Project1: Spartan6 - DSP48A1

Name: Mina Adel

RTL Code

Register multiplexer block

```
module reg_mul(clk,rst,ce,in,out);
    parameter REG=1;//select
    parameter RSTtype="SYNC";//rst is syncronus
    parameter width=18;
    input clk,rst,ce;
    input [width-1:0] in;
    output reg [width-1:0] out;
    generate
        if(REG==1)begin
            if (RSTtype == "SYNC") begin
                always @(posedge clk) begin
                    if(rst)
                        out<=0;
                    else if(ce)begin
                        out<=in;
                    end
                end
            end
            else if (RSTtype == "ASYNC") begin
                always @(posedge clk or posedge rst) begin
                    if(rst)
                        out<=0;
                    else if(ce)begin
                        out<=in;
                    end
                end
            end
        end
        else begin
            always @(*) begin
                out=in;
            end
        end
    endgenerate
endmodule
```

DSP

```
module DSP(a,b,c,d,cin,opmode,Pcin,Bcin,clk,CEa,CEb,CEc,CEcin,CEd,CEm,

CEopmode,CEp,rsta,rstb,rstc,rstcin,rstd,rstM,rstopmode,rstP,M,P,cout,coutf,bcout,
Pcout);

    parameter A0REG=0 , A1REG=1 ,B0REG=0 ,B1REG=1;
    parameter CREG=1, DREG=1, MREG=1, PREG=1, CARRYINREG=1, CARRYOUTREG=1,
OPREG=1;
    parameter CARRYINSEL="OPMODE5";
    parameter B_INPUT="DIRECT";
    parameter RSTTYPE ="SYNC";
    parameter w1=1, w8=8, w18=18, w36=36, w48=48;

    input [w18-1:0] a, b,d;
    input [w1-1:0] cin;
    input [w48-1:0] c;
    input [w8-1:0] opmode;
    input [47:0] Pcin;
    input [17:0] Bcin;
    input clk,CEa,CEb,CEc,CEcin,CEd,CEm,CEopmode,CEp;
    input rsta,rstb,rstc,rstcin,rstd,rstM,rstopmode,rstP;

    output [35:0] M;
    output [47:0] P;
    output cout,coutf;
    output [17:0] bcout;
    output [47:0] Pcout;

    wire [w18-1:0] d_reg;
    wire [w18-1:0] b0_reg;
    wire [w18-1:0] bb;
    wire [w18-1:0] a0_reg;
    wire [w48-1:0] c_reg;
    wire [w8-1:0] opmode_reg;
    wire [w18-1:0] k0;
    wire [w18-1:0] add1_out;
    wire [w18-1:0] k0_reg;
    wire [w18-1:0] a1_reg;
    wire [w36-1:0] m;
    wire [w36-1:0] m_reg;
    wire k1;
```

```
    wire Cin;
    wire [47:0] conc;
    wire [47:0] x_out,z_out;
    wire [47:0] add2_out;
    wire cout0;

    assign bb=(B_INPUT=="DIRECT")?b:(B_INPUT=="CASCADE")?Bcin:18'b0;

    reg_mul #(.REG(DREG),.RSTtype(RSTTYPE),.width(w18))
D_REG(clk,rstd,CEd,d,d_reg);
    reg_mul #(.REG(B0REG),.RSTtype(RSTTYPE),.width(w18))
B0_REG(clk,rstb,CEb,bb,b0_reg);
    reg_mul #(.REG(A0REG),.RSTtype(RSTTYPE),.width(w18))
A0_REG(clk,rsta,CEa,a,a0_reg);
    reg_mul #(.REG(CREG),.RSTtype(RSTTYPE),.width(w48))
C_REG(clk,rstc,CEc,c,c_reg);
    reg_mul #(.REG(OPREG),.RSTtype(RSTTYPE),.width(w8))
opmode_REG(clk,rstopmode,CEopmode,opmode,opmode_reg);

    assign add1_out=(opmode_reg[6]==0)?d_reg+b0_reg:d_reg-b0_reg;
    assign k0=(opmode_reg[4]==0)?b0_reg:add1_out;

    reg_mul #(.REG(B1REG),.RSTtype(RSTTYPE),.width(w18))
B1_REG(clk,rstb,CEb,k0,k0_reg);
    reg_mul #(.REG(A1REG),.RSTtype(RSTTYPE),.width(w18))
A1_REG(clk,rsta,CEa,a0_reg,a1_reg);

    assign bcout=k0_reg;
    assign m=k0_reg*a1_reg;

    reg_mul #(.REG(MREG),.RSTtype(RSTTYPE),.width(w36))
M_REG(clk,rstM,CEm,m,m_reg);

    assign M=m_reg;
    assign k1=(CARRYINSEL=="OPMODE5")?opmode_reg[5]:cin;

    reg_mul #(.REG(CARRYINREG),.RSTtype(RSTTYPE),.width(w1))
CYI(clk,rstcin,CEcin,k1,Cin);

    assign conc={d[11:0],a1_reg[17:0],k0_reg[17:0]};
```

```
    assign x_out=(opmode_reg[1:0]==2'b00)?48'b0:(opmode_reg[1:0]==2'b01)?{{(w48 -
w36){1'b0}},m_reg}:
    (opmode_reg[1:0]==2'b10)?Pcout:conc;

    assign
z_out=(opmode_reg[3:2]==2'b00)?48'b0:(opmode_reg[3:2]==2'b01)?Pcin:(opmode_reg[3:
2]==2'b10)?Pcout:c_reg;
    assign {cout0,add2_out}=(opmode_reg[7]==0)?z_out+x_out+Cin : z_out-
(x_out+Cin);

    reg_mul #(.REG(CARRYOUTREG),.RSTtype(RSTTYPE),.width(w1))
CYO(clk,rstcin,CEcin,cout0,cout);

    assign coutf=cout;

    reg_mul #(.REG(PREG),.RSTtype(RSTTYPE),.width(w48))
P_REG(clk,rstP,CEp,add2_out,P);

    assign Pcout = P;


endmodule
```

DSP Testbench

```
module DSP_tb();

    parameter A0REG_tb=0 , A1REG_tb=1 ,B0REG_tb=0 ,B1REG_tb=1;
    parameter CREG_tb=1, DREG_tb=1, MREG_tb=1, PREG_tb=1, CARRYINREG_tb=1,
CARRYOUTREG_tb=1, OPREG_tb=1;
    parameter CARRYINSEL_tb="OPMODE5";
    parameter B_INPUT_tb="DIRECT";
    parameter RSTTYPE_tb ="SYNC";
    parameter w1_tb=1, w8_tb=8, w18_tb=18, w36_tb=36, w48_tb=48;

    reg [w18_tb-1:0] a_tb, b_tb,d_tb;
    reg [w1_tb-1:0] cin_tb;
    reg [w48_tb-1:0] c_tb;
    reg [w8_tb-1:0] opmode_tb;
    reg [47:0] Pcin_tb;
    reg [17:0] Bcin_tb;
    reg clk_tb,CEa_tb,CEb_tb,CEc_tb,CEcin_tb,CEd_tb,CEm_tb,CEopmode_tb,CEp_tb;
    reg rsta_tb,rstb_tb,rstc_tb,rstcin_tb,rstd_tb,rstM_tb,rstopmode_tb,rstP_tb;

    wire [35:0] M_dut;
    wire [47:0] P_dut;
    wire cout_dut,coutf_dut;
    wire [17:0] bcout_dut;
    wire [47:0] Pcout_dut;

    DSP #(.A0REG(A0REG_tb), .A1REG(A1REG_tb), .B0REG(B0REG_tb), .B1REG(B1REG_tb),
          .CREG(CREG_tb), .DREG(DREG_tb), .MREG(MREG_tb), .PREG(PREG_tb),
          .CARRYINREG(CARRYINREG_tb), .CARRYOUTREG(CARRYOUTREG_tb), .OPREG(OPREG_
tb),
          .CARRYINSEL(CARRYINSEL_tb), .B_INPUT(B_INPUT_tb), .RSTTYPE(RSTTYPE_tb),
          .w1(w1_tb), .w8(w8_tb), .w18(w18_tb), .w36(w36_tb), .w48(w48_tb))

        D1(a_tb, b_tb, c_tb, d_tb, cin_tb, opmode_tb, Pcin_tb, Bcin_tb, clk_tb,
           CEa_tb, CEb_tb, CEc_tb, CEcin_tb, CEd_tb, CEm_tb, CEopmode_tb, CEp_tb,
           rsta_tb, rstb_tb, rstc_tb, rstcin_tb, rstd_tb, rstM_tb, rstopmode_tb,
           rstP_tb, M_dut, P_dut, cout_dut, coutf_dut, bcout_dut, Pcout_dut);
```

```
    initial begin
        clk_tb=0;
        forever begin
            #1 clk_tb=~clk_tb;
        end
    end

    integer i;
    initial begin
        rsta_tb=1; rstb_tb=1; rstc_tb=1; rstcin_tb=1; rstd_tb=1; rstM_tb=1;
rstopmode_tb=1; rstP_tb=1;
        a_tb=$random; b_tb=$random; c_tb=$random; d_tb=$random;
        cin_tb=$random;
        opmode_tb=$random;
        Pcin_tb=$random;
        Bcin_tb=$random;
        CEa_tb=$random; CEb_tb=$random; CEc_tb=$random; CEcin_tb=$random;
CEd_tb=$random;
        CEm_tb=$random; CEopmode_tb=$random; CEp_tb=$random;

        @(negedge clk_tb);
        if (M_dut!=0 || P_dut!=0 || cout_dut!=0 || coutf_dut!=0 || bcout_dut!=0
|| Pcout_dut!=0 ) begin
            $display("error");
            $stop;
        end
        else begin
            $display("BCOUT=%h, M=%h, P=%h, PCOUT=%h, Carryout=%h,
Carryoutf=%h",bcout_dut, M_dut, P_dut, Pcout_dut, cout_dut, coutf_dut);
            $display("reset function is good");
        end

        rsta_tb=0; rstb_tb=0; rstc_tb=0; rstcin_tb=0; rstd_tb=0; rstM_tb=0;
rstopmode_tb=0; rstP_tb=0;
        CEa_tb=1; CEb_tb=1; CEc_tb=1; CEcin_tb=1; CEd_tb=1; CEm_tb=1;
CEopmode_tb=1; CEp_tb=1;

        opmode_tb= 8'b11011101;
        a_tb = 20; b_tb = 10; c_tb= 350; d_tb = 25;
        cin_tb=$random; Pcin_tb=$random; Bcin_tb=$random;

        repeat(4) @(negedge clk_tb);
```

```
        if (bcout_dut !='hf ||  M_dut != 'h12c || P_dut !='h32 || Pcout_dut !=
'h32 || cout_dut != 0 || coutf_dut!= 0) begin
            $display("error");
            $stop;
        end
        else begin
            $display("BCOUT=%h, M=%h, P=%h, PCOUT=%h, Carryout=%h,
Carryoutf=%h",bcout_dut,M_dut, P_dut, Pcout_dut, cout_dut, coutf_dut);
            $display("no errors in path 1");
        end

        opmode_tb= 8'b00010000;
        a_tb = 20; b_tb = 10; c_tb= 350; d_tb = 25;
        cin_tb=$random; Pcin_tb=$random; Bcin_tb=$random;
        repeat(3) @(negedge clk_tb);
        if (bcout_dut !='h23 ||  M_dut != 'h2bc || P_dut !='h0 || Pcout_dut !=
'h0 || cout_dut != 0 || coutf_dut!= 0) begin
            $display("error");
            $stop;
        end
        else begin
            $display("BCOUT=%h, M=%h, P=%h, PCOUT=%h, Carryout=%h,
Carryoutf=%h",bcout_dut, M_dut, P_dut, Pcout_dut, cout_dut, coutf_dut);
            $display("no errors in path 2");
        end

        opmode_tb= 8'b00001010;
        a_tb = 20; b_tb = 10; c_tb= 350; d_tb = 25;
        cin_tb=$random; Pcin_tb=$random; Bcin_tb=$random;
        repeat(3) @(negedge clk_tb);
        if (bcout_dut !='ha ||  M_dut != 'hc8 || P_dut !='h0 || Pcout_dut != 'h0
|| cout_dut != 0 || coutf_dut!= 0) begin
            $display("error");
            $stop;
        end
        else begin
            $display("BCOUT=%h, M=%h, P=%h, PCOUT=%h, Carryout=%h,
Carryoutf=%h",bcout_dut, M_dut, P_dut, Pcout_dut, cout_dut, coutf_dut);
            $display("no errors in path 3");
        end

        opmode_tb= 8'b10100111;
        a_tb = 5; b_tb = 6; c_tb= 350; d_tb = 25; Pcin_tb=3000;
```

```
        cin_tb=$random; Bcin_tb=$random;
        repeat(3) @(negedge clk_tb);
        if (bcout_dut !='h6 ||  M_dut != 'h1e || P_dut !='hfe6fffec0bb1 ||
Pcout_dut !='hfe6fffec0bb1 || cout_dut != 1 || coutf_dut!= 1) begin
            $display("BCOUT=%h, M=%h, P=%h, PCOUT=%h, Carryout=%h,
Carryoutf=%h",bcout_dut, M_dut, P_dut, Pcout_dut, cout_dut, coutf_dut);
            $display("error");
            $stop;
        end
        else begin
            $display("BCOUT=%h, M=%h, P=%h, PCOUT=%h, Carryout=%h,
Carryoutf=%h",bcout_dut, M_dut, P_dut, Pcout_dut, cout_dut, coutf_dut);
            $display("no errors in path 4");
        end

        $display("all is good isa");
        $stop;

    end

endmodule
```

Do file

(automating simulation)

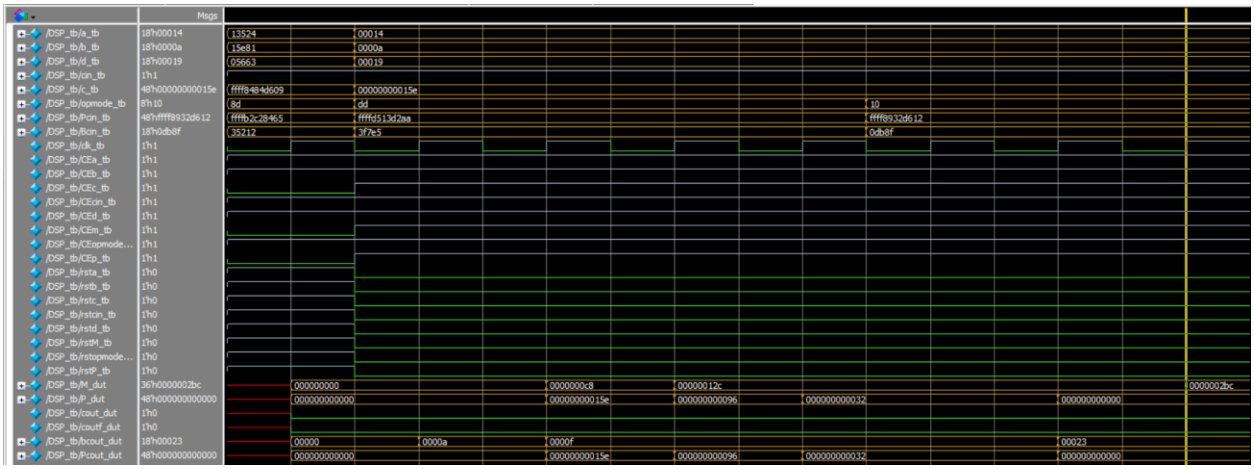
```
vlib work
vlog reg_mul_block.v DSP.v DSP_tb.v
 vsim -voptargs=+acc work.DSP_tb
add wave *
run –all
 #quit -sim
```

QuestaSim Snippets

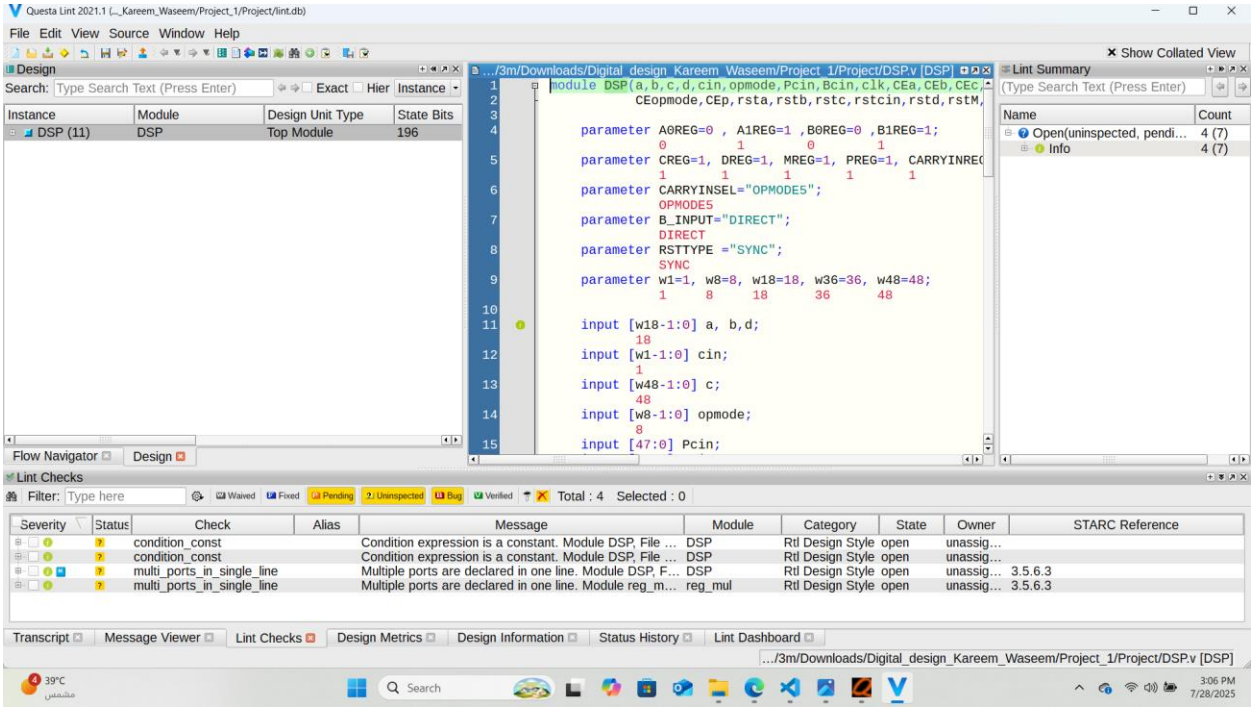
Transcript

```
# BCOUT=00000, M=000000000, P=000000000000, PCOUT=000000000000, Carryout=0, Carryoutf=0
# reset function is good
# BCOUT=0000f, M=00000012c, P=000000000032, PCOUT=000000000032, Carryout=0, Carryoutf=0
# no errors in path 1
# BCOUT=00023, M=0000002bc, P=000000000000, PCOUT=000000000000, Carryout=0, Carryoutf=0
# no errors in path 2
# BCOUT=0000a, M=0000000c8, P=000000000000, PCOUT=000000000000, Carryout=0, Carryoutf=0
# no errors in path 3
# BCOUT=00006, M=00000001e, P=fe6fffec0bb1, PCOUT=fe6fffec0bb1, Carryout=1, Carryoutf=1
# no errors in path 4
# all is good isa
```

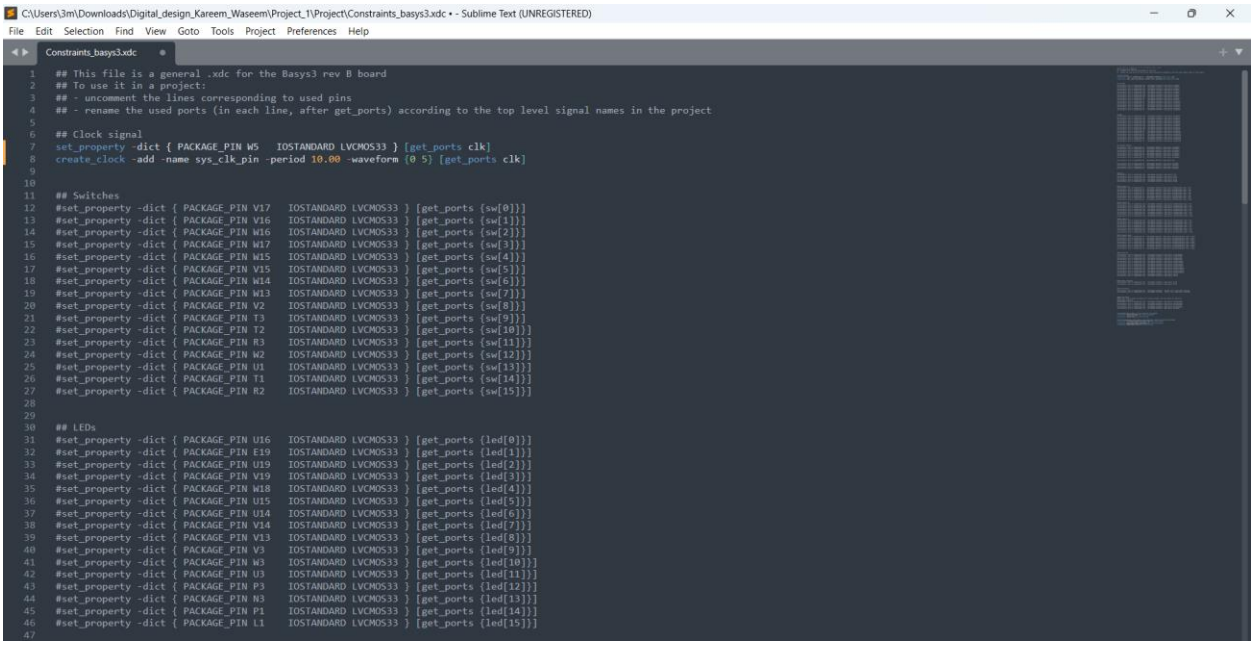
Waveform



Linting


Questa Lint 2021.1 (..._Kareem_Waseem/Project_1/Project/lint.db)
File Edit View Source Window Help
Show Collated View
Design
Search: Type Search Text (Press Enter) Exact Hier Instance
Instance Module Design Unit Type State Bits
DSP (11) DSP Top Module 196
Flow Navigator Design
.../3m/Downloads/Digital_design_Kareem_Waseem/Project_1/Project/DSP.v [DSP]
module DSP(a,b,c,d,cin,opmode,Pcin,Bcin,clk,CEa,CEb,CEc,
CEopmode,CEp,rsta,rstb,rstc,rstcin,rstd,rstM,
parameter A0REG=0 , A1REG=1 ,B0REG=0 ,B1REG=1;
parameter CREG=1, DREG=1, MREG=1, PREG=1, CARRYINREG
parameter CARRYINSEL="OPMODE5";
parameter B_INPUT="DIRECT";
parameter RSTTYPE ="SYNC";
parameter w1=1, w8=8, w18=18, w36=36, w48=48;
input [w18-1:0] a, b,d;
input [w1-1:0] cin;
input [w48-1:0] c;
input [w8-1:0] opmode;
input [47:0] Pcin;
Lint Summary
(Type Search Text (Press Enter)
Name Count
Open(uninspected, pendi... 4 (7)
Info 4 (7)
Lint Checks
Filter: Type here Waived Fixed Pending Uninspected Bug Verified Total : 4 Selected : 0
Severity Status Check Alias Message Module Category State Owner STARC Reference
condition_const Condition expression is a constant. Module DSP, File ... DSP Rtl Design Style open unassig...
condition_const Condition expression is a constant. Module DSP, File ... DSP Rtl Design Style open unassig...
multi_ports_in_single_line Multiple ports are declared in one line. Module DSP, F... DSP Rtl Design Style open unassig... 3.5.6.3
multi_ports_in_single_line Multiple ports are declared in one line. Module reg_m... reg_mul Rtl Design Style open unassig... 3.5.6.3
Transcript Message Viewer Lint Checks Design Metrics Design Information Status History Lint Dashboard
.../3m/Downloads/Digital_design_Kareem_Waseem/Project_1/Project/DSP.v [DSP]
39°C
Search
3:06 PM
7/28/2025

Constraint File

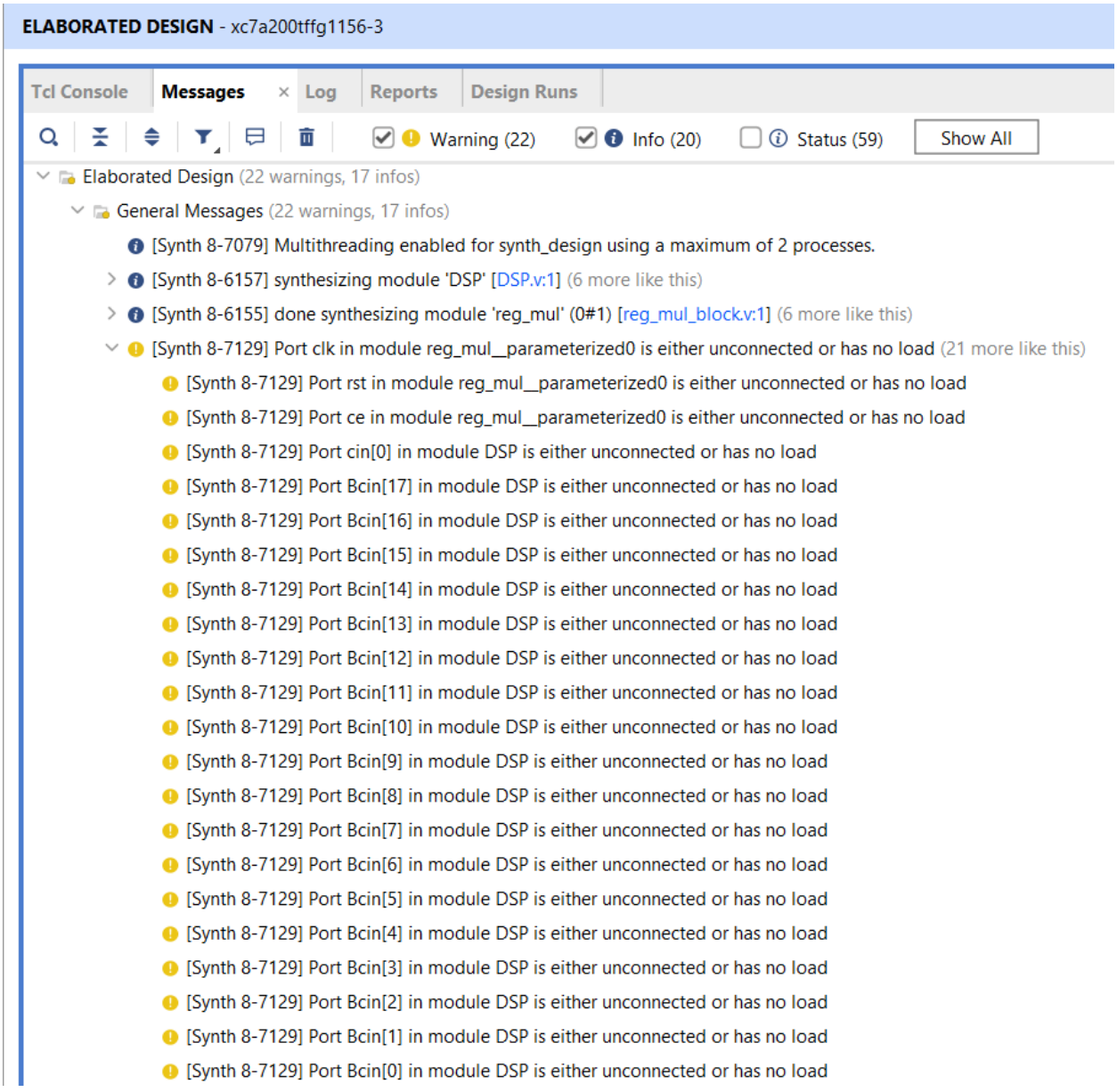


Vivado

Elaboration

Messages

to check that there is no errors or critical



The 22 errors because of the selector parameters when we select an input port another input port will be unused so there will be a warning for it

Analyzing warnings

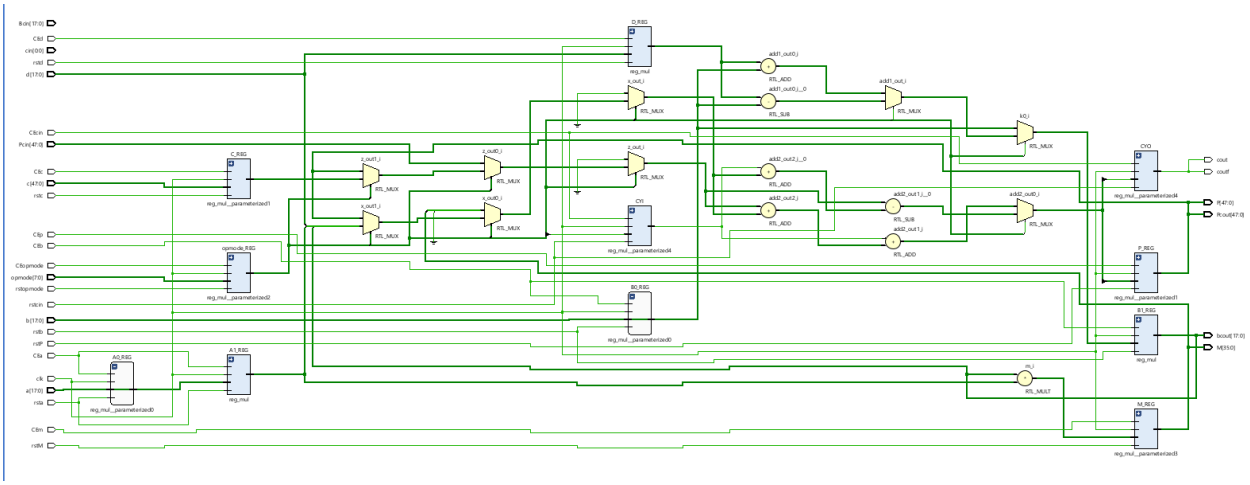
Some warnings is reg_mul.v file as when REG ==0 so clk and rst and ce will not be used
and others in DSP.v file as when carryinsel == opmode5 so cin will not be used and when B_INPUT == DIRECT so b_cin will not be used

```
// Dummy usage to avoid unconnected port warnings
// generate
//     if (CARRYINSEL == "OPMODE5") begin
//         (* DONT_TOUCH = "true" *) wire cin_unused = |cin;
//     end

//     if (B_INPUT != "CASCADE") begin
//         (* DONT_TOUCH = "true" *) wire bcin_unused = |Bcin;
//     end
// endgenerate
```

```
else begin
    always @(*) begin
        out=in;
    end
    // Dummy usage to avoid unconnected port warnings

    // (* DONT_TOUCH = "true" *) reg clk_unused, rst_unused, ce_unused;
    // always @(posedge clk) begin
    //     clk_unused <= clk_unused;
    //     rst_unused <= rst;
    //     ce_unused<=ce;
    //  end
end
```

these is a solutions but it will increase the hardware so it doesn’t make sense to avoide some not important warnings increase the hardware

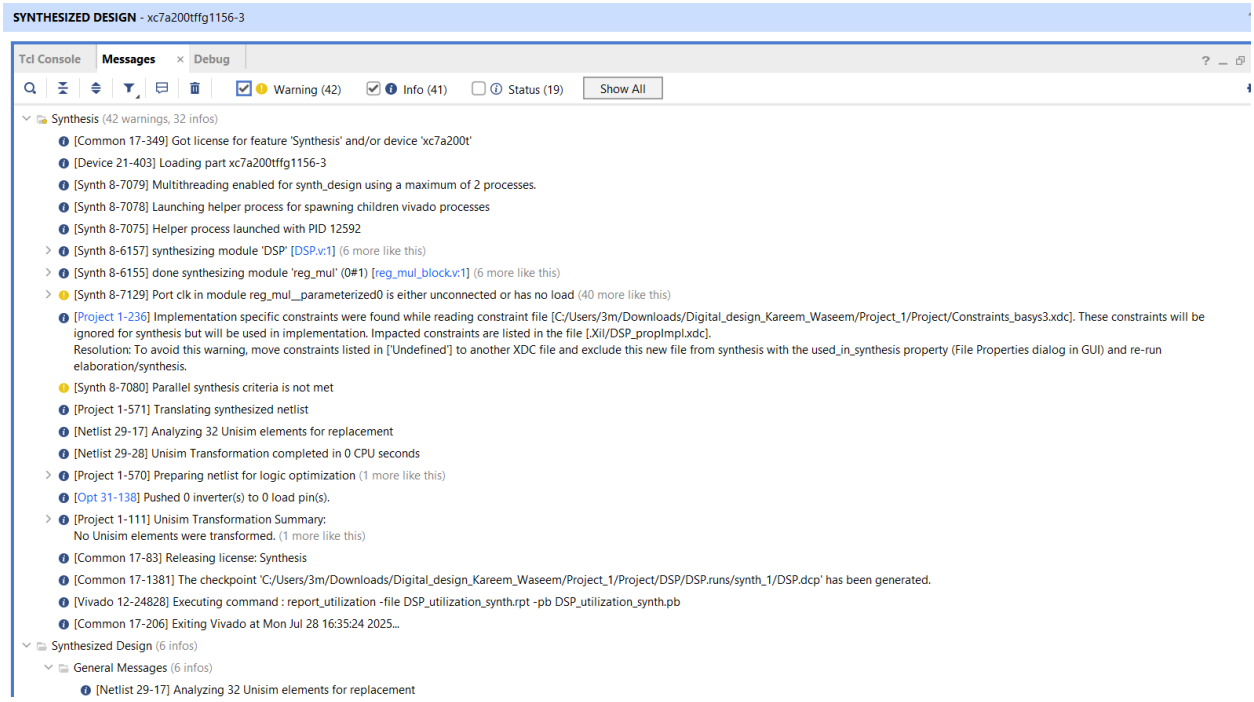
Schematic


B:on[17:0]
C:bl
cin[0:0]
rstd
d[17:0]
C:lc:in
P:on[47:0]
C:lc
c[47:0]
rstc
C:lp
C:lb
C:lp:pmode
opmode[7:0]
rstopmode
rstc:in
b[17:0]
rstb
rstP
C:la
clk
a[17:0]
rsta
C:bn
rstM
C_REG
reg_mul__parameterized1
opmode_REG
reg_mul__parameterized2
A0_REG
reg_mul__parameterized0
A1_REG
reg_mul
D_REG
reg_mul
B0_REG
reg_mul__parameterized0
CY1
reg_mul__parameterized4
CY0
reg_mul__parameterized4
P_REG
reg_mul__parameterized1
B1_REG
reg_mul
M_REG
reg_mul__parameterized3
RTL_MUX
RTL_ADD
RTL_SUB
RTL_MULT
cout
cout1
P[47:0]
P_out[47:0]
bcout[17:0]
M[35:0]

Synthesis

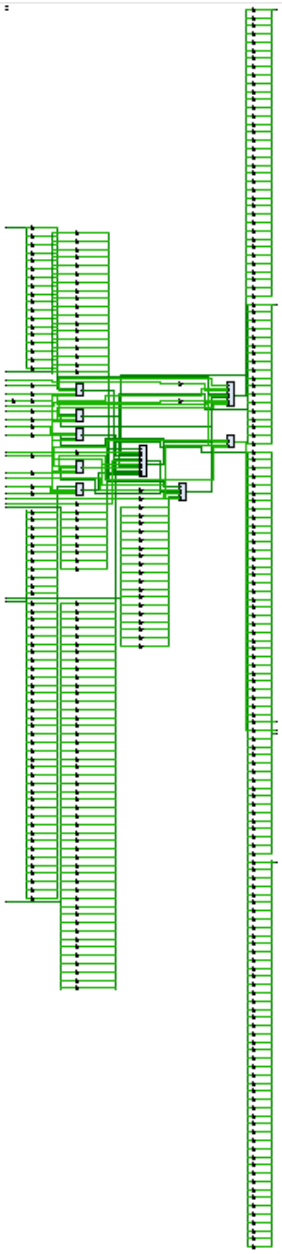
Messages

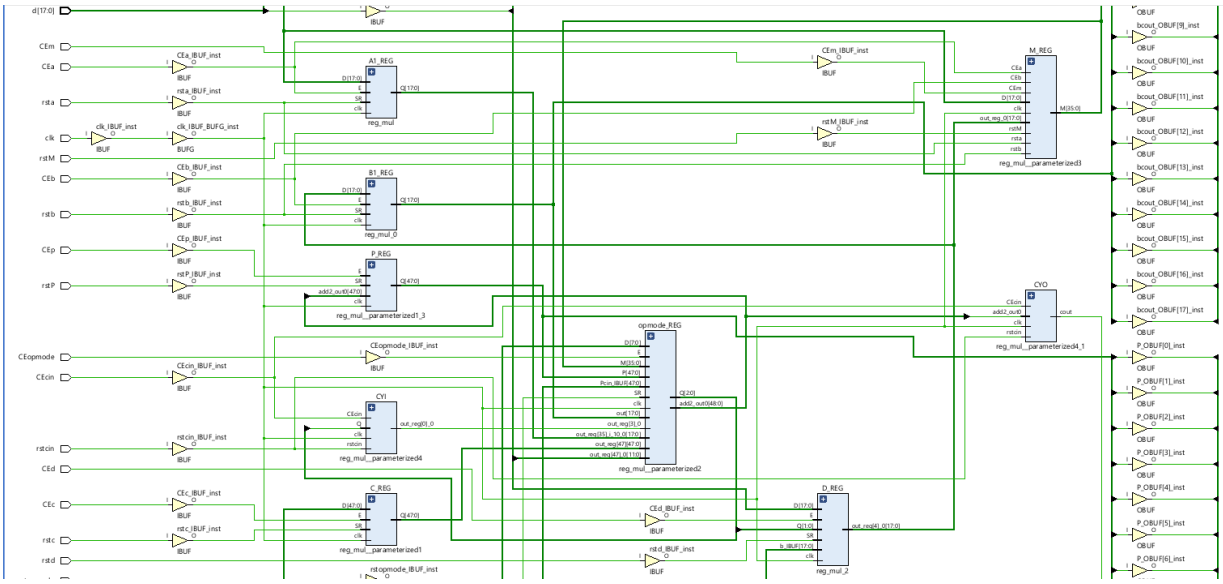
to check that there is no errors or critical



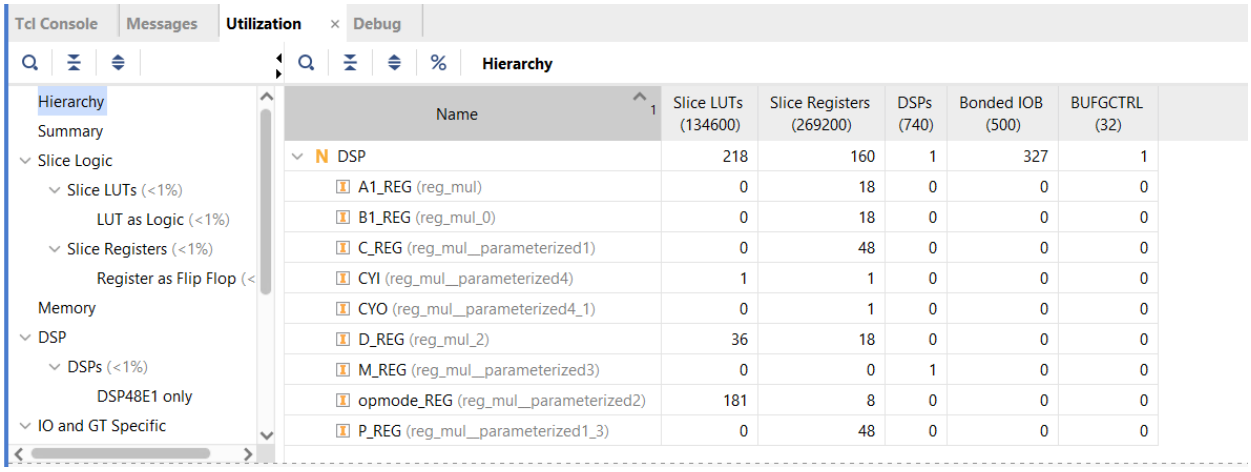
Warning becaue the same thing about unconnected ports in some events so we can ignore them

Schematic snippets

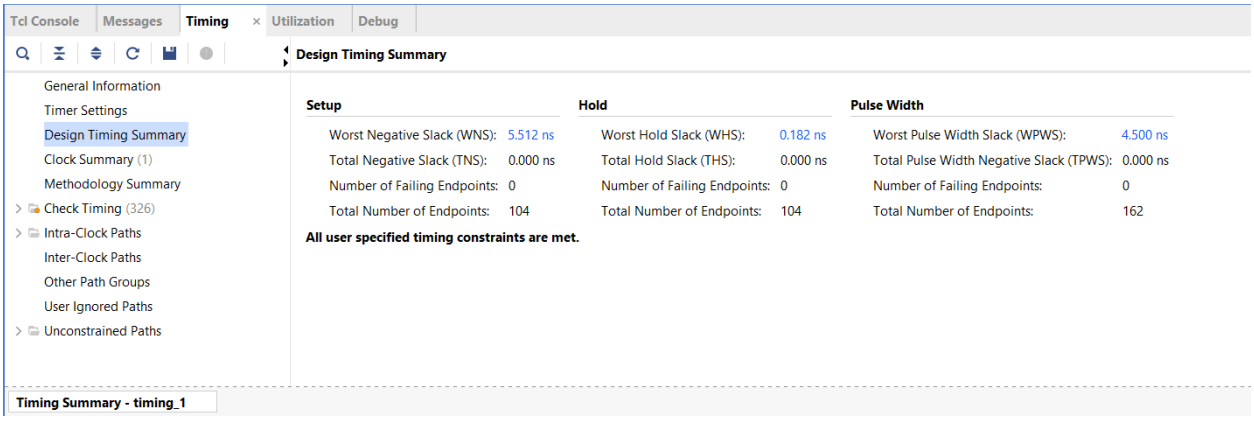



d[17:0]
CEm
CEa
rsta
clk
rstM
CEb
rstb
CEp
rstP
CEopmode
CEcin
rstcin
CEd
CEc
rstc
rstd
CEa_IBUF_inst
rsta_IBUF_inst
clk_IBUF_inst
clk_IBUF_BUFG_inst
CEb_IBUF_inst
rstb_IBUF_inst
CEp_IBUF_inst
rstP_IBUF_inst
CEopmode_IBUF_inst
CEcin_IBUF_inst
rstcin_IBUF_inst
CEc_IBUF_inst
rstc_IBUF_inst
rstopmode_IBUF_inst
CEm_IBUF_inst
rstM_IBUF_inst
CEd_IBUF_inst
rstd_IBUF_inst
IBUF
BUFG
A1_REG
reg_mul
B1_REG
reg_mul_0
P_REG
reg_mul__parameterized1_3
C_REG
reg_mul__parameterized1
CY1
reg_mul__parameterized4
opmode_REG
reg_mul__parameterized2
M_REG
reg_mul__parameterized3
D_REG
reg_mul_2
CYO
reg_mul__parameterized4_1
OBUF
bcout_OBUF[9]_inst
bcout_OBUF[10]_inst
bcout_OBUF[11]_inst
bcout_OBUF[12]_inst
bcout_OBUF[13]_inst
bcout_OBUF[14]_inst
bcout_OBUF[15]_inst
bcout_OBUF[16]_inst
bcout_OBUF[17]_inst
P_OBUF[0]_inst
P_OBUF[1]_inst
P_OBUF[2]_inst
P_OBUF[3]_inst
P_OBUF[4]_inst
P_OBUF[5]_inst
P_OBUF[6]_inst

Utilization report



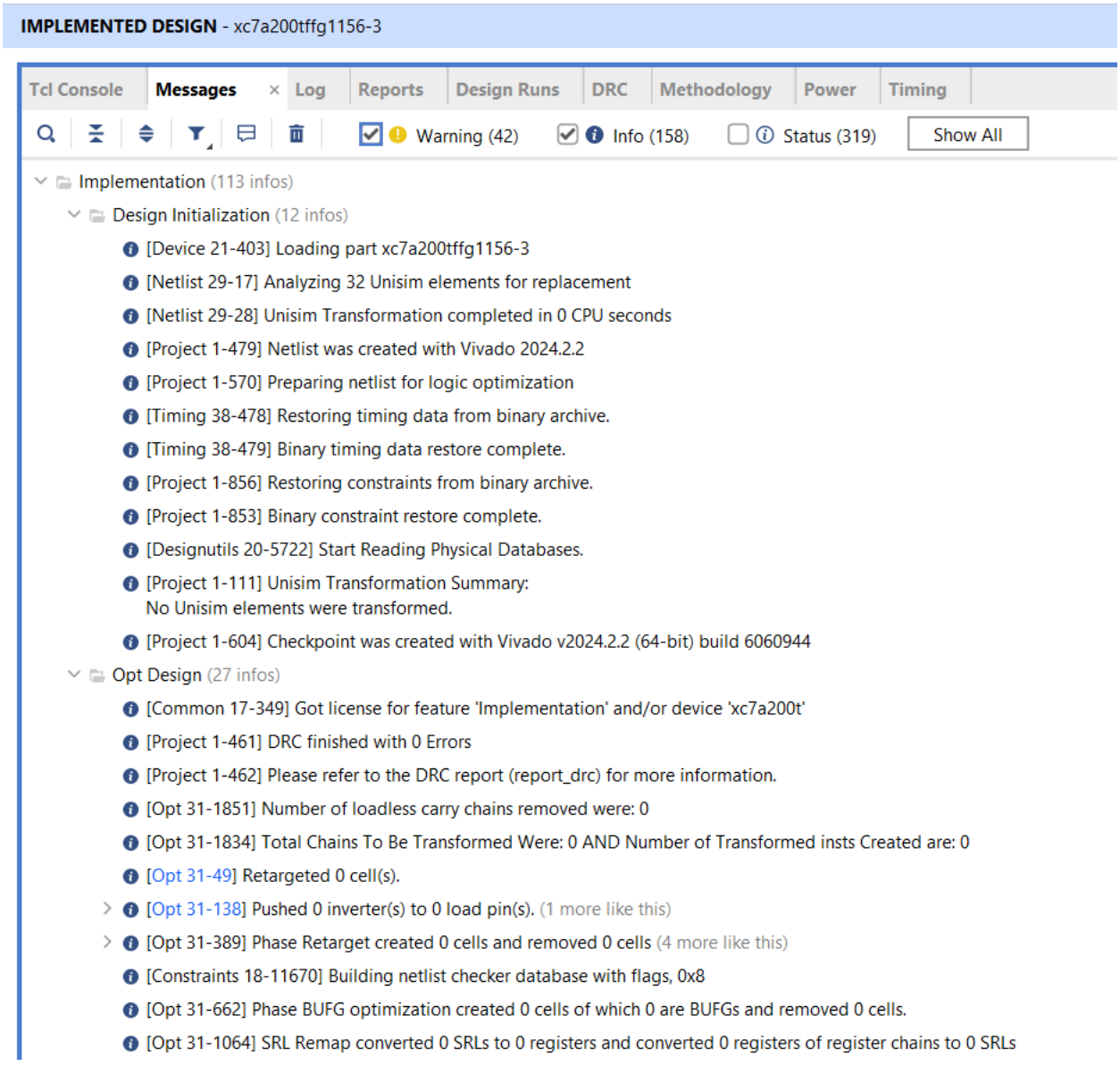
timing report



Implementation

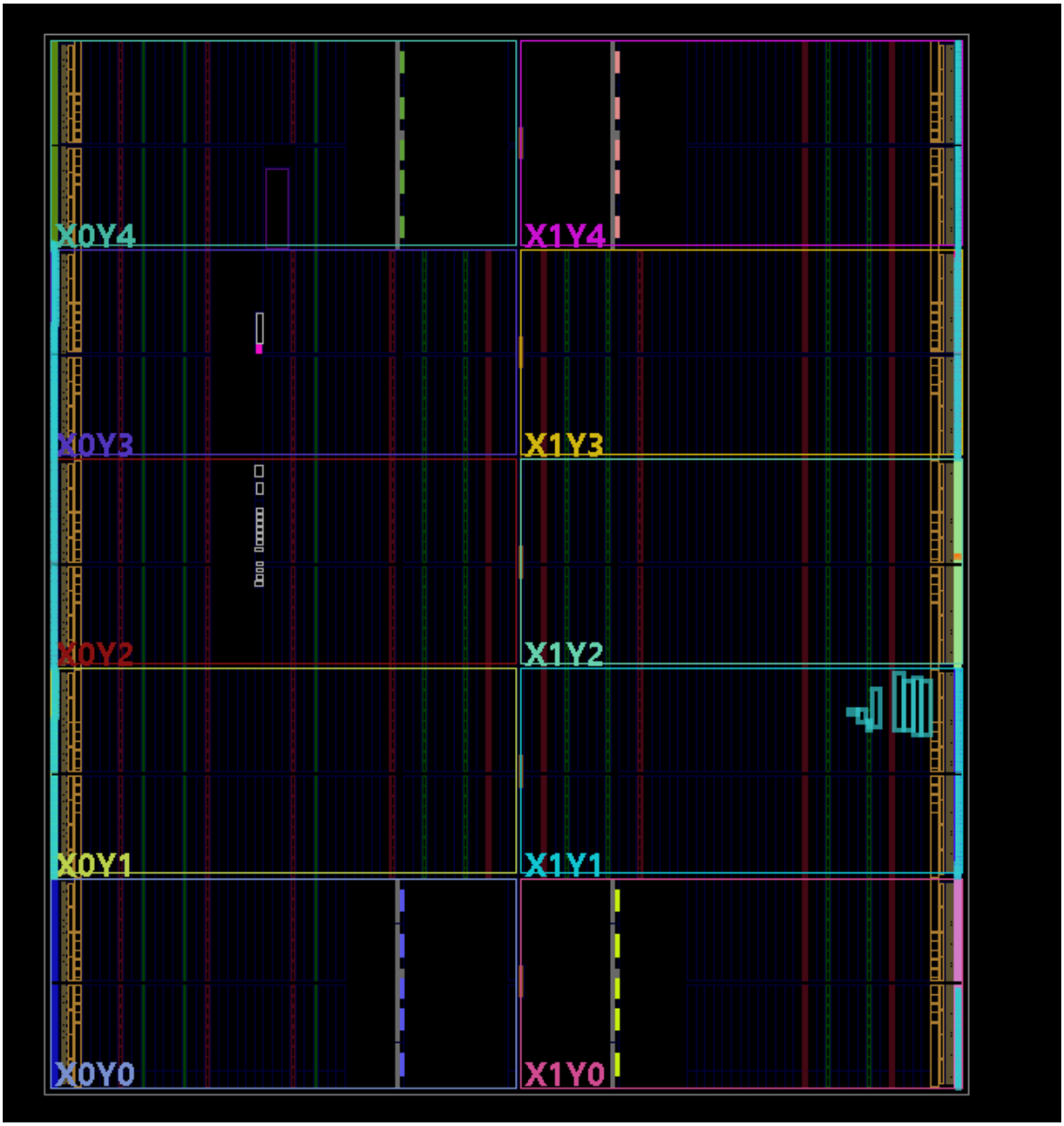
Messages

to check that there is no errors or critical

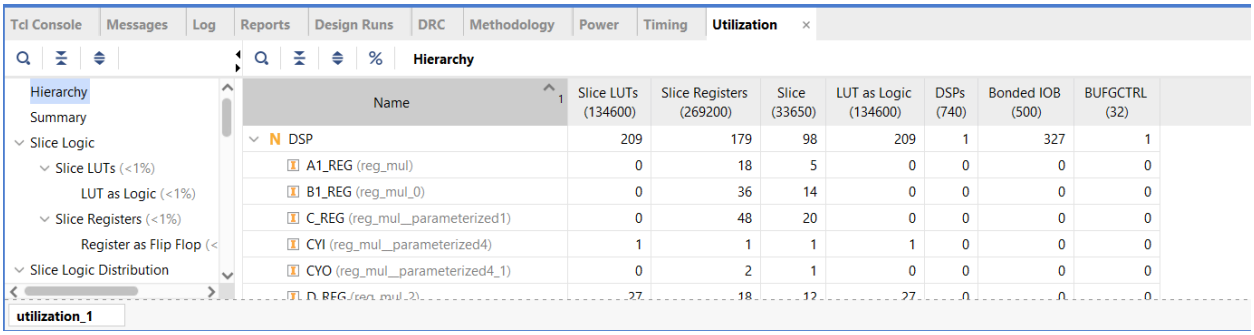


no warnings no errors

Device



Utilization report



timing report