DSP Mini Project

Codes:

Reg_Mux_module(lower module)

```
//MUX with FlipFlop
module reg_mux(data,clk,en_clk,rst,block_out);
    parameter INPUT_SIZE = 18;
    parameter USE_REG = 0;  //0:No register(combination), 1: Sequential(Pipeline)
    parameter RSTTYPE = "SYNC";
    //Input & Output Ports
    input [INPUT_SIZE-1:0]data;
    input clk,en_clk,rst;
    output [INPUT_SIZE-1:0]block_out;

    //Internal signals
    reg [INPUT_SIZE-1:0]data_reg;

    /***************Sequential****************/
    generate
        if(RSTTYPE == "ASYNC")begin
          //ASYNC
          always@(posedge clk or posedge rst)begin
            if (rst)begin
              data_reg <= 0;
            end
            else begin
              if(en_clk)
                data_reg <= data;
            end
          end
        end
        else begin
          //sync
          always@(posedge clk)begin
            if (rst)begin
              data_reg <= 0;
```

```
            end
            else begin
              if(en_clk)
                data_reg <= data;
            end
          end
        end
    endgenerate

    //Final output based on USE_REG parameter
    assign block_out  = (USE_REG) ? data_reg : data;

endmodule
```

DPS Module(Top module)

```
module
DSP_module(A,B,C,D,CARRYIN,M,P,CARRYOUT,CARRYOUTF,CLK,OPMODE,CEA,CEB,CEC,CECARRYI
N,CED,CEM,CEOPMODE,CEP,
           RSTA,RSTB,RSTC,RSTCARRYIN,RSTD,RSTM,RSTOPMODE,RSTP,BCOUT,PCIN,PCOUT,BC
IN);

    /*********************************************
    *
    *                parameters
    *
    *********************************************/
    /**these values define the number of pipeline registers in the A and B input
paths**/
    parameter A0REG = 0;
    parameter A1REG = 1;
    parameter B0REG = 0;
    parameter B1REG = 1;

    /** defines the number of pipeline stages**/
    parameter CREG = 1;
    parameter DREG = 1;
    parameter MREG = 1;
    parameter PREG = 1;
    parameter CARRYINREG = 1;
    parameter CARRYOUTREG = 1;
```

```
    parameter OPMODEREG = 1;

    /*The CARRYINSEL attribute is used in the carry cascade input,
     either the CARRYIN input will be considered or the value of opcode[5]*/
    //This attribute can be set to the string CARRYIN or OPMODE5.
    //Tie the output of the mux to 0 if none of these string values exist.
    parameter CARRYINSEL = "OPMODE5";

    /*This attribute can be set to the string DIRECT or CASCADE*/
    /*CASCADE -> (BCIN)*/
    /*Tie the output of the mux to 0 if none of these string values exist*/

    parameter B_INPUT = "DIRECT";
    // parameter B_INPUT = "CASCADE";

    /*This attribute can be set to ASYNC or SYNC*/
    parameter RSTTYPE = "SYNC";


    /*************************************************
    *
    *                Data Ports
    *
    *************************************************/
    //Input Ports
    input [17:0]A;
    input [17:0]B;
    input [47:0]C;
    input [17:0]D;
    input CARRYIN;
    input [17:0]BCIN;
    //Control Input Ports
    input CLK;
    input [7:0]OPMODE;
    //clock enable input ports
    input CEA;  /*Clock enable for the A port registers: (A0REG & A1REG)*/
    input CEB;  /*Clock enable for the B port registers: (B0REG & B1REG)*/
    input CEC;  /*Clock enable for the C port registers (CREG)*/
    input CECARRYIN;    /*Clock enable for the carry-in register (CYI) and the
carry-out register (CYO)*/
    input CED;  /*Clock enable for the D port register (DREG)*/
    input CEM;  /*Clock enable for the multiplier register (MREG)*/
    input CEOPMODE; /*Clock enable for the opmode register (OPMODEREG)*/
    input CEP;  /*Clock enable for the P output port registers (PREG = 1)*/
    //Reset Input Ports
```

```
    input RSTA; /*Reset for the A registers: (A0REG & A1REG)*/
    input RSTB; /*Reset for the B registers: (B0REG & B1REG)*/
    input RSTC; /*Reset for the C registers (CREG)*/
    input RSTCARRYIN; /*Reset for the carry-in register (CYI) and the carry-out
register (CYO)*/
    input RSTD; /*Reset for the D register (DREG)*/
    input RSTM; /*Reset for the multiplier register (MREG)*/
    input RSTOPMODE;  /*Reset for the opmode register (OPMODEREG)*/
    input RSTP; /*Reset for the P output registers (PREG = 1)*/
    //Cascade Ports
    /*Inputs*/
    input [47:0]PCIN; /*Cascade input for Port P*/
    /*outputs*/
    output [47:0]PCOUT; /*Cascade output for Port P*/
    output [17:0]BCOUT; /*Cascade output for Port B*/
    //output ports
    output [47:0]P;
    output [35:0]M;
    output CARRYOUT;
    output CARRYOUTF;

    /*************************************************************
    *
    *                    Modules Instantiations
    *
    *************************************************************/
    //Instantiation 1st stage
    wire [17:0]dwire,bwire0,bwire1,awire0,awire1;
    wire [47:0]cwire;
    reg_mux
#(.USE_REG(DREG))  D_REG(.data(D),.clk(CLK),.en_clk(CED),.rst(RSTD),.block_out(dw
ire));
    reg_mux #(.USE_REG(A0REG))
A0_REG(.data(A),.clk(CLK),.en_clk(CEA),.rst(RSTA),.block_out(awire0));
    reg_mux #(.USE_REG(CREG),.INPUT_SIZE(48))
C_REG(.data(C),.clk(CLK),.en_clk(CEC),.rst(RSTC),.block_out(cwire));

    //For B Input
    assign bwire0 = (B_INPUT == "DIRECT") ? B : BCIN;
    reg_mux #(.USE_REG(B0REG))
B0_REG(.data(bwire0),.clk(CLK),.en_clk(CEB),.rst(RSTB),.block_out(bwire1));

    //Instantiation 2nd stage
    wire [17:0]pre_add_sub_out,bd_mux_out,bwire2;
    wire [7:0]opmodewire;
```

```
    reg_mux #(.USE_REG(OPMODEREG),.INPUT_SIZE(8))
OPMODE_REG(.data(OPMODE),.clk(CLK),.en_clk(CEOPMODE),.rst(RSTOPMODE),.block_out(o
pmodewire));
    assign pre_add_sub_out = opmodewire[6] ? (dwire-bwire1): (dwire+bwire1);
    assign bd_mux_out = opmodewire[4] ? pre_add_sub_out : bwire1;
    reg_mux #(.USE_REG(B1REG))
B1_REG(.data(bd_mux_out),.clk(CLK),.en_clk(CEB),.rst(RSTB),.block_out(bwire2));
    reg_mux #(.USE_REG(A1REG))
A1_REG(.data(awire0),.clk(CLK),.en_clk(CEA),.rst(RSTA),.block_out(awire1));

    //Instantiation 3rd stage
    wire [35:0]multiplier_out,mreg_out;
    assign multiplier_out = bwire2 * awire1;
    reg_mux #(.USE_REG(MREG),.INPUT_SIZE(36))
M_REG(.data(multiplier_out),.clk(CLK),.en_clk(CEM),.rst(RSTM),.block_out(mreg_out
));

    /*For carry cascade*/
    wire carrywire0,carrywire1;

    /**Generate if condition used with constant expression only(Parameters)*/
    generate
        if(CARRYINSEL=="OPMODE5")
            assign carrywire0 = opmodewire[5];
        else if(CARRYINSEL == "CARRYIN")
            assign carrywire0 = CARRYIN;
        else
            assign carrywire0 = 0;
    endgenerate

    reg_mux #(.USE_REG(CARRYINREG),.INPUT_SIZE(1))
CYI_REG(.data(carrywire0),.clk(CLK),.en_clk(CECARRYIN),.rst(RSTCARRYIN),.block_ou
t(carrywire1));

    //X Z multiplexers
    wire [47:0]xmulti_out,zmulti_out;
    assign xmulti_out = (opmodewire[1:0]== 0) ? 0 : (opmodewire[1:0]== 1) ?
mreg_out : (opmodewire[1:0]== 2 && PREG) ? P : {D[11:0],A[17:0],B[17:0]};
    assign zmulti_out = (opmodewire[3:2]== 0) ? 0 : (opmodewire[3:2]== 1) ? PCIN
: (opmodewire[3:2]== 2 && PREG) ? P : cwire;

    //Post adder/substractor
    wire [48:0]temp_out;
    wire [47:0]post_add_sub_out;
    wire carryoutwire;
```

```
    assign temp_out = (opmodewire[7]) ? (zmulti_out - (xmulti_out + carrywire1))
: (zmulti_out + xmulti_out + carrywire1);
    assign carryoutwire = temp_out[0];
    //check if the carryin(carrywire0) equal 1,then decrement 1 from the temp and
store it in (post_add_sub_out)
    assign post_add_sub_out = (carrywire0)? temp_out - 1 : temp_out;

    /*************************************************************
    *
    *                          Outputs
    *
    *************************************************************/

    //carry out cascade
    reg_mux #(.USE_REG(CARRYOUTREG),.INPUT_SIZE(1))
CYO_REG(.data(carryoutwire),.clk(CLK),.en_clk(CECARRYIN),.rst(RSTCARRYIN),.block_
out(CARRYOUT));
    //CARRYOUTF
    assign CARRYOUTF = CARRYOUT;
    //P (final output)
    reg_mux #(.USE_REG(PREG),.INPUT_SIZE(48))
P_REG(.data(post_add_sub_out),.clk(CLK),.en_clk(CEP),.rst(RSTP),.block_out(P));
    //PCOUT
    assign PCOUT = (PREG) ? P : post_add_sub_out;
    //BCOUT
    assign BCOUT = bwire2;
    //M
    assign M =  mreg_out;
endmodule
```

Test Bench:

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

```
    reg CARRYIN,CLK;

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

    //Clock genneration
    initial begin
      CLK = 0;
      forever
        #1 CLK = ~CLK;
    end

    //Module instantiations
    DSP_module
    #(.A0REG(1),.B0REG(1),.A1REG(1), .B1REG(1), .CREG(1), .DREG(1), .MREG(1),
.PREG(1), .CARRYINREG(0), .CARRYOUTREG(0), .OPMODEREG(0))
    DUT(.A(A),.B(B),.C(C),.D(D),.CARRYIN(CARRYIN),.M(M),.P(P),
        .CARRYOUT(CARRYOUT),.CARRYOUTF(CARRYOUTF),.CLK(CLK),
        .OPMODE(OPMODE),.CEA(CEA),.CEB(CEB),.CEC(CEC),
        .CECARRYIN(CECARRYIN),.CED(CED),.CEM(CEM),.CEOPMODE(CEOPMODE),
        .CEP(CEP),.RSTA(RSTA),.RSTB(RSTB),.RSTC(RSTC),.RSTCARRYIN(RSTCARRYIN),
        .RSTD(RSTD),.RSTM(RSTM),.RSTOPMODE(RSTOPMODE),.RSTP(RSTP),.BCOUT(BCOUT),
        .PCIN(PCIN),.PCOUT(PCOUT),.BCIN(BCIN));

    //Test stimulus generation
    initial begin
        //Enable reset
        RSTA =1;RSTB =1;RSTC =1;RSTCARRYIN= 1;RSTD =1;RSTM =1;RSTOPMODE =1;RSTP
=1;
        CEA = 0;CEB = 0;CEC = 0;CECARRYIN = 0;CED = 0;CEM = 0;CEOPMODE = 0;CEP =
0;
        CARRYIN = 1'b0;
        OPMODE = 8'b0;
        BCIN = 18'd0;
        PCIN = 48'd0;
        A = 18'd0; B = 18'd0; C = 48'd0; D = 18'd0;

        // @(posedge CLK);
        @(negedge CLK);

        //Disable reset and enable clk
        RSTA =0;RSTB =0;RSTC =0;RSTCARRYIN= 0;RSTD =0;RSTM =0;RSTOPMODE =0;RSTP
=0;
```

```
        CEA = 1;CEB = 1;CEC = 1;CECARRYIN = 1;CED = 1;CEM = 1;CEOPMODE = 1;CEP =
1;

        /*****************************************
        *
        *                   Test case 1
        *
        *****************************************/
        //P = 700 -> Binput = "Direct" , P = 1100 -> Binput = "Cascaded"
        A = 18'd20; B = 18'd30; C = 48'd300; D = 18'd50;
        CARRYIN = 1'b0;
        OPMODE = 8'b01111101;
        BCIN = 18'd10;
        PCIN = 48'd1000;


        repeat(2)@(negedge CLK);    //at least one clock to appear 700

        /*****************************************
        *
        *                   Test case 2
        *
        *****************************************/
        // P = 1900 ->Binput = "Direct"
        CARRYIN = 1'b0;
        OPMODE = 8'b00011101;   //carryin = 0
        BCIN = 18'd10;
        PCIN = 48'd1000;


        repeat(3)@(negedge CLK);
        /*****************************************
        *
        *                   Test case 3
        *
        *****************************************/
        // P = 2600 ->Binput = "Direct"
        CARRYIN = 1'b0;
        OPMODE = 8'b00110101;   //opmode [3:2] = 1 >> PCIN
        BCIN = 18'd10;
        PCIN = 48'd1000;

        repeat(3)@(negedge CLK);
        /*****************************************
        *
```

```
        *                   Test case 4
        *
        ******************************************/
        // P = 1000 ->Binput = "Direct"
        CARRYIN = 1'b0;
        OPMODE = 8'b10010100;   //opmode [1:0] = 1 >> P = Z
        BCIN = 18'd10;
        PCIN = 48'd1000;

        repeat(3)@(negedge CLK);

        $stop;
    end

    initial begin
        // @(posedge CLK);
        $monitor("Opmode =%b,P = %0d,Pcout = %0d,Bcout = %0d,M =
%0d",OPMODE,P,PCOUT,BCOUT,M);
    end



endmodule
```

Do File:

vlib work

vlog reg_mux.v DSP_module.v DSP_tb.v

vsim -voptargs=+acc work.DSP_tb

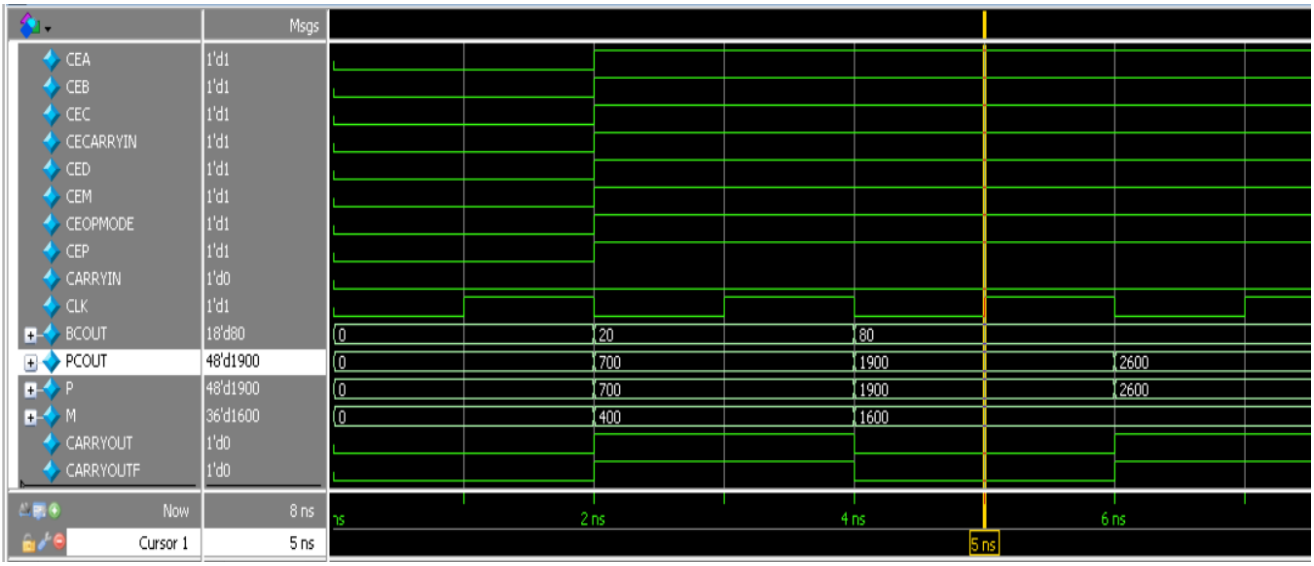
add wave *

run -all

#quit –sim

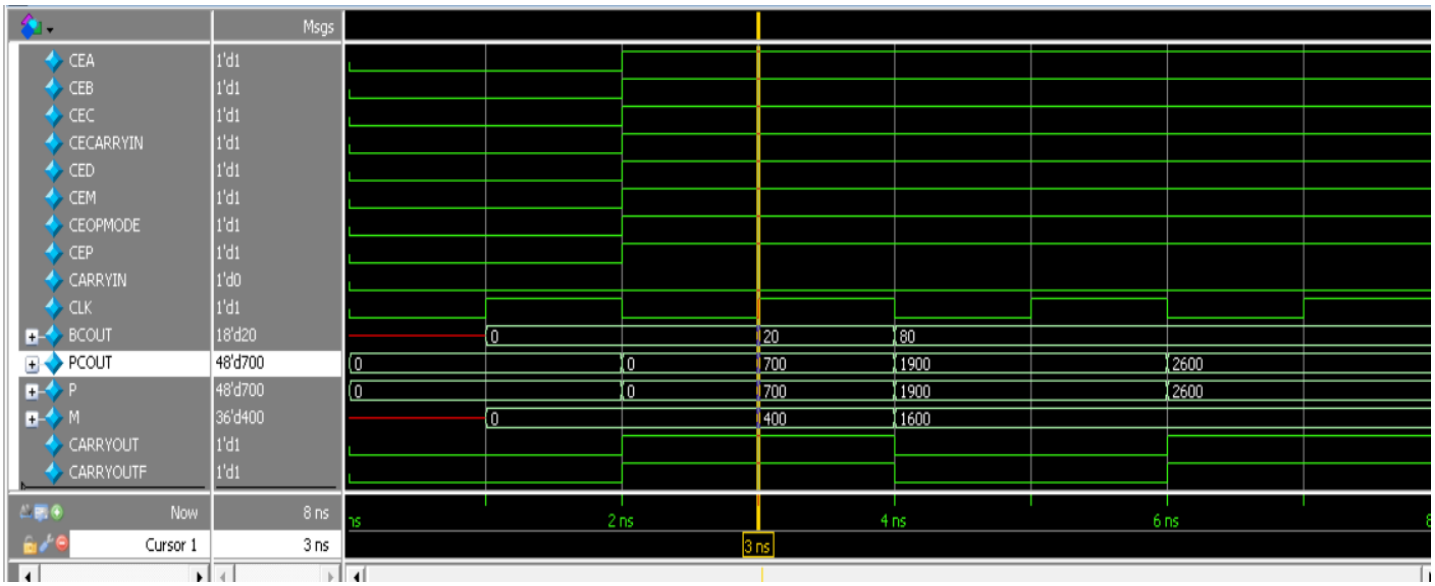
Questasim waveform:

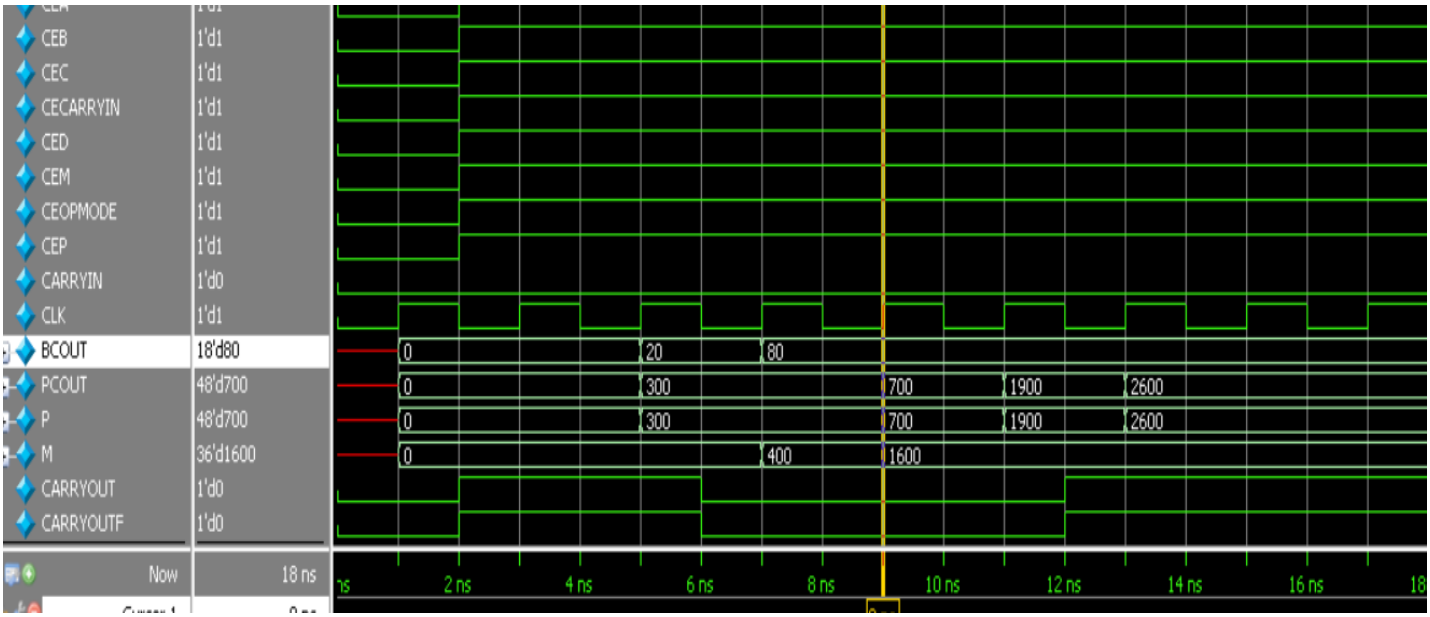
No Pipeline and RSTTYPE is “SYNC”




```
# Loading work.reg_mux(fast__2)
# Loading work.reg_mux(fast__3)
# Loading work.reg_mux(fast__4)
# Opmode =00000000,P = 0,Pcout = 0,Bcout = 0,M = 0
# Opmode =01111101,P = 700,Pcout = 700,Bcout = 20,M = 400
# Opmode =00011101,P = 1900,Pcout = 1900,Bcout = 80,M = 1600
# Opmode =00110101,P = 2600,Pcout = 2600,Bcout = 80,M = 1600
# ** Note: $stop    : DSP_tb.v(85)
#    Time: 8 ns  Iteration: 1  Instance: /DSP_tb
# Break in Module DSP_tb at DSP_tb.v line 85
```

Simulation with Pipelining:



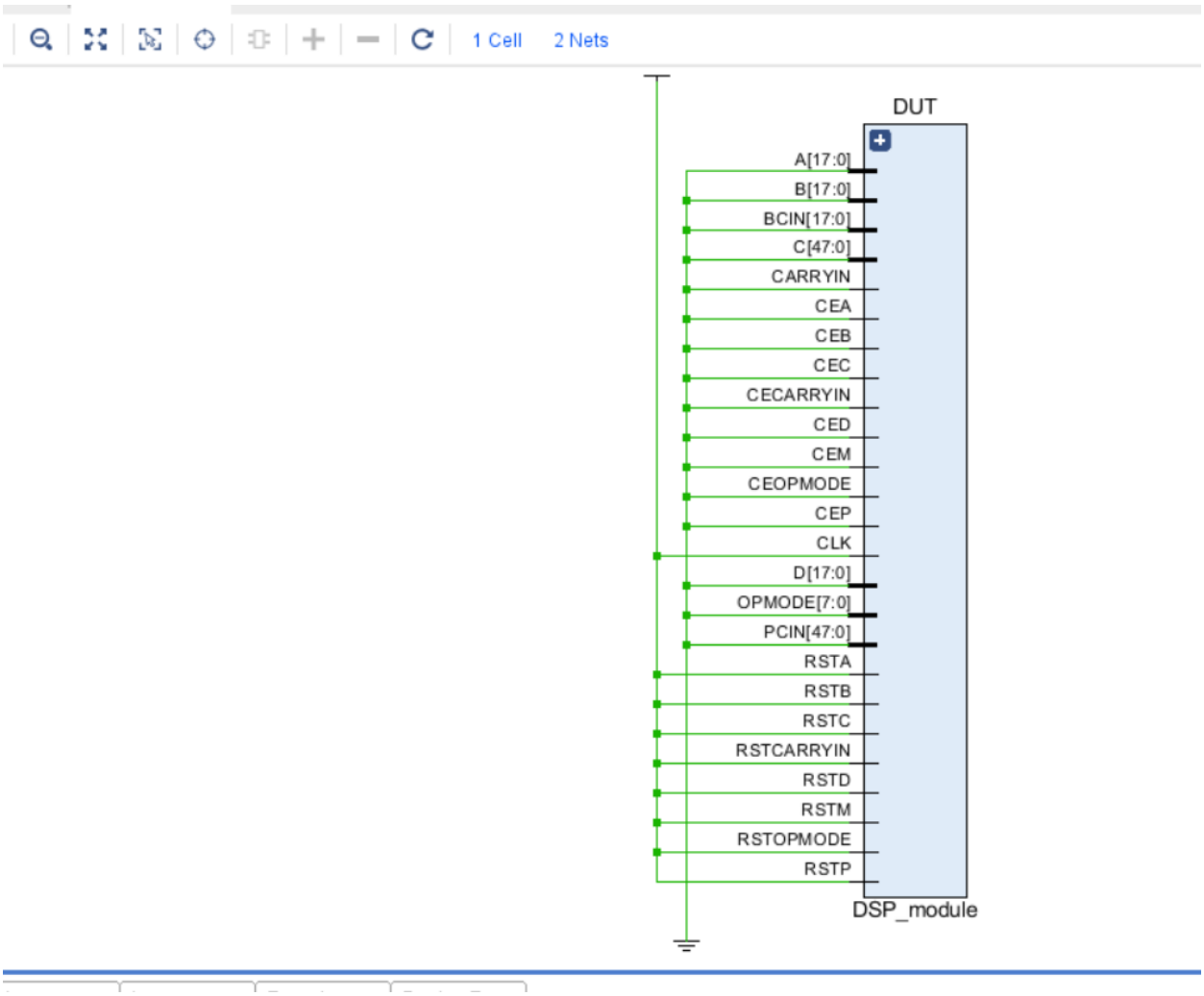


Transcript

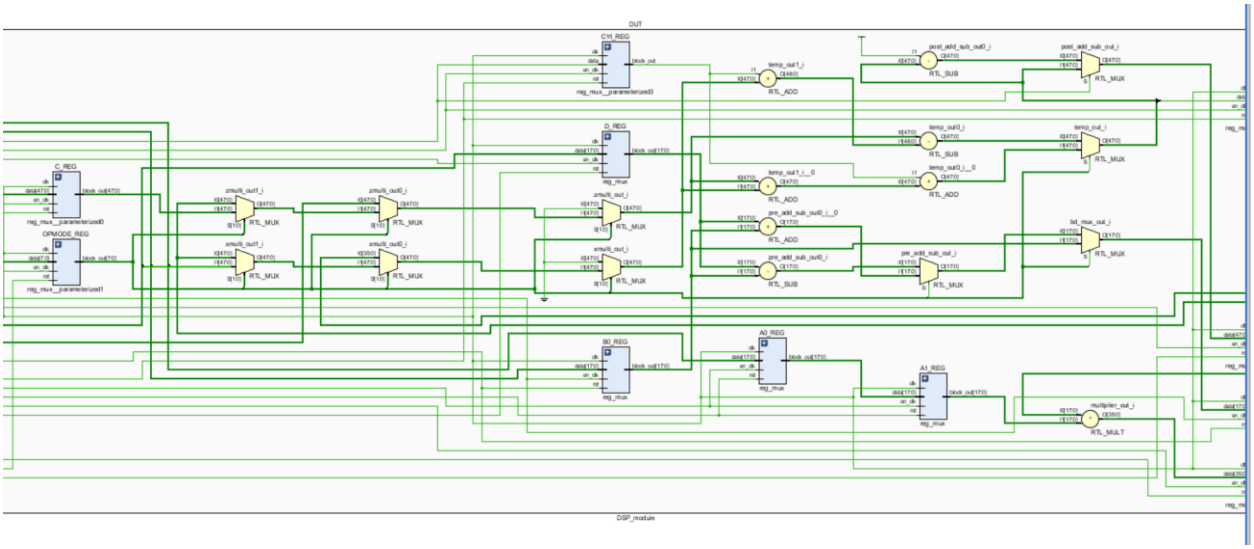
```
# Loading work.reg_mux(fast__6)
# Opmode =00000000,P = 0,Pcout = 0,Bcout = x,M = x
# Opmode =00000000,P = 0,Pcout = 0,Bcout = 0,M = 0
# Opmode =01111101,P = 0,Pcout = 0,Bcout = 0,M = 0
# Opmode =01111101,P = 700,Pcout = 700,Bcout = 20,M = 400
# Opmode =00011101,P = 1900,Pcout = 1900,Bcout = 80,M = 1600
# Opmode =00110101,P = 2600,Pcout = 2600,Bcout = 80,M = 1600
# ** Note: $stop    : DSP_tb.v(97)
#    Time: 8 ns  Iteration: 1  Instance: /DSP_tb
# Break in Module DSP_tb at DSP_tb.v line 97
```

Vivado:

Elaboration:



Schematic zoom in:



Message tap:

Tcl Console | Messages | Log | Reports | Design Runs

Critical warning (2) | Warning (26) | Info (21) | Status (11) | Show All

[IP_Flow 19-1704] No user IP repositories specified

[IP_Flow 19-2313] Loaded Vivado IP repository 'D:/Vivado/2018.2/data/ip'.

Elaborated Design (2 critical warnings, 26 warnings, 18 infos)

General Messages (2 critical warnings, 26 warnings, 18 infos)

[Synth 8-6157] synthesizing module 'DSP_tb' [DSP_tb.v:2] (6 more like this)

[Synth 8-6155] done synthesizing module 'reg_mux' (1#1) [reg_mux.v:3] (6 more like this)

[Synth 8-3331] design reg_mux__parameterized3 has unconnected port clk (23 more like this)

[Device 21-403] Loading part xc7a200tffg1156-3

[Project 1-570] Preparing netlist for logic optimization

[Common 17-55] 'set_property' expects at least one object. [Constraints_basys3.xdc:7]

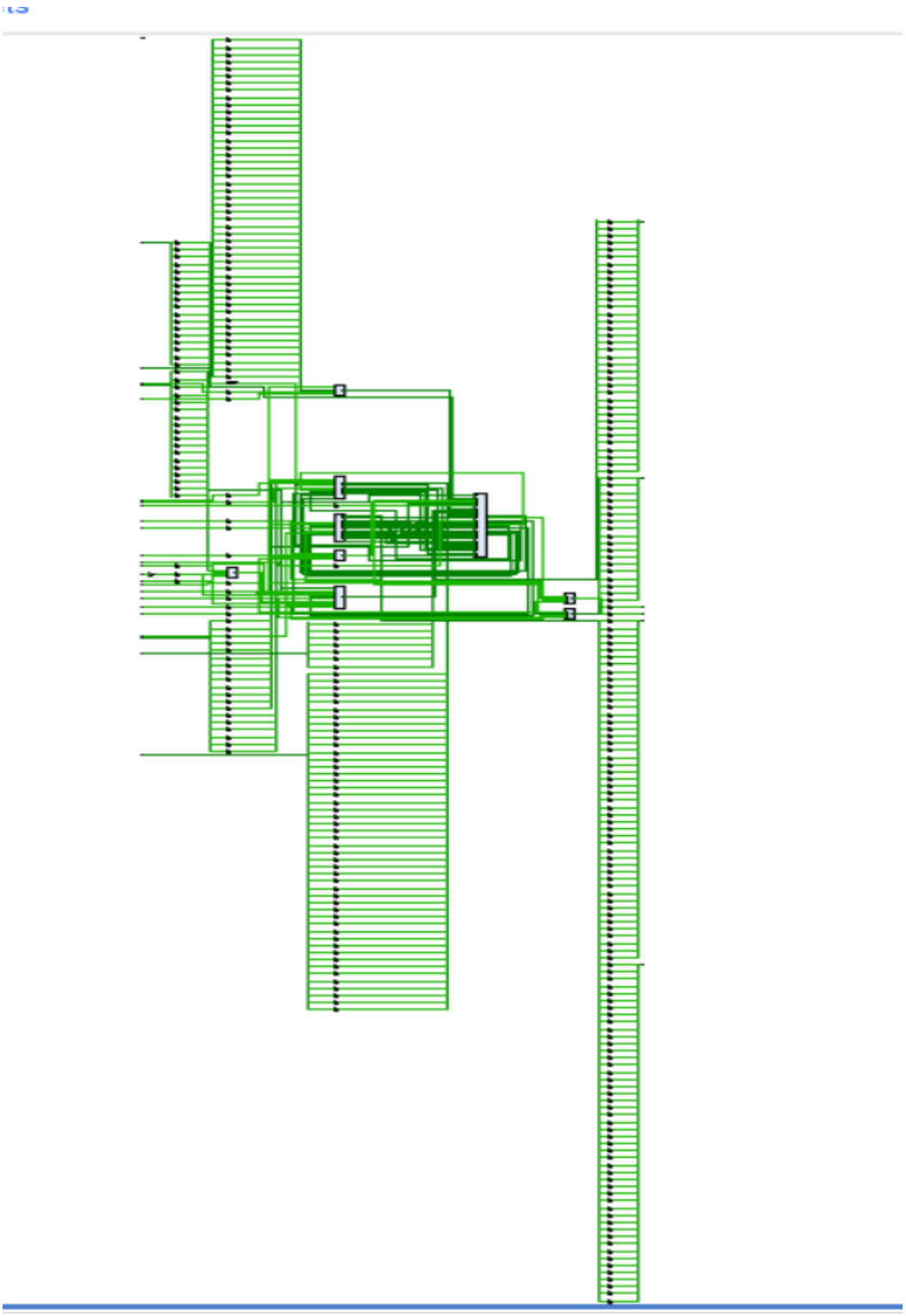
[Vivado 12-584] No ports matched 'CLK'. [Constraints_basys3.xdc:7] (1 more like this)

[Vivado 12-4739] create_clock:No valid object(s) found for '-objects [get_ports CLK]'. [Constraints_basys3.xdc:8]

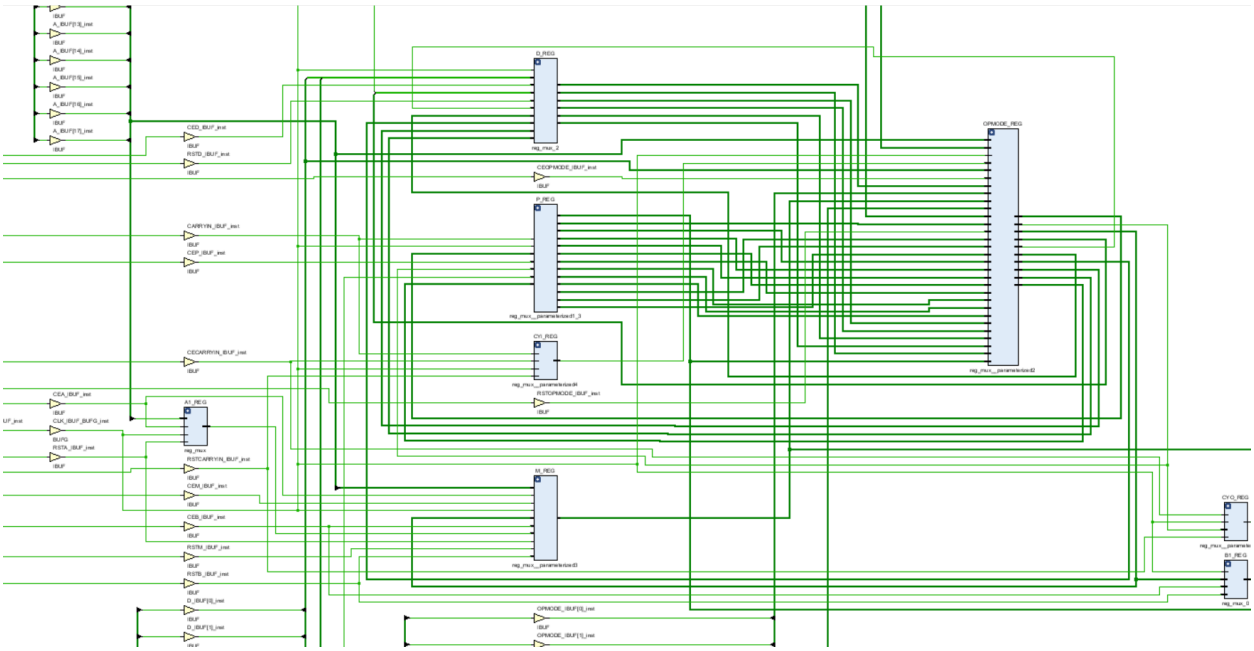
[Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).

[Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.

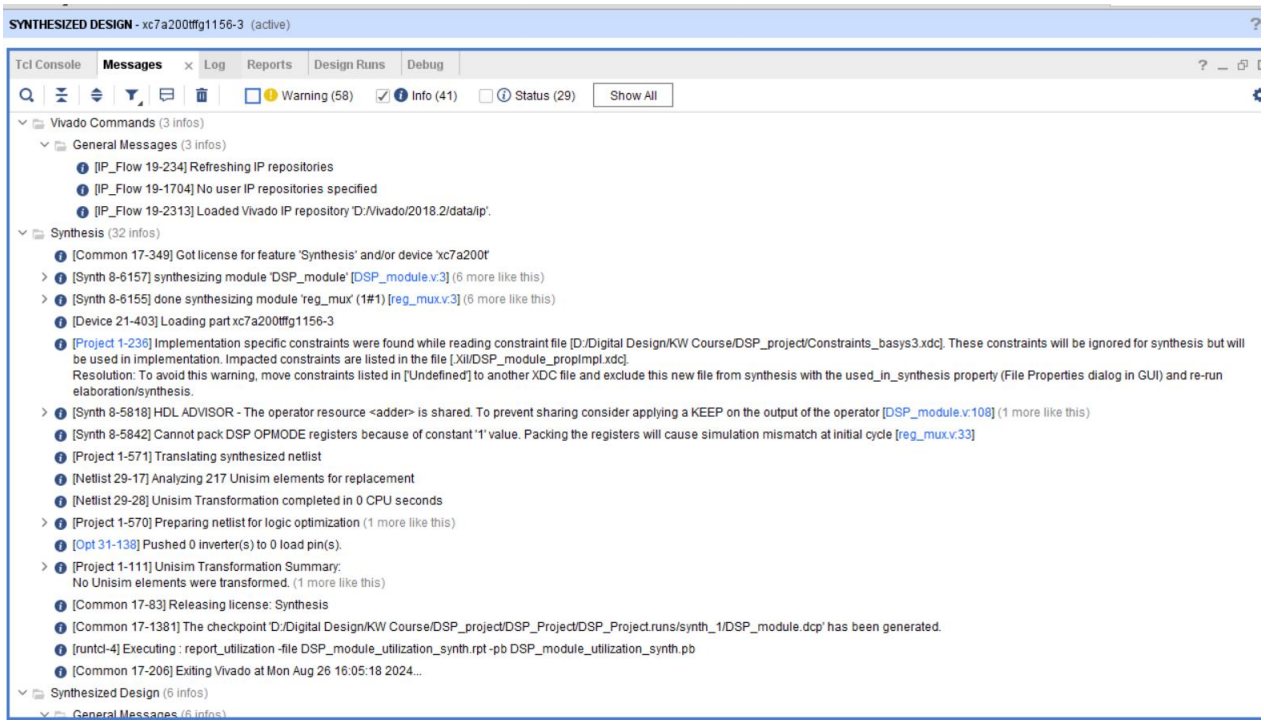
Synthesis:



Schematic zoom in:



Message tap:



Utilization report:

156-3 (active)

Reports | Design Runs | Utilization | Debug

Hierarchy

Name	Slice LUTs (134600)	Slice Registers (269200)	DSPs (740)	Bonded IOB (500)	BUFGCTRL (32)
DSP_module	264	142	1	327	1
A1_REG (reg_mux)	0	1	0	0	0
B1_REG (reg_mux_0)	0	18	0	0	0
C_REG (reg_mux__pa...	0	48	0	0	0
CYI_REG (reg_mux__...	0	1	0	0	0
CYO_REG (reg_mux_...	0	1	0	0	0
D_REG (reg_mux_2)	0	18	0	0	0
M_REG (reg_mux__pa...	0	0	1	0	0
OPMODE_REG (reg_...	215	7	0	0	0
P_REG (reg_mux__pa...	48	48	0	0	0

Timing report:

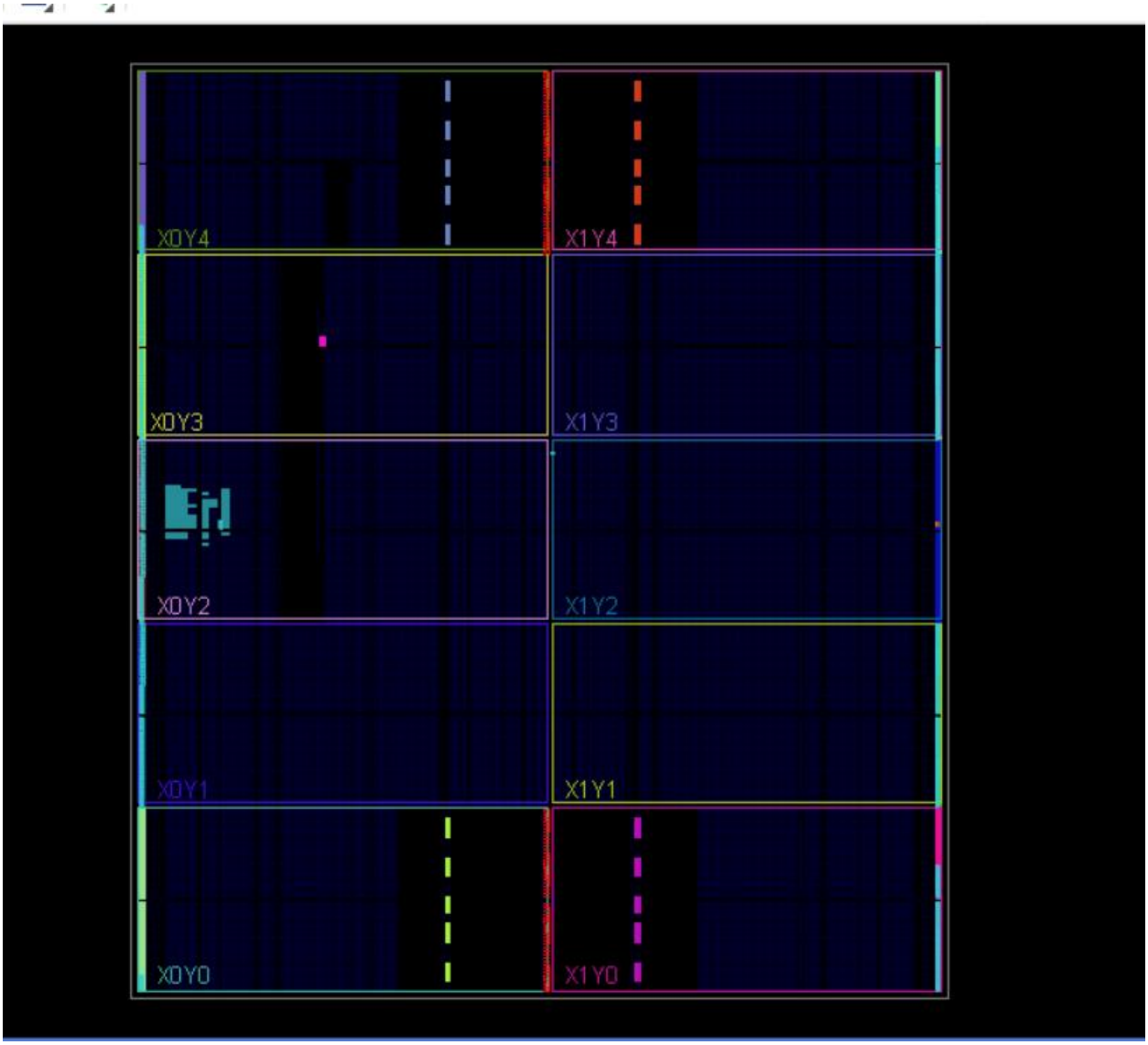
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	4.379 ns	Worst Hold Slack (WHS):	0.339 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:	105	Total Number of Endpoints:	105	Total Number of Endpoints:	144

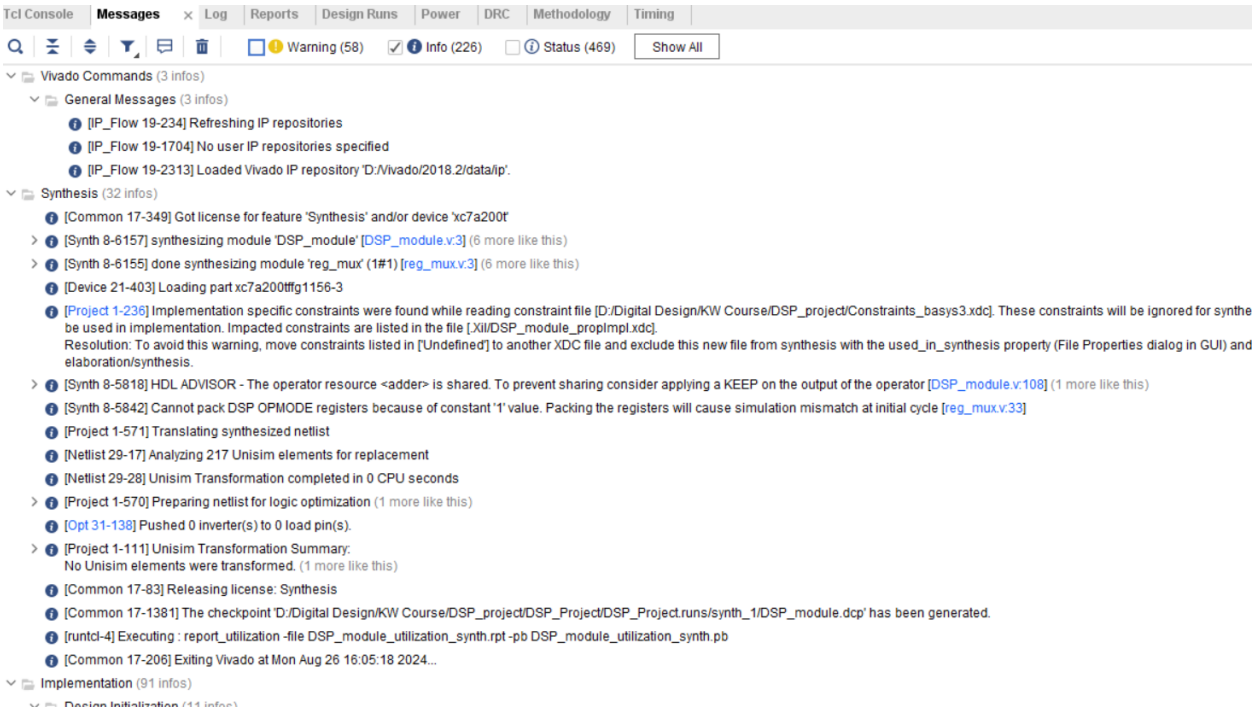
All user specified timing constraints are met.

Implementation:

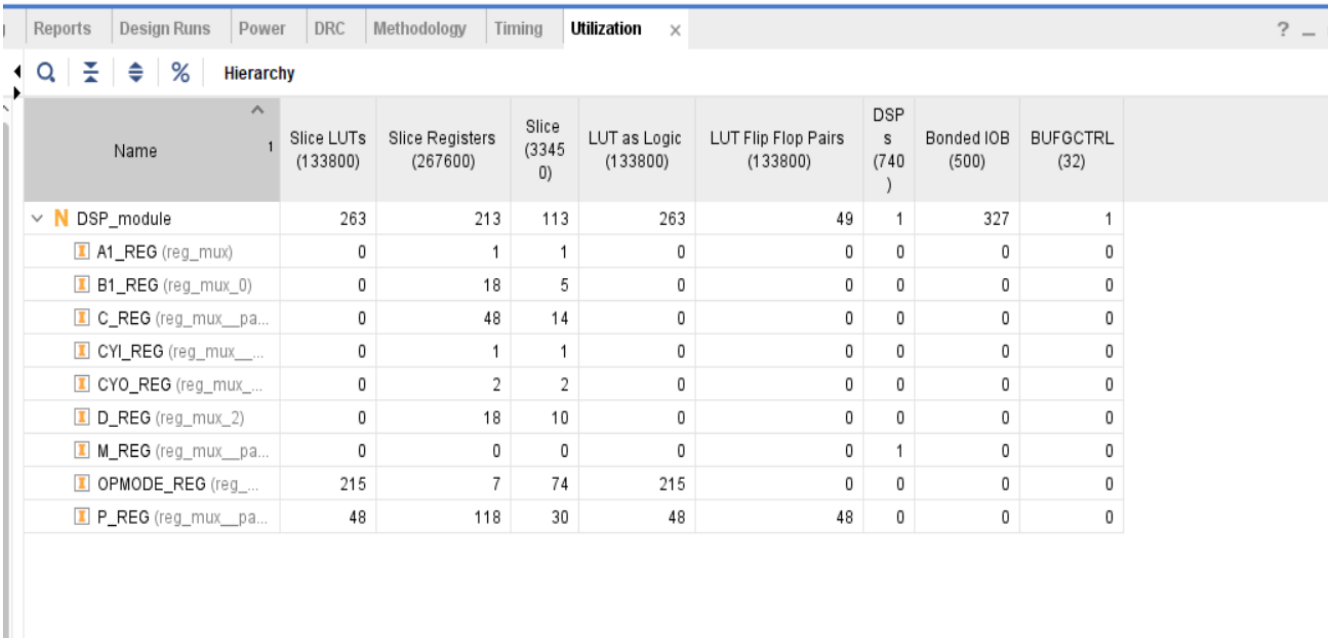
Device:



Message Tap:

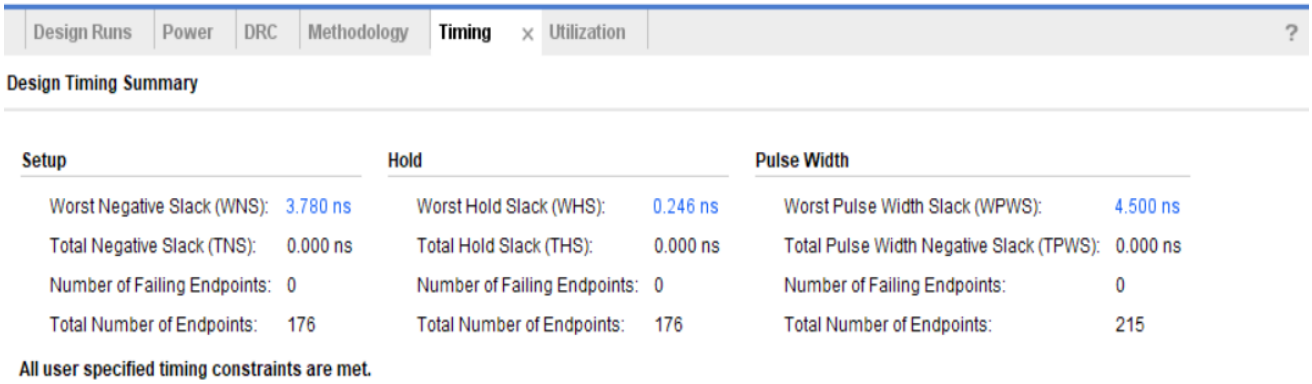


Utilization report:



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)
DSP_module	263	213	113	263	49	1	327	1
A1_REG (reg_mux)	0	1	1	0	0	0	0	0
B1_REG (reg_mux_0)	0	18	5	0	0	0	0	0
C_REG (reg_mux__pa...	0	48	14	0	0	0	0	0
CYI_REG (reg_mux__...	0	1	1	0	0	0	0	0
CYO_REG (reg_mux_...	0	2	2	0	0	0	0	0
D_REG (reg_mux_2)	0	18	10	0	0	0	0	0
M_REG (reg_mux__pa...	0	0	0	0	0	1	0	0
OPMODE_REG (reg_...	215	7	74	215	0	0	0	0
P_REG (reg_mux__pa...	48	118	30	48	48	0	0	0

Timing report:



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 -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_ports CLK]
```