يوسف أحمد محمد ابراهيم

# Project 1

DSP48A1

# Circuit_1 :

- **RTL code:**

```
module DSP48A1#(
    //Parameters :
    parameter A0_REG     = 0,
    parameter A1_REG     = 1,
    parameter B0_REG     = 0,
    parameter B1_REG     = 1,
    parameter C_REG      = 1,
    parameter D_REG      = 1,
    parameter M_REG      = 1,
    parameter P_REG      = 1,
    parameter CIN_REG    = 1,
    parameter COUT_REG   = 1,
    parameter OP_REG     = 1,
    parameter CIN_SEL    = "OPMODE5",
    parameter B_INPUT    = "DIRECT",
    parameter RST_TYPE   = "SYNC"
)(
```

```
)( //I/O declarations
    input   [17:0]  A,
    input   [17:0]  B,
    input   [47:0]  C,
    input   [17:0]  D,
    input           clk,
    input           CARRYIN,
    input   [7:0]   OPMODE,
    input   [17:0]  BCIN,
    input           rstA,
    input           rstB,
    input           rstC,
    input           rstD,
    input           rstP,
    input           rstM,
    input           rstCARRY,
    input           rstOP,
    input           CEA,
    input           CEB,
    input           CEC,
    input           CED,
    input           CEM,
    input           CEP,
    input           CECIN,
    input           CEOP,
    input   [47:0]  PCIN,
    output  [17:0]  BCOUT,
    output  [47:0]  PCOUT,
    output  [47:0]  P,
    output  [35:0]  M,
    output          CARRYOUT,
    output          CARRYOUTF
);
```

```
49  );
50      //internal signals
51      reg [17:0]  A0, A1;
52      reg [17:0]  B0, B1;
53      reg [47:0]  C0;
54      reg [17:0]  D0;
55      reg [35:0]  M0;
56      reg [47:0]  P0;
57      reg         CYO;
58      reg         CYI;
59      reg [7:0]   OP0;
60
61      wire        carryIN_cascade;
62      wire        carryOUT_cascade;
63      wire        CIN;
64      wire        COUT;
65      wire [35:0] multiplier_out;
66      wire [17:0] pre_adder_out;
67      wire [17:0] pre_adder_Din;
68      wire [17:0] pre_adder_Bin;
69      wire [17:0] A1_source;
70      wire [17:0] mult_Ain;
71      wire [17:0] mult_Bin;
72      wire [17:0] B1_source;
73      wire [47:0] C_port;
74      wire [48:0] post_adder_out;
75      wire [7:0]  OP;
76      wire [17:0] B_used;
77      wire [47:0] X, Z;
78
```

```
//sequential operations
always @(posedge clk) begin
    if (rstA) begin
        A0 <= 0;
        A1 <= 0;
    end
    else if (CEA) begin
            A0 <= A;
            A1 <= A0;
    end
    if (rstB) begin
        B0 <= 0;
        B1 <= 0;
    end
    else if(CEB) begin
        B0 <= B_used;
        B1 <= B1_source;
    end
```

```
if (rstC)
    C0 <= 0;
else if(CEC)
        C0 <= C;

if (rstD)
    D0 <= 0;
else if(CED)
        D0 <= D;

if (rstCARRY) begin
    CYO <= 0;
    CYI <= 0;
end
else if(CECIN)begin
    CYI <= carryIN_cascade;
    CYO <= COUT;
end
```

```
    if (rstM)
        M0 <= 0;
    else if(CEM)
        M0 <= multiplier_out;

    if (rstP)
        P0 <= 0;
    else if (CEP)
        P0 <= post_adder_out[47:0];

    if (rstOP)
        OP0 <= 0;
    else if (CEOP)
        OP0 <= OPMODE;
end
```

```
// *** PIPELINE STAGES ***

//first
assign pre_adder_Din = (D_REG)  ? D0 : D;
assign pre_adder_Bin = (B0_REG)  ? B0 : B_used;
assign A1_source     = (A0_REG) ? A0 : A;
assign C_port        = (C_REG)  ? C0 : C;

//second
assign mult_Ain      = (A1_REG) ? A1 : A1_source;
assign mult_Bin      = (A1_REG) ? B1 : B1_source;

//third
assign M = (M_REG) ? M0 : multiplier_out;
assign CIN           = (CIN_REG)  ? CYI : carryIN_cascade;

//forth
assign P = (P_REG) ? P0 : post_adder_out[47:0];
assign CARRYOUT = (COUT_REG) ? CYO : COUT;
```

```
// *** Logic ***
assign OP = (OP_REG) ? OP0 : OPMODE;

assign B_used = (B_INPUT == "DIRECT") ? B :
                (B_INPUT == "CASCADE") ? BCIN :
                0;

assign pre_adder_out = (OP[6]) ? (pre_adder_Din - pre_adder_Bin) :
                       (pre_adder_Din + pre_adder_Bin);

assign B1_source = (OP[4]) ? pre_adder_out : pre_adder_Bin;

assign carryIN_cascade = (CIN_SEL == "OPMODE5") ? OP[5] :
                         (CIN_SEL == "CARRYIN") ? CARRYIN :
                         0;

assign X =  (OP[1:0] == 2'd1) ? {{12{1'b0}} ,M} :
            (OP[1:0] == 2'd2) ? P :
            (OP[1:0] == 2'd3) ? {pre_adder_Din[11:0], mult_Ain[17:0], mult_Bin[17:0]} : {48{1'b0}};

assign Z =  (OP[3:2] == 2'd1) ? PCIN :
            (OP[3:2] == 2'd2) ? P :
            (OP[3:2] == 2'd3) ? C_port:
            {48{1'b0}};

assign multiplier_out = mult_Ain * mult_Bin;

assign post_adder_out = (OP[7]) ? ( Z - X - CIN) : (X + Z + CIN);

assign PCOUT = P;

assign BCOUT = (B1_REG)? B1 : B1_source;

assign COUT = post_adder_out[48];

assign CARRYOUTF = CARRYOUT;

endmodule
```

- **Testbench**

```
module DSP48A1_tb();
  //stimulus declaration and module instantiation
    reg   [17:0]  A;
    reg   [17:0]  B;
    reg   [47:0]  C;
    reg   [17:0]  D;
    reg           clk;
    reg           CARRYIN;
    reg   [7:0]   OPMODE;
    reg   [17:0]  BCIN;
    reg           rstA;
    reg           rstB;
    reg           rstC;
    reg           rstD;
    reg           rstP;
    reg           rstM;
    reg           rstCARRY;
    reg           rstOP;
    reg           CEA;
    reg           CEB;
    reg           CEC;
    reg           CED;
    reg           CEM;
    reg           CEP;
    reg           CECIN;
    reg           CEOP;
    reg   [47:0]  PCIN;
    reg [17:0] BCOUT_exp;
    reg [47:0] PCOUT_exp;
    reg [47:0] P_exp;
    reg [35:0] M_exp;
    reg CARRYOUT_exp;
    reg CARRYOUTF_exp;

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

    DSP48A1 dut (A, B, C, D, clk, CARRYIN, OPMODE, BCIN,
                 rstA, rstB, rstC, rstD, rstP, rstM,
                 rstCARRY, rstOP, CEA, CEB, CEC, CED,
                 CEM, CEP, CECIN, CEOP, PCIN, BCOUT, PCOUT,
                 P, M, CARRYOUT, CARRYOUTF);

```

```
48  initial begin
49      clk = 0;
50      forever
51          #1 clk = ~clk;
52  end
53
54  initial begin
55      //check reset functionality
56      rstA = 1;
57      rstB = 1;
58      rstC = 1;
59      rstD = 1;
60      rstP = 1;
61      rstM = 1;
62      rstCARRY = 1;
63      rstOP = 1;
64
65      A = $random;
66      B = $random;
67      C = $random;
68      D = $random;
69      CARRYIN = $random;
70      OPMODE = $random;
71      BCIN = $random;
72      CEA = $random;
73      CEB = $random;
74      CEC = $random;
75      CED = $random;
76      CEM = $random;
77      CEP = $random;
78      CECIN = $random;
79      CEOP = $random;
80      CEA = $random;
81      CEB = $random;
82      CEC = $random;
83      CED = $random;
84      CEM = $random;
85      CEP = $random;
86      CECIN = $random;
87      CEOP = $random;
88      PCIN = $random;
89      BCOUT_exp = 0;
90      PCOUT_exp = 0;
91      P_exp = 0;
92      M_exp = 0;
93      CARRYOUT_exp = 0;
94      CARRYOUTF_exp = 0;
95      repeat(5) @(negedge clk);
```

```
if(BCOUT_exp != BCOUT) begin
    $display("*** BCOUT RESET ERROR ***");
    $stop;
end
if(PCOUT_exp != PCOUT) begin
    $display("*** PCOUT RESET ERROR ***");
    $stop;
end
if(P_exp != P) begin
    $display("*** P RESET ERROR ***");
    $stop;
end
if(M_exp != M) begin
    $display("*** M RESET ERROR ***");
    $stop;
end
if(CARRYOUT_exp !=  wire          CARRYOUT;
    $display("*** CARRYOUT RESET ERROR ***");
    $stop;
end
if(CARRYOUTF_exp != CARRYOUTF) begin
    $display("*** CARRYOUTF RESET ERROR ***");
    $stop;
end
$display("*** Reset function works correctly! ***");
$stop;
```

```
rstA = 0;
rstB = 0;
rstC = 0;
rstD = 0;
rstP = 0;
rstM = 0;
rstCARRY = 0;
rstOP = 0;

CEA = 1;
CEB = 1;
CEC = 1;
CED = 1;
CEM = 1;
CEP = 1;
CECIN = 1;
CEOP =1;

//check dsp path 1
OPMODE = 8'b11011101;
A = 20;
B = 10;
C = 350;
D = 25;
BCIN = $random;
PCIN = $random;
CARRYIN = $random;
BCOUT_exp = 'hf;
M_exp = 'h12c;
P_exp = 'h32;
PCOUT_exp = P_exp;
CARRYOUT_exp = 0;
CARRYOUTF_exp = CARRYOUT_exp;
repeat(4) @(negedge clk);
```

```
if(BCOUT != BCOUT_exp) begin
    $display("*** ERROR in BCOUT ***");
    $stop;
end
if(M != M_exp) begin
    $display("*** ERROR in M ***");
    $stop;
end
if(P != P) begin
    $display("*** ERROR in P ***");
    $stop;
end
if(PCOUT != PCOUT) begin
    $display("*** ERROR in PCOUT ***");
    $stop;
end
if(CARRYOUT != CARRYOUT_exp) begin
    $display("*** ERROR in CARRYOUT ***");
    $stop;
end
if(CARRYOUTF != CARRYOUTF_exp) begin
    $display("*** ERROR in CARRYOUTF ***");
    $stop;
end
$display("*** Path 1 works correctly! ***");
$stop;
```

```
//check dsp path 2
OPMODE = 8'b00010000;
BCOUT_exp = 'h23;
M_exp = 'h2bc;
P_exp = 0;
PCOUT_exp = P_exp;
CARRYOUT_exp = 0;
CARRYOUTF_exp = CARRYOUT_exp;
repeat(3) @(negedge clk);
if(BCOUT != BCOUT_exp) begin
    $display("*** ERROR in BCOUT ***");
    $stop;
end
if(M != M_exp) begin
    $display("*** ERROR in M ***");
    $stop;
end
if(P != P) begin
    $display("*** ERROR in P ***");
    $stop;
end
if(PCOUT != PCOUT) begin
    $display("*** ERROR in PCOUT ***");
    $stop;
end
if(CARRYOUT != CARRYOUT_exp) begin
    $display("*** ERROR in CARRYOUT ***");
    $stop;
end
if(CARRYOUTF != CARRYOUTF_exp) begin
    $display("*** ERROR in CARRYOUTF ***");
    $stop;
end
$display("*** Path 2 works correctly! ***");
$stop;
```

```
//check dsp path 3
OPMODE = 8'b00001010;
BCOUT_exp = 'ha;
M_exp = 'hc8;
P_exp = 0;
reg CARRYOUTF_exp;
CARRYOUTF_exp = CARRYOUT_exp;
repeat(3) @(negedge clk);
if(BCOUT != BCOUT_exp) begin
    $display("*** ERROR in BCOUT ***");
    $stop;
end
if(M != M_exp) begin
    $display("*** ERROR in M ***");
    $stop;
end
if(P != P) begin
    $display("*** ERROR in P ***");
    $stop;
end
if(PCOUT != PCOUT) begin
    $display("*** ERROR in PCOUT ***");
    $stop;
end
if(CARRYOUT != CARRYOUT_exp) begin
    $display("*** ERROR in CARRYOUT ***");
    $stop;
end
if(CARRYOUTF != CARRYOUTF_exp) begin
    $display("*** ERROR in CARRYOUTF ***");
    $stop;
end
$display("*** Path 3 works correctly! ***");
$stop;
```

```
        //check dsp path 4
        OPMODE = 8'b10100111;
        A = 5;
        B = 6;
        PCIN = 3000;
        BCOUT_exp = 'h6;
        M_exp = 'h1e;
        P_exp = 'hfe6fffec0bb1;
        PCOUT_exp = P_exp;
        CARRYOUT_exp = 1;
        CARRYOUTF_exp = CARRYOUT_exp;
        repeat(3) @(negedge clk);
        if(BCOUT != BCOUT_exp) begin
            $display("*** ERROR in BCOUT ***");
            $stop;
        end
        if(M != M_exp) begin
            $display("*** ERROR in M ***");
            $stop;
        end
        if(P != P) begin
            $display("*** ERROR in P ***");
            $stop;
        end
        if(PCOUT != PCOUT) begin
            $display("*** ERROR in PCOUT ***");
            $stop;
        end
        if(CARRYOUT != CARRYOUT_exp) begin
            $display("*** ERROR in CARRYOUT ***");
            $stop;
        end
        if(CARRYOUTF != CARRYOUTF_exp) begin
            $display("*** ERROR in CARRYOUTF ***");
            $stop;
        end
        $display("*** Path 4 works correctly! ***");
        $stop;
    end

endmodule
```

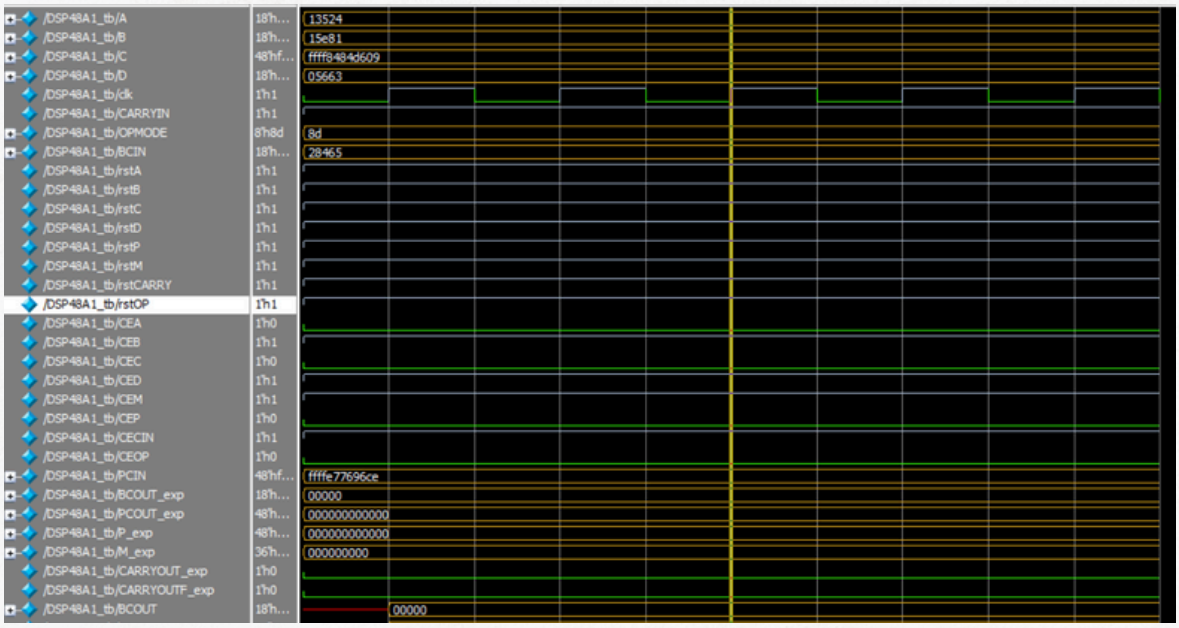
- **Do file :**

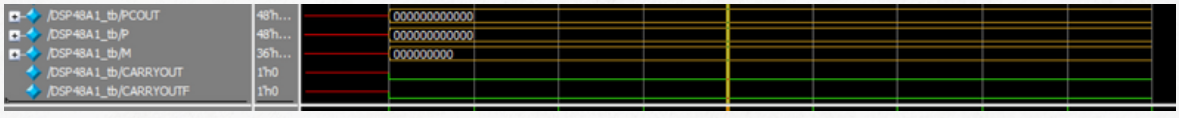
```
vlib work
vlog DSP48A1.v DSP48A1_tb.v
vsim -voptargs=+acc work.DSP48A1_tb
add wave *

run -all

#quit -sim
```

- **Questasim snippets :**
  - Reset check :

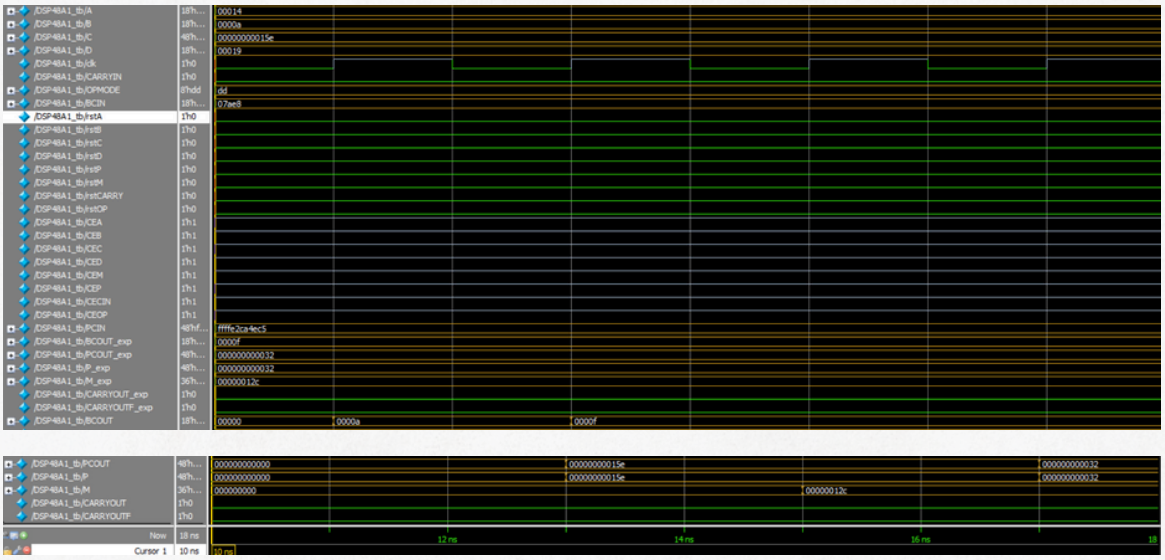



```
# *** Reset function works correctly! ***
# ** Note: $stop    : DSP48A1_tb.v(121)
#    Time: 10 ns  Iteration: 1  Instance: /DSP48A1_tb
```

## 2.2. Verify DSP Path 1

- This test path as shown in figure 1 evaluates the pre-subtractor, allowing it to propagate and post-subtractor stages by enabling the multiplier output to route through Mux X and the C-port through Mux Z, corresponding to OPMODE = 8'b11011101.

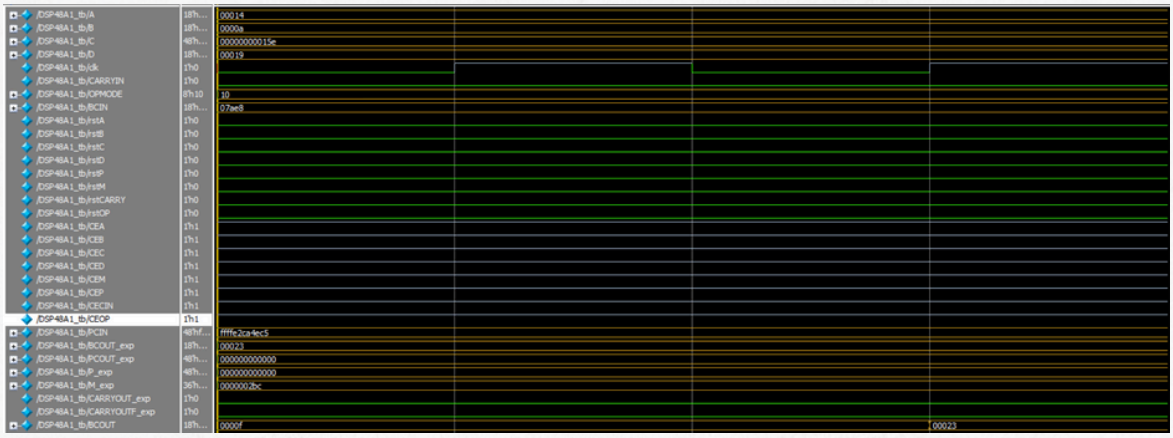


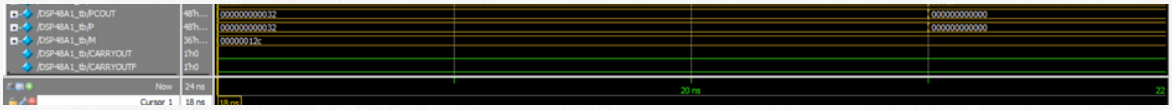


```
# *** Path 1 works correctly! ***
# ** Note: $stop    : DSP48A1_tb.v(182)
#    Time: 18 ns  Iteration: 1  Instance: /DSP48A1_tb
```

## 2.3. Verify DSP Path 2

- This test path evaluates the pre-addition allowing it to propagate and post-addition stages by enabling the zeros to route through Mux X and Mux Z, corresponding to OPMODE = 8'b00010000.

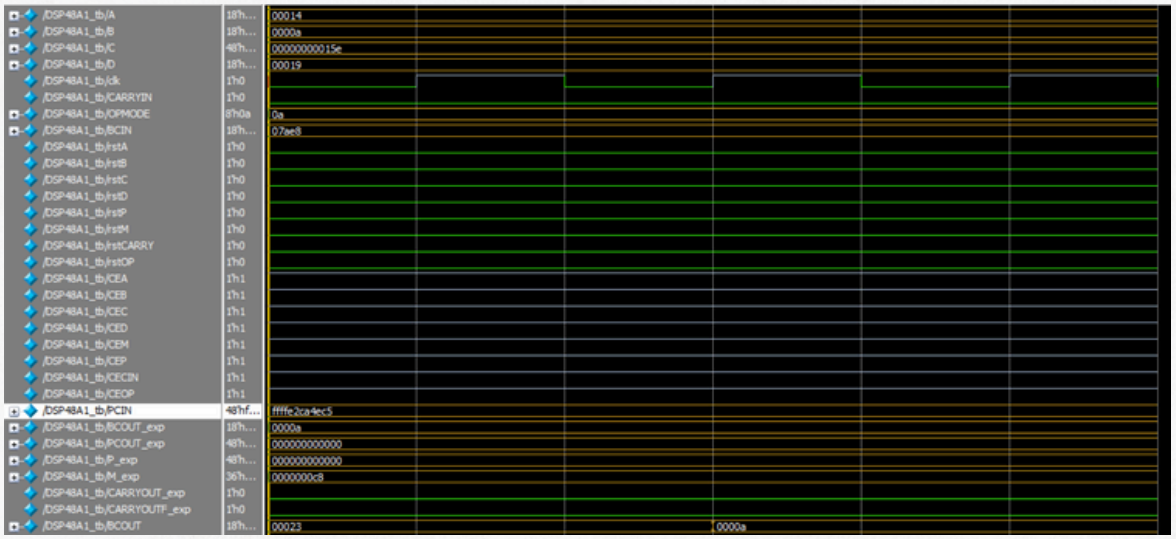


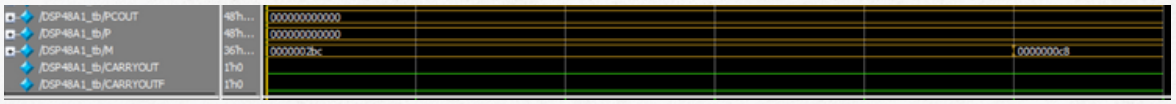


```
# *** Path 2 works correctly! ***
# ** Note: $stop    : DSP48A1_tb.v(218)
#    Time: 24 ns  Iteration: 1  Instance: /DSP48A1_tb
```

## 2.4. Verify DSP Path 3

- This test path has no pre-addition/subtractor (doesn’t allow it to propagate) and allows post-addition stages by enabling the P feedback to route through Mux X and Mux Z, corresponding to OPMODE = 8'b00001010.

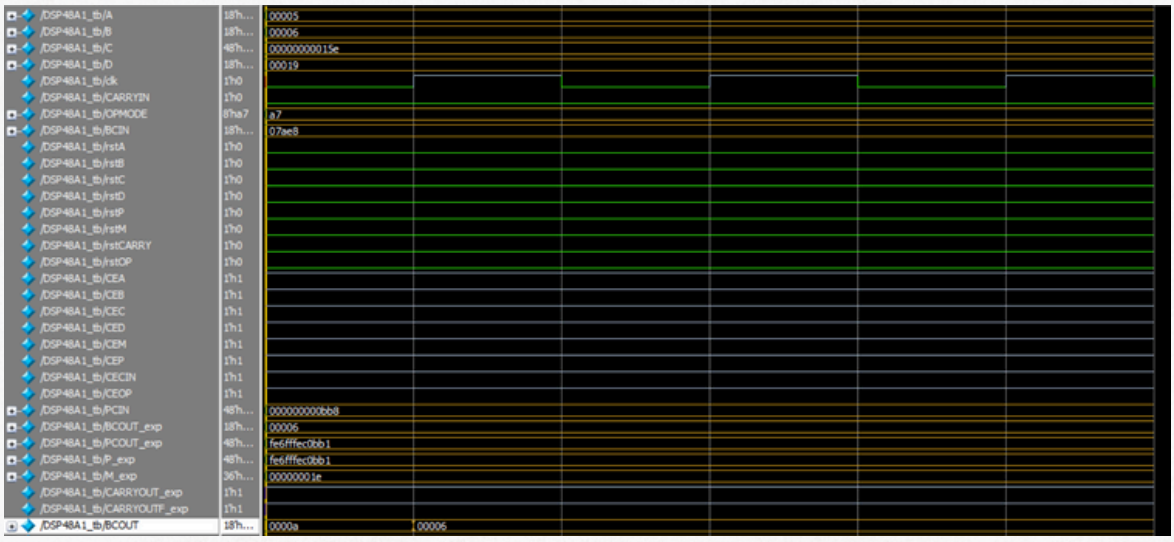


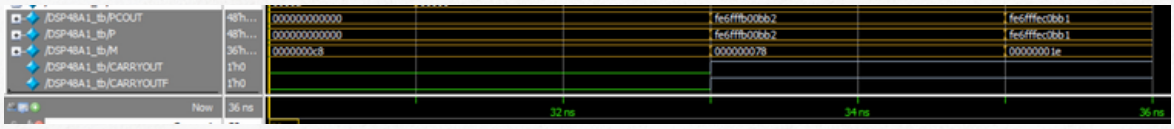




```
# *** Path 3 works correctly! ***
# ** Note: $stop    : DSP48A1_tb.v(254)
#    Time: 30 ns  Iteration: 1  Instance: /DSP48A1_tb
```

## 2.5. Verify DSP Path 4

- This test path has no pre-addition/subtractor (doesn't allow it to propagate) and allows post-subtractor stages by enabling D:A:B Concatenated to route through Mux X and PCIN to Mux Z, corresponding to OPMODE = 8'b10100111.



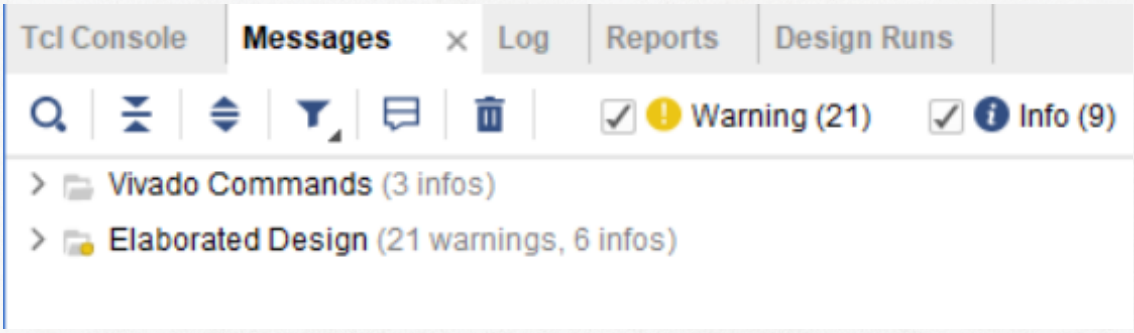




```
# *** Path 3 works correctly! ***
# ** Note: $stop    : DSP48A1_tb.v(254)
#    Time: 30 ns  Iteration: 1  Instance: /DSP48A1_tb
```

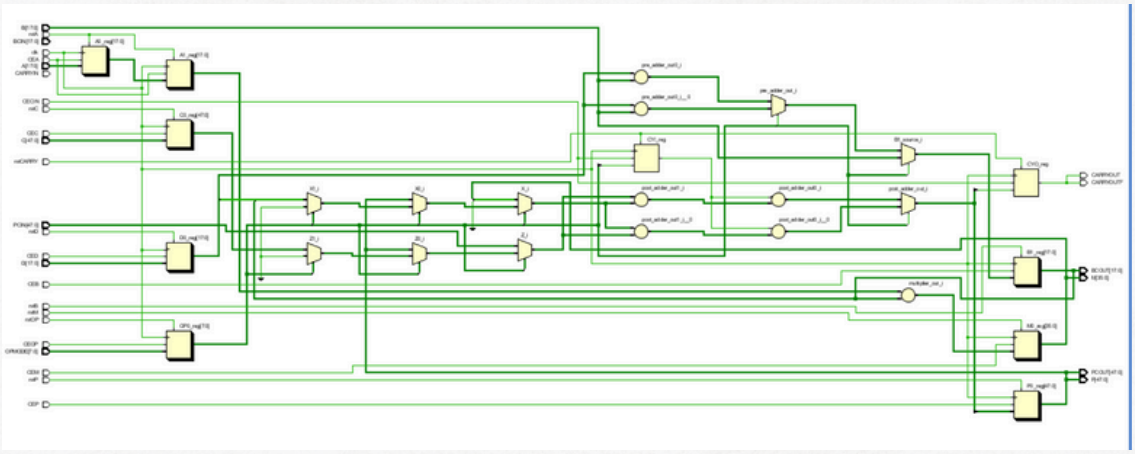
- **Constraints file : (**Timing constraints only are set**)**

```
## 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]
```

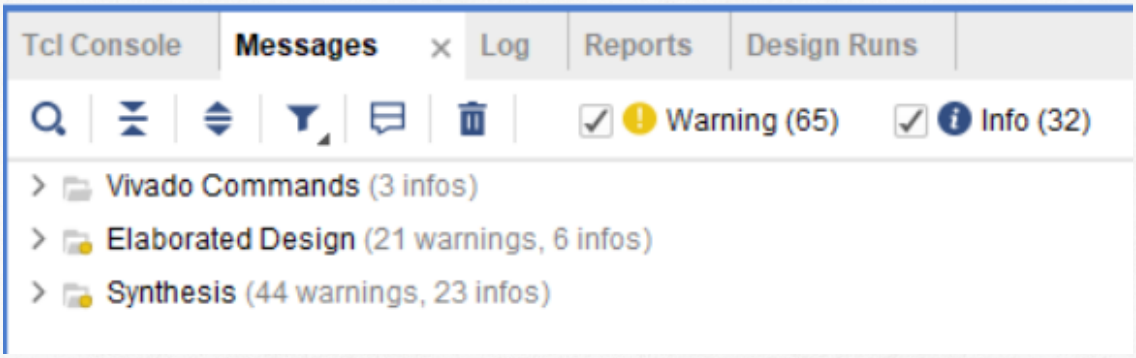
- **Elaporation stage :**
  - messages :



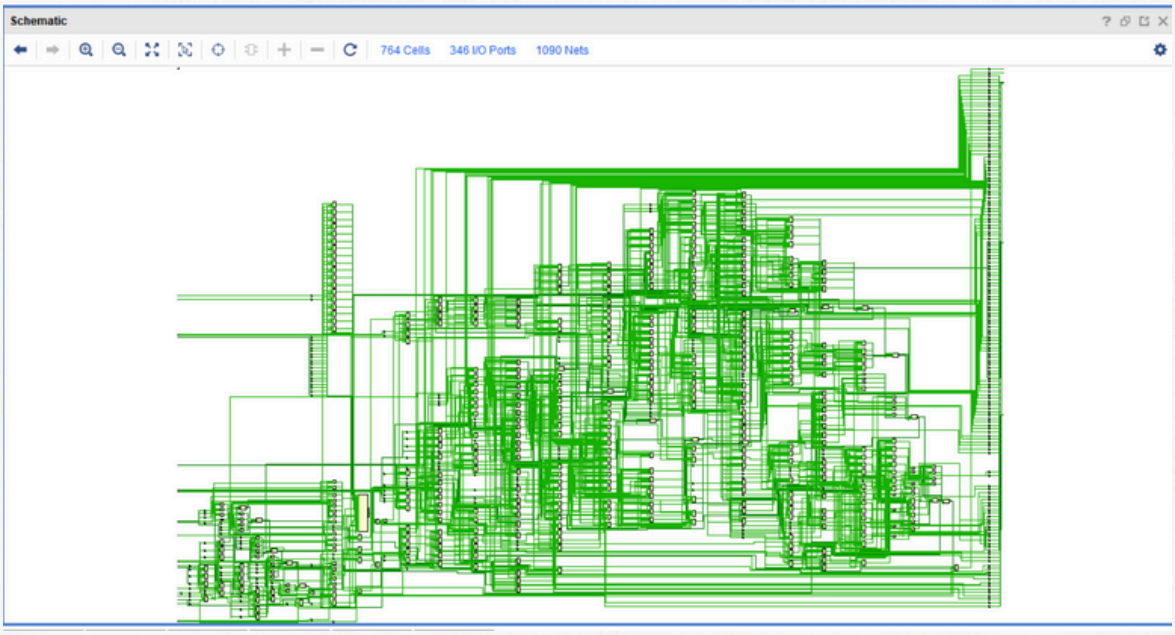

  - RTL schematic :



- **Synthesis stage :**
  - messages :

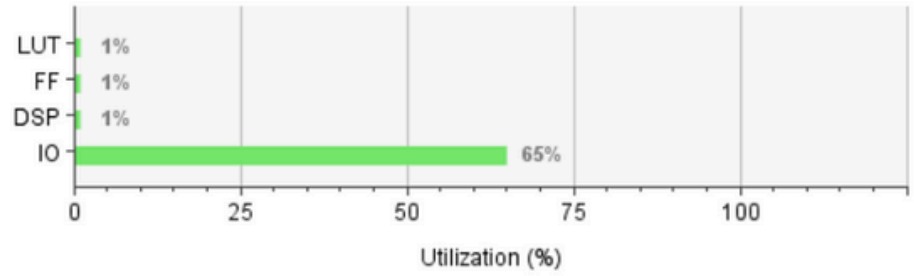



  - technology schematic :



- **Synthesis stage :**
  - utilization report :

| Resource | Utilization | Available | Utilization % |
|---|---|---|---|
| LUT | 227 | 133800 | 0.17 |
| FF | 197 | 267600 | 0.07 |
| DSP | 1 | 740 | 0.14 |
| IO | 327 | 500 | 65.40 |



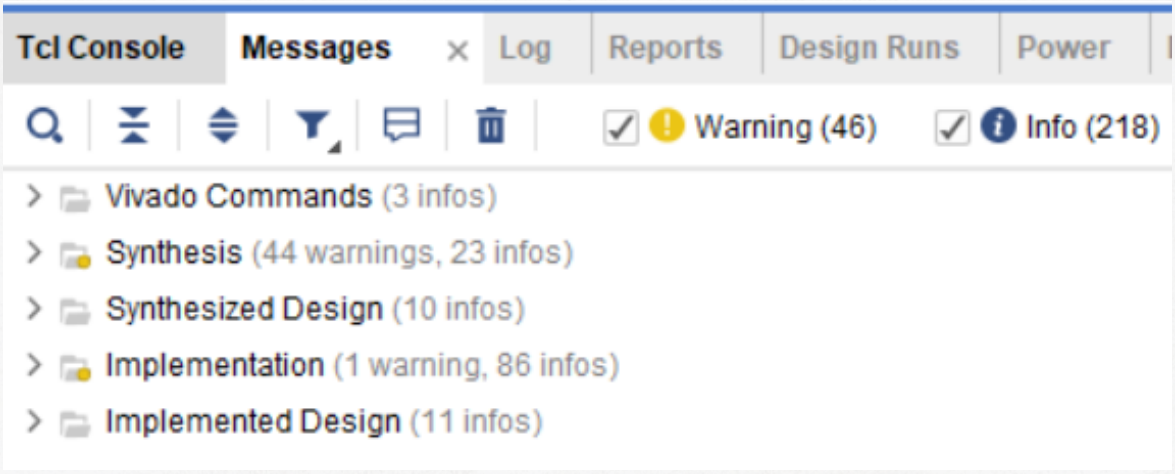

  - timing report :

**Design Timing Summary**

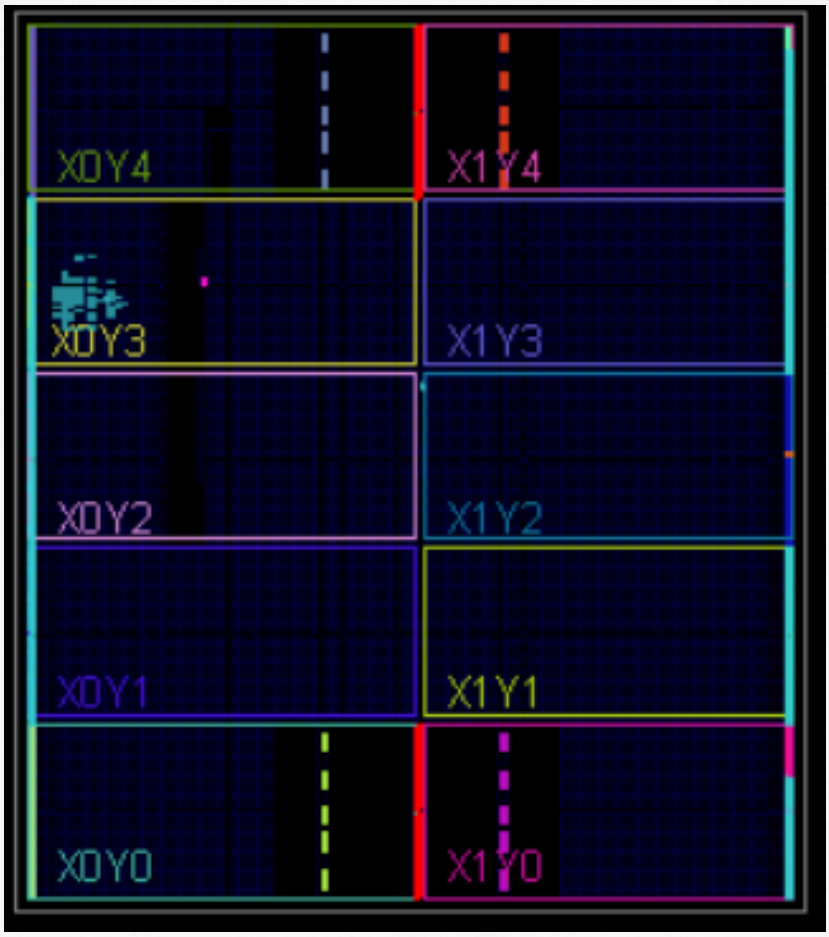
| Setup | | Hold | | Pulse Width | |
|---|---|---|---|---|---|
| Worst Negative Slack (WNS): | 5.547 ns | Worst Hold Slack (WHS): | 0.147 ns | Worst Pulse Width Slack (WPWS): | 4.500 ns |
| Total Negative Slack (TNS): | 0.000 ns | Total Hold Slack (THS): | 0.000 ns | Total Pulse Width Negative Slack (TPWS): | 0.000 ns |
| Number of Failing Endpoints: | 0 | Number of Failing Endpoints: | 0 | Number of Failing Endpoints: | 0 |
| Total Number of Endpoints: | 123 | Total Number of Endpoints: | 123 | Total Number of Endpoints: | 180 |

**All user specified timing constraints are met.**

- **Implementation stage :**
  - messages :

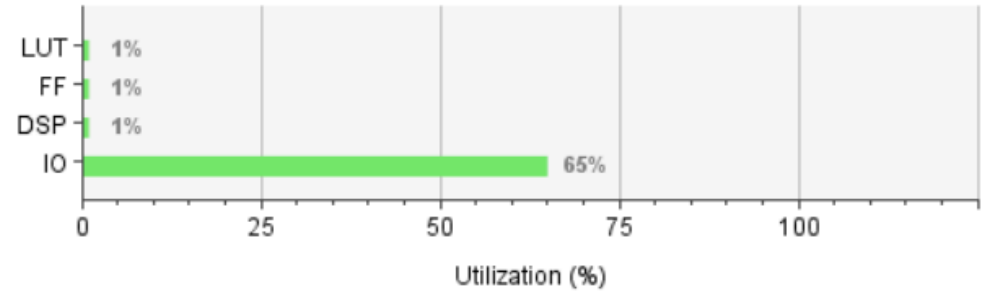



  - technology schematic :



- **Implementation stage :**
  - utilization report :

| Resource | Utilization | Available | Utilization % |
|---|---|---|---|
| LUT | 227 | 133800 | 0.17 |
| FF | 197 | 267600 | 0.07 |
| DSP | 1 | 740 | 0.14 |
| IO | 327 | 500 | 65.40 |




  - timing report :

**Design Timing Summary**

| Setup | | Hold | | Pulse Width | |
|---|---|---|---|---|---|
| Worst Negative Slack (WNS): | 3.462 ns | Worst Hold Slack (WHS): | 0.129 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: | 142 | Total Number of Endpoints: | 142 | Total Number of Endpoints: | 199 |

**All user specified timing constraints are met.**

- **Lint result :**

| Severity | Status | Check |
|---|---|---|
| | ? | condition_const |
| | ? | condition_const |
| | ? | condition_const |
| | ? | condition_const |
| | ? | condition_const |
| M | ? | condition_const |
| | ? | condition_const |
| | ? | condition_const |
| | ? | condition_const |
| | ? | condition_const |
| | ? | condition_const |
| | ? | condition_const |
| | ? | condition_const |
| | ? | always_signal_assign_large |