs=+acc work.DSP_tb
4    add wave *
5    run -all
6    #quit -sim
```

7-QuestaLint run :



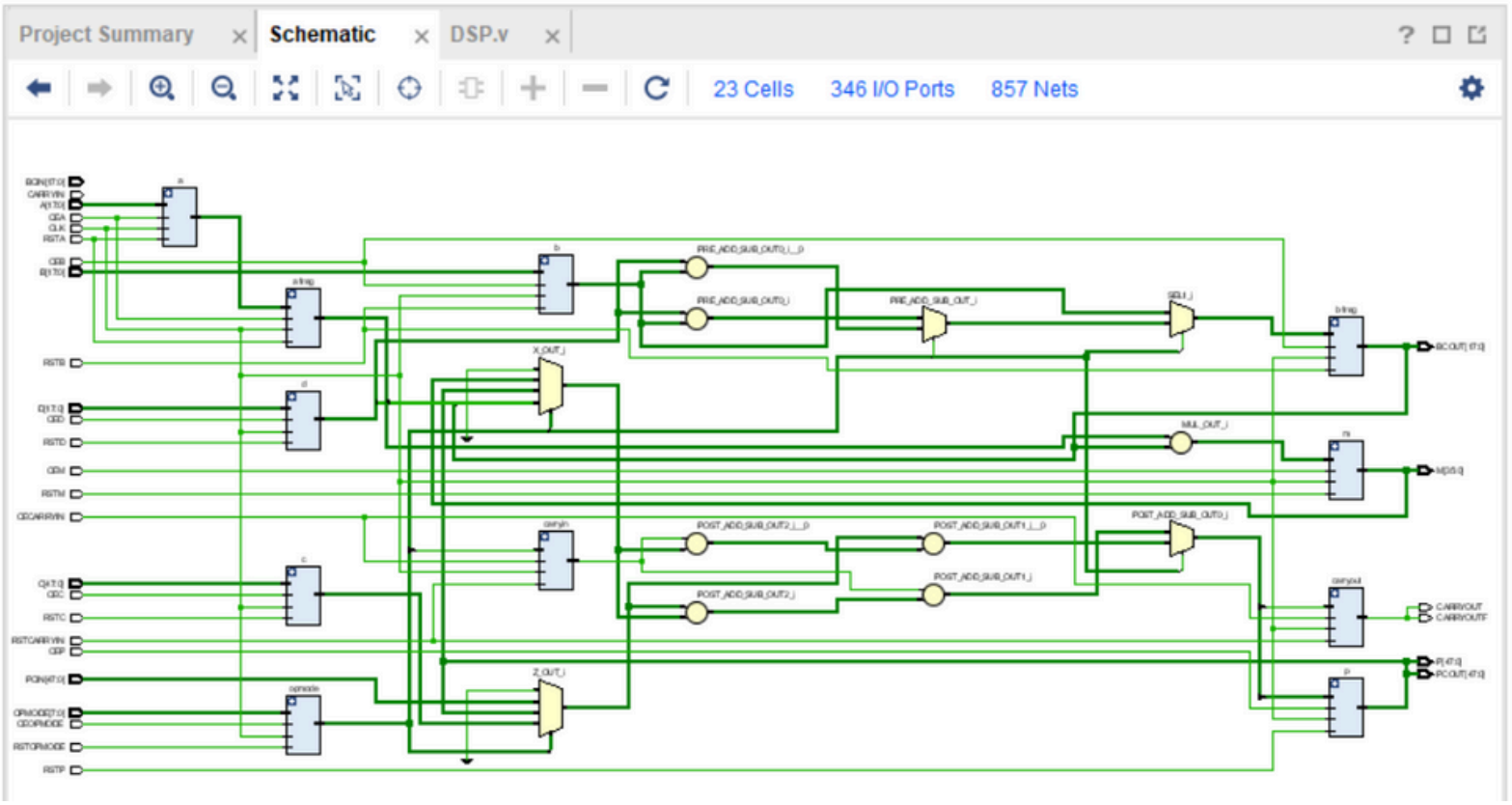
8-Constraint file :

Constraints_basys3.xdc

```
## 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]


## Switches
#set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {sw[0]}]
#set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {sw[1]}]
#set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports {sw[2]}]
#set_property -dict { PACKAGE_PIN W17   IOSTANDARD LVCMOS33 } [get_ports {sw[3]}]
#set_property -dict { PACKAGE_PIN W15   IOSTANDARD LVCMOS33 } [get_ports {sw[4]}]
#set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports {sw[5]}]
#set_property -dict { PACKAGE_PIN W14   IOSTANDARD LVCMOS33 } [get_ports {sw[6]}]
#set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports {sw[7]}]
#set_property -dict { PACKAGE_PIN V2    IOSTANDARD LVCMOS33 } [get_ports {sw[8]}]
#set_property -dict { PACKAGE_PIN T3    IOSTANDARD LVCMOS33 } [get_ports {sw[9]}]
#set_property -dict { PACKAGE_PIN T2    IOSTANDARD LVCMOS33 } [get_ports {sw[10]}]
#set_property -dict { PACKAGE_PIN R3    IOSTANDARD LVCMOS33 } [get_ports {sw[11]}]
#set_property -dict { PACKAGE_PIN W2    IOSTANDARD LVCMOS33 } [get_ports {sw[12]}]
#set_property -dict { PACKAGE_PIN U1    IOSTANDARD LVCMOS33 } [get_ports {sw[13]}]
#set_property -dict { PACKAGE_PIN T1    IOSTANDARD LVCMOS33 } [get_ports {sw[14]}]
#set_property -dict { PACKAGE_PIN R2    IOSTANDARD LVCMOS33 } [get_ports {sw[15]}]


## LEDs
#set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {led[0]}]
#set_property -dict { PACKAGE_PIN E19   IOSTANDARD LVCMOS33 } [get_ports {led[1]}]
#set_property -dict { PACKAGE_PIN U19   IOSTANDARD LVCMOS33 } [get_ports {led[2]}]
#set_property -dict { PACKAGE_PIN V19   IOSTANDARD LVCMOS33 } [get_ports {led[3]}]
#set_property -dict { PACKAGE_PIN W18   IOSTANDARD LVCMOS33 } [get_ports {led[4]}]
#set_property -dict { PACKAGE_PIN U15   IOSTANDARD LVCMOS33 } [get_ports {led[5]}]
#set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports {led[6]}]
```

Ln 8, Col 81 Spaces: 4 UTF-8 CR

9-Elaborated Design :



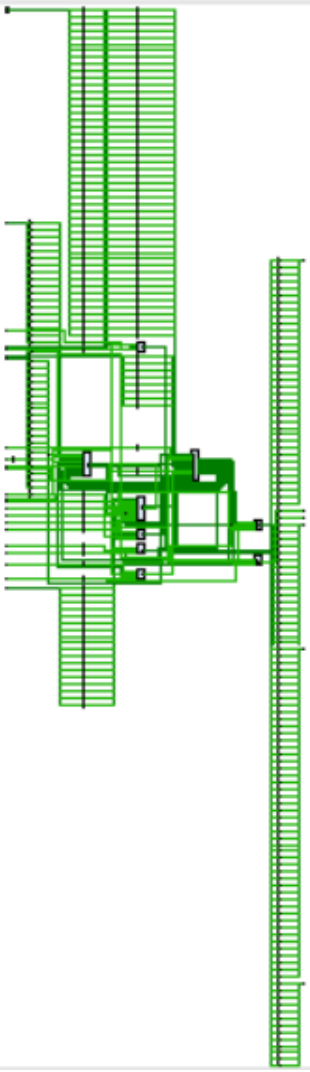
10-Synthesized Design :

1-Timing Report

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.

2-schematic



3- Utilization Report

og | Reports | Design Runs | Utilization | Debug

Hierarchy

Name	Slice LUTs (134600)	Slice Registers (269200)	DSPs (740)	Bonded IOB (500)	BUFGCTRL (32)
dsp	230	160	1	327	1
a1reg (MUX)	0	18	0	0	0
b1reg (MUX_0)	0	18	0	0	0
c (MUX__parameterize...	0	48	0	0	0
carryin (MUX__parame...	1	1	0	0	0
carryout (MUX__param...	0	1	0	0	0
d (MUX_2)	0	18	0	0	0
m (MUX__parameteriz...	0	0	1	0	0
opmode (MUX__para...	228	8	0	0	0
p (MUX__parameterize...	0	48	0	0	0

11-Implementation :

1- Utilization Report

Reports | Design Runs | Power | DRC | Methodology | Timing | **Utilization**

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)
dsp	229	179	98	229	50	1	327	1
a1reg (MUX)	0	18	6	0	0	0	0	0
b1reg (MUX_0)	0	36	10	0	0	0	0	0
c (MUX__parameterize...	0	48	14	0	0	0	0	0
carryin (MUX__parame...	1	1	1	1	1	0	0	0
carryout (MUX__param...	0	2	2	0	0	0	0	0
d (MUX_2)	0	18	10	0	0	0	0	0
m (MUX__parameteriz...	0	0	0	0	0	1	0	0
opmode (MUX__para...	228	8	75	228	0	0	0	0
p (MUX__parameterize...	0	48	12	0	0	0	0	0

2-Timing Report

orts | Design Runs | Power | DRC | Methodology | **Timing**

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.858 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.

3-Device