**Spartan6**

**DSP48A1**

Ameer Ashraf Louly – G3

1. RTL Code:

1.1 Mux Reg Module:

module mux\_reg(in, out, clk, ce, rst);

    parameter WIDTH = 18;

    parameter RSTTYPE = "SYNC";

    parameter REG = 0;

    input [WIDTH - 1 : 0] in;

    input   clk,

            ce,

            rst;

    output [WIDTH - 1 : 0] out;

    reg [WIDTH - 1 : 0] in\_reg;

    generate

        if(REG == 1) begin

            if(RSTTYPE == "SYNC") begin

            always @(posedge clk) begin

                    if(rst)

                        in\_reg <= 0;

                    else if(ce)

                        in\_reg <= in;

                end

            end

            else if (RSTTYPE == "ASYNC") begin

                always @(posedge clk or posedge rst) begin

                    if(rst)

                        in\_reg <= 0;

                    else if(ce)

                        in\_reg <= in;

                end

            end

            assign out = in\_reg;

        end

        else if (REG == 0)

           assign out = in;

    endgenerate

endmodule

* 1. DSP RTL:

module dsp48a1 #(

    parameter   A0REG = 0,

                A1REG = 1,

                B0REG = 0,

                B1REG = 1,

                CREG = 1,

                DREG = 1,

                MREG = 1,

                PREG = 1,

                CARRYINREG = 1,

                CARRYOUTREG = 1,

                OPMODEREG = 1,

                CARRYINSEL = "OPMODE5",

                B\_INPUT = "DIRECT",

                RSTTYPE = "SYNC"

) (

    input [17 : 0]  A,

                    B,

                    D,

                    BCIN,

    input [7 : 0]   OPMODE,

    input [47 : 0]  C,

                    PCIN,

    input   CLK,

            CARRYIN,

            RSTA,

            RSTB,

            RSTM,

            RSTP,

            RSTC,

            RSTD,

            RSTCARRYIN,

            RSTOPMODE,

            CEA,

            CEB,

            CEM,

            CEP,

            CEC,

            CED,

            CECARRYIN,

            CEOPMODE,

    output  [17 : 0]    BCOUT,

    output  [47 : 0]    PCOUT,

                        P,

    output  [35 : 0]    M,

    output  CARRYOUT,

            CARRYOUTF

);

    wire [17 : 0]   A0\_out,

                    D\_out,

                    B0\_in,

                    B0\_out,

                    B1\_in,

                    A1\_out,

                    B1\_out;

    reg [17 : 0]    pre\_adder\_subtractor\_out;

    reg [47 : 0]    x\_out,

                    z\_out;

    wire [47 : 0]   C\_out,

                    post\_adder\_subtractor\_out,

                    P\_in;

    wire [35 : 0]   multiplier\_out,

                    M\_out;

    wire [7 : 0]    opmode\_out;

    wire    CYI\_in,

            CYI\_out,

            CYO\_in;

    // Input Stage

    mux\_reg #(  .REG(A0REG),

                .RSTTYPE(RSTTYPE),

                .WIDTH(18)) A0\_REG (

                    .in(A),

                    .out(A0\_out),

                    .clk(CLK),

                    .ce(CEA),

                    .rst(RSTA)

                );

    mux\_reg #(  .REG(DREG),

                .RSTTYPE(RSTTYPE),

                .WIDTH(18)) D\_REG (

                    .in(D),

                    .out(D\_out),

                    .clk(CLK),

                    .ce(CED),

                    .rst(RSTD)

                );

    // assign B0\_in = B;

    generate

        if(B\_INPUT == "DIRECT") begin

            assign B0\_in = B;

        end

        else if (B\_INPUT == "CASCADE") begin

            assign B0\_in = BCIN;

        end

        else begin

            assign B0\_in = 0;

        end

    endgenerate

    mux\_reg #(  .REG(B0REG),

                .RSTTYPE(RSTTYPE),

                .WIDTH(18)) B0\_REG (

                    .in(B0\_in),

                    .out(B0\_out),

                    .clk(CLK),

                    .ce(CEB),

                    .rst(RSTB)

                );

    mux\_reg #(  .REG(OPMODEREG),

                .RSTTYPE(RSTTYPE),

                .WIDTH(8)) OPMODE\_REG (

                    .in(OPMODE),

                    .out(opmode\_out),

                    .clk(CLK),

                    .ce(CEOPMODE),

                    .rst(RSTOPMODE)

                );

    mux\_reg #(  .REG(CREG),

                .RSTTYPE(RSTTYPE),

                .WIDTH(48)) C\_REG (

                    .in(C),

                    .out(C\_out),

                    .clk(CLK),

                    .ce(CEC),

                    .rst(RSTC)

                );

    // Pre Adder Subtractor

    always @(\*) begin

        if(opmode\_out[6])

            pre\_adder\_subtractor\_out = D\_out - B0\_out;

        else

            pre\_adder\_subtractor\_out = D\_out + B0\_out;

    end

    assign B1\_in = (opmode\_out[4]) ? pre\_adder\_subtractor\_out : B0\_out;

    mux\_reg #(  .REG(A1REG),

                .RSTTYPE(RSTTYPE),

                .WIDTH(18)) A1\_REG (

                    .in(A0\_out),

                    .out(A1\_out),

                    .clk(CLK),

                    .ce(CEA),

                    .rst(RSTA)

                );

    mux\_reg #(  .REG(B1REG),

                .RSTTYPE(RSTTYPE),

                .WIDTH(18)) B1\_REG (

                    .in(B1\_in),

                    .out(B1\_out),

                    .clk(CLK),

                    .ce(CEB),

                    .rst(RSTB)

                );

    assign BCOUT = B1\_out; //! Might Need to Change Later

    assign multiplier\_out = B1\_out \* A1\_out;

    mux\_reg #(  .REG(MREG),

                .RSTTYPE(RSTTYPE),

                .WIDTH(36)) M\_REG (

                    .in(multiplier\_out),

                    .out(M\_out),

                    .clk(CLK),

                    .ce(CEM),

                    .rst(RSTM)

                );

    assign M = M\_out;

    //\* Mux X

    always @(\*) begin

        case(opmode\_out[1 : 0])

            2'b00: x\_out = 0;

            2'b01: x\_out = {12'b000000000000, M\_out};

            2'b10: x\_out = PCOUT;

            2'b11: x\_out = {D\_out[11:0], A1\_out[17:0], B1\_out[17:0]};

        endcase

    end

    //\* Mux Z

    always @(\*) begin

        case(opmode\_out[3 : 2])

            2'b00: z\_out = 0;

            2'b01: z\_out = PCIN;

            2'b10: z\_out = PCOUT;

            2'b11: z\_out = C\_out;

        endcase

    end

    generate

        if(CARRYINSEL == "CARRYIN")

            assign CYI\_in = CARRYIN;

        else if(CARRYINSEL == "OPMODE5")

            assign CYI\_in = opmode\_out[5];

    endgenerate

    mux\_reg #(  .REG(CARRYINREG),

                .RSTTYPE(RSTTYPE),

                .WIDTH(1)) CYI\_REG (

                    .in(CYI\_in),

                    .out(CYI\_out),

                    .clk(CLK),

                    .ce(CECARRYIN),

                    .rst(RSTCARRYIN)

                );

    assign {CYO\_in, P\_in} = (opmode\_out[7]) ?

                            z\_out - (x\_out + CYI\_out) :

                            x\_out + z\_out + CYI\_out;

    mux\_reg #(  .REG(PREG),

                .RSTTYPE(RSTTYPE),

                .WIDTH(48)) P\_REG (

                    .in(P\_in),

                    .out(P),

                    .clk(CLK),

                    .ce(CEP),

                    .rst(RSTP)

                );

    assign PCOUT = P;

    mux\_reg #(  .REG(CARRYOUTREG),

                .RSTTYPE(RSTTYPE),

                .WIDTH(1)) CYO\_REG (

                    .in(CYO\_in),

                    .out(CARRYOUT),

                    .clk(CLK),

                    .ce(CECARRYIN),

                    .rst(RSTCARRYIN)

                );

    assign CARRYOUTF = CARRYOUT;

endmodule

1. Testbench Code:

module dsp48a1\_tb();

    reg [17 : 0]    A,

                    B,

                    D,

                    BCIN;

    reg [7 : 0]     OPMODE;

    reg [47 : 0]    C,

                    PCIN;

    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;

    dsp48a1 DUT(.A(A),

                .B(B),

                .D(D),

                .BCIN(BCIN),

                .OPMODE(OPMODE),

                .C(C),

                .PCIN(PCIN),

                .CLK(CLK),

                .CARRYIN(CARRYIN),

                .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),

                .BCOUT(BCOUT),

                .PCOUT(PCOUT),

                .P(P),

                .M(M),

                .CARRYOUT(CARRYOUT),

                .CARRYOUTF(CARRYOUTF));

    initial begin

        CLK = 0;

        forever begin

            #1 CLK = ~CLK;

        end

    end

    initial begin

        // Testing Reset

        RSTA = 1;

        RSTB = 1;

        RSTM = 1;

        RSTP = 1;

        RSTC = 1;

        RSTD = 1;

        RSTCARRYIN = 1;

        RSTOPMODE = 1;

        repeat(100) begin

            A = $random;

            B = $random;

            D = $random;

            BCIN = $random;

            OPMODE = $random;

            C = $random;

            PCIN = $random;

            CLK = $random;

            CARRYIN = $random;

            CEA = $random;

            CEB = $random;

            CEM = $random;

            CEP = $random;

            CEC = $random;

            CED = $random;

            CECARRYIN = $random;

            CEOPMODE = $random;

            @(negedge CLK);

            if (BCOUT     != 0 ||

                PCOUT     != 0 ||

                P         != 0 ||

                M         != 0 ||

                CARRYOUT  != 0 ||

                CARRYOUTF != 0) begin

                $display("ERROR - Reset");

                $exit;

            end

        end // End of Reset Test Loop

        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;

        OPMODE = 8'b11011101;

        A = 20;

        B = 10;

        C = 350;

        D = 25;

        repeat(100) begin

            BCIN     = $random;

            PCIN     = $random;

            CARRYIN  = $random;

            repeat(4) @(negedge CLK);

            if (BCOUT      != 'hf     &&

                M          != 'h12c   &&

                P          != 'h32    &&

                PCOUT      != 'h32    &&

                CARRYOUT   != 0       &&

                CARRYOUTF  != 0) begin

                $display("ERROR - DSP Path 1");

                $exit;

            end

        end // End of DSP Path 1 Loop

        OPMODE = 8'b00010000;

        A      = 20;

        B      = 10;

        C      = 350;

        D      = 25;

        repeat(100) begin

            BCIN     = $random;

            PCIN     = $random;

            CARRYIN  = $random;

            repeat(3) @(negedge CLK); // Wait for DREG, B1REG, MREG propagation

            if (

                BCOUT      != 'h23   &&

                M          != 'h2bc  &&

                P          != 0      &&

                PCOUT      != 0      &&

                CARRYOUT   != 0      &&

                CARRYOUTF  != 0

            )begin

                $display("ERROR - DSP Path 2");

                $exit;

            end

        end // End of DSP Path 2 Loop

        // DSP Path 3 — No Pre-addition, P Feedback through Mux X and Z

        OPMODE = 8'b00001010;

        A      = 20;

        B      = 10;

        C      = 350;

        D      = 25;

        repeat(100) begin

            BCIN     = $random;

            PCIN     = $random;

            CARRYIN  = $random;

            repeat(3) @(negedge CLK); // Wait for B1REG, MREG, PREG propagation

            if (

                BCOUT      != 'ha   &&

                M          != 'hc8  &&

                P          != PCOUT &&

                CARRYOUT   != CARRYOUTF

            ) begin

                $display("ERROR - DSP Path 3");

                $exit;

            end

        end // End of DSP Path 3 Loop

        // DSP Path 4 — Post-subtraction with D:A:B concatenation and PCIN routing

        OPMODE = 8'b10100111;

        A      = 5;

        B      = 6;

        C      = 350;

        D      = 25;

        PCIN   = 3000;

        repeat(100) begin

            BCIN     = $random;

            CARRYIN  = $random;

            repeat(3) @(negedge CLK); // Wait for B1REG, MREG, PREG propagation

            if (

                BCOUT      != 'h6                    &&

                M          != 'h1e                   &&

                P          != 'hfe6fffec0bb1         &&

                PCOUT      != 'hfe6fffec0bb1         &&

                CARRYOUT   != 1                      &&

                CARRYOUTF  != 1

            ) begin

                $display("ERROR - DSP Path 4");

                $exit;

            end

        end // End of DSP Path 4 Loop

        $display("Test Successful");

        $exit;

    end

endmodule

1. Do File:

A screenshot of a computer program

AI-generated content may be incorrect.

1. Simulation Snippet:

A screenshot of a computer

AI-generated content may be incorrect.

Figure Reset Test

A black screen with green lines

AI-generated content may be incorrect.

Figure DSP Path 1

A black and white grid with white lines

AI-generated content may be incorrect.

Figure DSP Path 2

A black and white grid with white lines

AI-generated content may be incorrect.

Figure DSP Path 3

A black screen with green lines

AI-generated content may be incorrect.

Figure DSP Path 4

1. Linting:

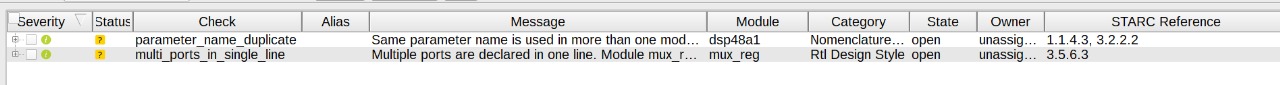


Figure Linter Showing no Critical Errors

1. Constraint File:

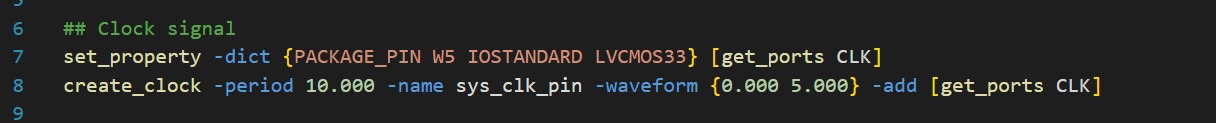


Figure Setting Clock Signal through Constraint File, Nothing else was edited

1. Vivado Steps:

7.1 Elaboration:

A screenshot of a computer

AI-generated content may be incorrect.

Figure No Critical Warning or Errors After Elaboration

A diagram of a computer

AI-generated content may be incorrect.

Figure Elaborated Schematic

* 1. Synthesis

A screenshot of a computer

AI-generated content may be incorrect.

Figure No Critical Warnings or Errors After Synthesis

A screenshot of a computer

AI-generated content may be incorrect.

Figure Synthesis Timing Report

A screenshot of a computer

AI-generated content may be incorrect.Figure Synthesis Utilization Report

A green drawing of a machine

AI-generated content may be incorrect.

Figure Synthesized Schematic

* 1. Implementation:

A screenshot of a computer

AI-generated content may be incorrect.

Figure No Cirtical Warnings or Errors After Implementation

A screenshot of a computer

AI-generated content may be incorrect.

Figure Implementation Timing Report

A screenshot of a computer

AI-generated content may be incorrect.

Figure Implementation Utilization Report

A screenshot of a game

AI-generated content may be incorrect.

Figure Implemented Device Snippet