Spartan6 - DSP48A1

Supervisor: Eng. Kareem Wassem
Created by: Ahmed Amr Ali

Contents

Spartan-6 FPGA: DSP48A1 Slice Overview 3

- Description 3
- Key Features 3
- Applications 3
- Configuration and Usage 4

RTL Code: 5

- The Pipeline_mux: 5
- The DSP: 6

The TestBench: 9

DSP.do File: 13

The waveform: 13

Constraint file: 14

Elaboration: 14

- Messages 14
- Schematic 15

Synthesis: 16

- Schematic 16
- Messages 16
- Timing report 16
- Utilization report 16

Implementation: 17

- Device 17
- Messages 17
- Timing report 17
- Utilization report 17

Spartan-6 FPGA: DSP48A1 Slice Overview

- ## Description

  The Spartan-6 FPGA series by Xilinx is designed for a wide range of applications that require low power, cost-effectiveness, and high performance. The DSP48A1 slice is an integral part of the Spartan-6 FPGA family, specifically optimized for digital signal processing (DSP) tasks. This slice is a flexible and powerful arithmetic unit that can perform various operations like addition, subtraction, multiplication, and more complex functions, making it a cornerstone for implementing DSP algorithms efficiently.

- ## Key Features

  - **16x16-bit Multiplier:** Provides high-performance multiplication capability, essential for DSP algorithms.
  - **48-bit Accumulator:** Allows for large dynamic range in accumulation operations, critical for applications like filtering and FFTs.
  - **Integrated Logic Functions:** Supports various logical operations and shift functions, increasing the slice's versatility.
  - **Pipelining and Bypass Options:** Enables high-speed data processing with options for pipelining or bypassing stages, depending on the application requirements.
  - **Low Power Consumption:** Optimized for power efficiency, making it suitable for battery-operated and low-power designs.
  - **Configurable Input and Output Registers:** Flexible configuration options to support different operational modes and data paths.

- ## Applications

  - **Digital Signal Processing (DSP):** Ideal for implementing high-performance DSP algorithms such as FIR filters, FFTs, and convolutions.
  - **Wireless Communication:** Used in modulation, demodulation, and other signal processing tasks in wireless communication systems.
  - **Audio and Video Processing:** Facilitates real-time audio and video processing, including encoding, decoding, and filtering.
  - **Image Processing:** Supports complex image processing operations like edge detection, filtering, and transformation.
  - **Medical Electronics:** Utilized in medical imaging devices and diagnostic tools for real-time data processing.
  - **Automotive Systems:** Plays a key role in advanced driver-assistance systems (ADAS) and other in-vehicle signal processing tasks.

- ## Configuration and Usage

  The DSP48A1 slice can be configured in multiple modes to cater to various application needs. It can be used as a standalone arithmetic unit or as part of a larger, more complex DSP engine within the FPGA. Configuration involves setting up the slice through the Xilinx design tools (e.g., Vivado or ISE) to define the specific operations, pipelining, and data paths. The flexibility in configuration allows for tailored performance optimizations, enabling developers to balance power, speed, and resource utilization based on the application's requirements.

RTL Code:

- The Pipeline_mux:

```
module pipeline_mux (in, sel, CE, clk, rst, out);
//Parameters
parameter WIDTH = 18;       //The width of the inputs and outputs
parameter RSTTYPE = "SYNC"; //SYNC or ASYNC for the resets

//Input ports
input sel;              //Selection of the mux
input [WIDTH-1 : 0] in; //The input
input CE;               //Enable comtrol signal for reg
input clk, rst;         //Clock and reset

//Output ports
output [WIDTH-1 : 0] out;   //The output after MUX selection
reg [WIDTH-1 : 0] in_r;     //Register output

//Generate for the register if the reset is sync or async
generate
    //For sync resets
    if (RSTTYPE == "SYNC") begin
            always @(posedge clk) begin
                if (rst)
                    in_r <= 0;
                else if (CE)
                    in_r <= in;
            end
    end
    //For async resets
    else if (RSTTYPE == "ASYNC") begin
            always @(posedge clk or posedge rst) begin
                if (rst)
                    in_r <= 0;
                else if (CE)
                    in_r <= in;
            end
    end
endgenerate

//Choose between SQ output or comb
assign out = (sel)? in_r : in;
endmodule
```

- The DSP:

```
module DSP48A1 (A, B, BCIN, C, D, opmode,
CARRYIN, CEA, CEB, CEC, CECARRYIN, CED, CEM,
CEOPMODE, CEP, clk, RSTA, RSTB, RSTC, RSTCARRYIN,
RSTD, RSTM, RSTOPMODE, RSTP, M, P, CARRYOUT, CARRYOUTF,
BCOUT, PCIN, PCOUT);

//DSP parameters
parameter A0REG = 0;                //0 for SQ 1 for comb
parameter A1REG = 1;                //0 for SQ 1 for comb
parameter B0REG = 0;                //0 for SQ 1 for comb
parameter B1REG = 1;                //0 for SQ 1 for comb
parameter CREG = 1;                 //0 for SQ 1 for comb
parameter DREG = 1;                 //0 for SQ 1 for comb
parameter MREG = 1;                 //0 for SQ 1 for comb
parameter PREG  = 1;                //0 for SQ 1 for comb
parameter CARRYINREG = 1;           //0 for SQ 1 for comb
parameter CARRYOUTREG = 1;          //0 for SQ 1 for comb
parameter OPMODEREG = 1;            //0 for SQ 1 for comb
parameter CARRYINSEL  = "OPMODE5";  //CARRYIN or OPMODES if you want the carry to be
forced from opmode or not
parameter B_INPUT = "DIRECT";       //DIRECT or CASCADE if you want the B to be
cascaded from other DSP or an input
parameter RSTTYPE = "SYNC";         //SYNC or ASYNC if you want all the resets to be
sync or not

//input ports
input [17:0] A, B, D;
input [47:0] C;
input [17:0] BCIN;      //Cascaded input from other DSPs
input [47:0] PCIN;      //Cascaded input from other DSPs
input [7:0] opmode;     //Operation mode selection
input CARRYIN;
input CEA, CEB, CEC, CECARRYIN, CED, CEM, CEOPMODE, CEP;            //Control signals
input clk;                                                          //Clock signal
input RSTA, RSTB, RSTC, RSTCARRYIN, RSTD, RSTM, RSTOPMODE, RSTP;    //Reset signals

//output ports
output [47:0] P, PCOUT;
output [17:0] BCOUT;        //The output from B to cascade it to other DSPs
output [35:0] M;            //the multiplyer output to cascade it to other DSPs
```

```
output CARRYOUT, CARRYOUTF; //CARRYOUTF fromn the FPGA, CARRYOUT to other DSPs

wire [7:0] opmodeMUX;
pipeline_mux #(.WIDTH(8), .RSTTYPE(RSTTYPE)) OPMODEREG_pipeline (.in(opmode),
.sel(OPMODEREG), .CE(CEOPMODE), .clk(clk), .rst(RSTOPMODE), .out(opmodeMUX));

wire [17:0] DMUX;
pipeline_mux #(.WIDTH(18), .RSTTYPE(RSTTYPE)) DREG_pipeline (.in(D), .sel(DREG),
.CE(CED), .clk(clk), .rst(RSTD), .out(DMUX));

wire [17:0] B0IN;
assign B0IN = (B_INPUT == "DIRECT")? B : (B_INPUT == "CASCADE")? BCIN : 0;
wire [17:0] B0MUX;
pipeline_mux #(.WIDTH(18), .RSTTYPE(RSTTYPE)) B0REG_pipeline (.in(B0IN), .sel(B0REG),
.CE(CEB), .clk(clk), .rst(RSTB), .out(B0MUX));

//Pre - adder / subtracter
wire [17:0] adder1;
assign adder1 = (opmodeMUX[6])? DMUX - B0MUX : DMUX + B0MUX;

//MUX for choosing to skip pre - adder / subtracter or use it
wire [17:0] B1IN;
assign B1IN = (opmodeMUX[4])? adder1 : B0MUX;
wire [17:0] B1MUX;
pipeline_mux #(.WIDTH(18), .RSTTYPE(RSTTYPE)) B1REG_pipeline (.in(B1IN), .sel(B1REG),
.CE(CEB), .clk(clk), .rst(RSTB), .out(B1MUX));
assign BCOUT = B1MUX;   //The output of BCOUT to be cascaded in other DSPs

wire [17:0] A0MUX;
pipeline_mux #(.WIDTH(18), .RSTTYPE(RSTTYPE)) A0REG_pipeline (.in(A), .sel(A0REG),
.CE(CEA), .clk(clk), .rst(RSTA), .out(A0MUX));
wire [17:0] A1MUX;
pipeline_mux #(.WIDTH(18), .RSTTYPE(RSTTYPE)) A1REG_pipeline (.in(A0MUX), .sel(A1REG),
.CE(CEA), .clk(clk), .rst(RSTA), .out(A1MUX));

//Multiplier stage
wire [35:0] multiplier;
assign multiplier = A1MUX * B1MUX;
pipeline_mux #(.WIDTH(36), .RSTTYPE(RSTTYPE)) MREG_pipeline (.in(multiplier), .sel(MREG),
.CE(CEM), .clk(clk), .rst(RSTM), .out(M));

//The concatenation of D A B ports
wire [47:0] D_A_B_concatebated;
assign D_A_B_concatebated = {DMUX[11:0], A1MUX[17:0], B1MUX[17:0]};
```

```
//The X-MUX
wire [47:0] XMUX;
assign XMUX = (opmodeMUX[1:0] == 3)? D_A_B_concatebated : (opmodeMUX[1:0] == 2)? PCOUT
: (opmodeMUX[1:0] == 1)? {12'b0, M} : 0;

wire [47:0] CMUX;
pipeline_mux #(.WIDTH(48), .RSTTYPE(RSTTYPE)) CREG_pipeline (.in(C), .sel(CREG),
.CE(CEC), .clk(clk), .rst(RSTC), .out(CMUX));

//The Z-MUX
wire [47:0] ZMUX;
assign ZMUX = (opmodeMUX[3:2] == 3)? CMUX : (opmodeMUX[3:2] == 2)? PCOUT :
(opmodeMUX[3:2] == 1)? PCIN : 0;

//The selection of the CARRYIN signal
wire CARRYIN_SEL;
assign CARRYIN_SEL = (CARRYINSEL  == "OPMODE5")? opmodeMUX[5] : (CARRYINSEL  ==
"CARRYIN")? CARRYIN : 0;
wire CARRYINMUX;
pipeline_mux #(.WIDTH(1), .RSTTYPE(RSTTYPE)) CYI_pipeline (.in(CARRYIN_SEL),
.sel(CARRYINREG), .CE(CECARRYIN), .clk(clk), .rst(RSTCARRYIN), .out(CARRYINMUX));

//Post - adder / subtracter
wire [47:0] adder2;
wire CYO;           //CARRYOUT of the post - added / subtracter
assign {CYO, adder2} = (opmodeMUX[7])? ZMUX - (XMUX + CARRYINMUX) : ZMUX + XMUX +
CARRYINMUX;

pipeline_mux #(.WIDTH(1), .RSTTYPE(RSTTYPE)) CYO_pipeline (.in(CYO), .sel(CARRYOUTREG),
.CE(CECARRYIN), .clk(clk), .rst(RSTCARRYIN), .out(CARRYOUT));
assign CARRYOUTF = CARRYOUT;

pipeline_mux #(.WIDTH(48), .RSTTYPE(RSTTYPE)) PREG_pipeline (.in(adder2), .sel(PREG),
.CE(CEP), .clk(clk), .rst(RSTP), .out(P));
assign PCOUT = P;
endmodule
```

The TestBench:

```
module DSP48A1_tb ();
parameter A0REG = 0;
parameter A1REG = 1;
parameter B0REG = 0;
parameter B1REG = 1;
parameter CREG = 1;
parameter DREG = 1;
parameter MREG = 1;
parameter PREG  = 1;
parameter CARRYINREG = 1;
parameter CARRYOUTREG = 1;
parameter OPMODEREG = 1;
parameter CARRYINSEL  = "OPMODE5";
parameter B_INPUT = "DIRECT";
parameter RSTTYPE = "SYNC";

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

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

DSP48A1 #( A0REG, A1REG, B0REG,
B1REG, CREG, DREG, MREG, PREG ,
CARRYINREG, CARRYOUTREG,  OPMODEREG, CARRYINSEL,
B_INPUT, RSTTYPE) dut
(A, B, BCIN, C, D, opmode,
CARRYIN, CEA, CEB, CEC, CECARRYIN, CED, CEM,
CEOPMODE, CEP, clk, RSTA, RSTB, RSTC, RSTCARRYIN,
RSTD, RSTM, RSTOPMODE, RSTP, M, P, CARRYOUT, CARRYOUTF,
BCOUT, PCIN, PCOUT);

initial begin
    clk = 0;
    forever
        #10 clk = ~clk;
end
```

```
initial begin

    //Test the resets
    RSTA = 1;
    RSTB = 1;
    RSTC = 1;
    RSTCARRYIN = 1;
    RSTD = 1;
    RSTM = 1;
    RSTOPMODE = 1;
    RSTP = 1;
    @(negedge clk);
    if (P != 0) begin
        $display("there was an error");
        $stop;
    end

    //enable all the control signals
    RSTA = 0;           CEA = 1;
    RSTB = 0;           CEB = 1;
    RSTC = 0;           CEC = 1;
    RSTCARRYIN = 0;     CECARRYIN = 1;
    RSTD = 0;           CED = 1;
    RSTM = 0;           CEM  = 1;
    RSTOPMODE = 0;      CEOPMODE = 1;
    RSTP = 0;           CEP = 1;

    //Testing the pre and post addders and the multiplier
    opmode = 8'b0001_1101;
    repeat (10)begin
        A = $urandom_range(100,3000);
        B = $urandom_range(100,3000);
        BCIN = $urandom_range(100,3000);
        C = $urandom_range(100,3000);
        D = $urandom_range(100,3000);
        CARRYIN = $random;
        PCIN = $urandom_range(100,3000);
        repeat(4) @(negedge clk);
        if (P != (C + (A * (D + B)))) begin
            $display("there was an error");
            $stop;
        end
    end
```

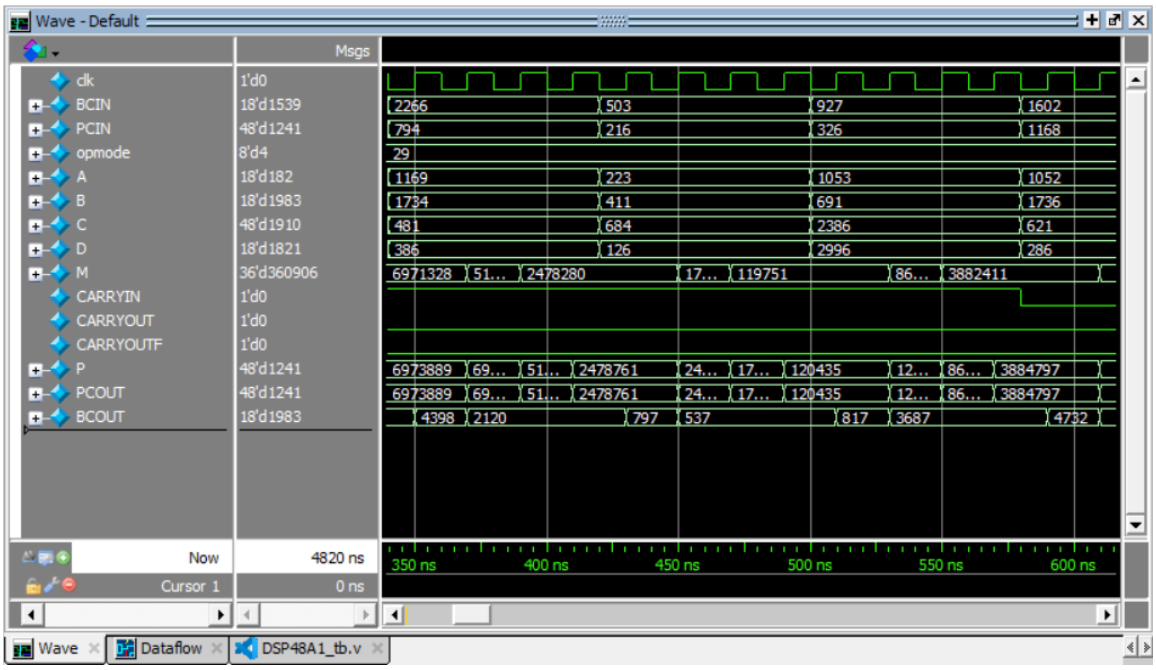
```
    //Testing the port C
    opmode = 8'b0001_1100;
    repeat (10)begin
        A = $urandom_range(100,3000);
        B = $urandom_range(100,3000);
        BCIN = $urandom_range(100,3000);
        C = $urandom_range(100,3000);
        D = $urandom_range(100,3000);
        CARRYIN = $random;
        PCIN = $urandom_range(100,3000);
        repeat(4) @(negedge clk);
        if (P != C) begin
            $display("there was an error");
            $stop;
        end
    end

    //Testing to skip the post adder
    opmode = 8'b0001_0001;
    repeat (10)begin
        A = $urandom_range(100,3000);
        B = $urandom_range(100,3000);
        BCIN = $urandom_range(100,3000);
        C = $urandom_range(100,3000);
        D = $urandom_range(100,3000);
        CARRYIN = $random;
        PCIN = $urandom_range(100,3000);
        repeat(4) @(negedge clk);
        if (P != (A * (D + B))) begin
            $display("there was an error");
            $stop;
        end
    end

    //Testing to skip the pre adder
    opmode = 8'b0000_0001;
    repeat (10)begin
        A = $urandom_range(100,3000);
        B = $urandom_range(100,3000);
        BCIN = $urandom_range(100,3000);
        C = $urandom_range(100,3000);
        D = $urandom_range(100,3000);
        CARRYIN = $random;
        PCIN = $urandom_range(100,3000);
```

```
        repeat(4) @(negedge clk);
        if (P != (A * B)) begin
            $display("there was an error");
            $stop;
        end
    end

    //Testing the concatenated port
    opmode = 8'b0000_0011;
    repeat (10)begin
        A = $urandom_range(100,3000);
        B = $urandom_range(100,3000);
        BCIN = $urandom_range(100,3000);
        C = $urandom_range(100,3000);
        D = $urandom_range(100,3000);
        CARRYIN = $random;
        PCIN = $urandom_range(100,3000);
        repeat(4) @(negedge clk);
        if (P != {D[11:0], A[17:0], B[17:0]}) begin
            $display("there was an error");
            $stop;
        end
    end

    //Testing the cascaded PCIN
    opmode = 8'b0000_0100;
    repeat (10)begin
        A = $urandom_range(100,3000);
        B = $urandom_range(100,3000);
        BCIN = $urandom_range(100,3000);
        C = $urandom_range(100,3000);
        D = $urandom_range(100,3000);
        CARRYIN = $random;
        PCIN = $urandom_range(100,3000);
        repeat(4) @(negedge clk);
        if (P != PCIN) begin
            $display("there was an error");
            $stop;
        end
    end
    $stop;
end
endmodule
```

DSP.do File:

```
vlib work
vlog DSP48A1.v DSP48A1_tb.v pipeline_mux.v
vsim -voptargs=+acc work.DSP48A1_tb
add wave -position insertpoint  \
sim:/DSP48A1_tb/A \
sim:/DSP48A1_tb/B \
sim:/DSP48A1_tb/BCIN \
sim:/DSP48A1_tb/D \
sim:/DSP48A1_tb/C \
sim:/DSP48A1_tb/PCIN \
sim:/DSP48A1_tb/opmode \
sim:/DSP48A1_tb/CARRYIN \
sim:/DSP48A1_tb/clk \
sim:/DSP48A1_tb/P \
sim:/DSP48A1_tb/PCOUT \
sim:/DSP48A1_tb/BCOUT \
sim:/DSP48A1_tb/M \
sim:/DSP48A1_tb/CARRYOUT \
sim:/DSP48A1_tb/CARRYOUTF
run -all
#quit -sim
```

The waveform:

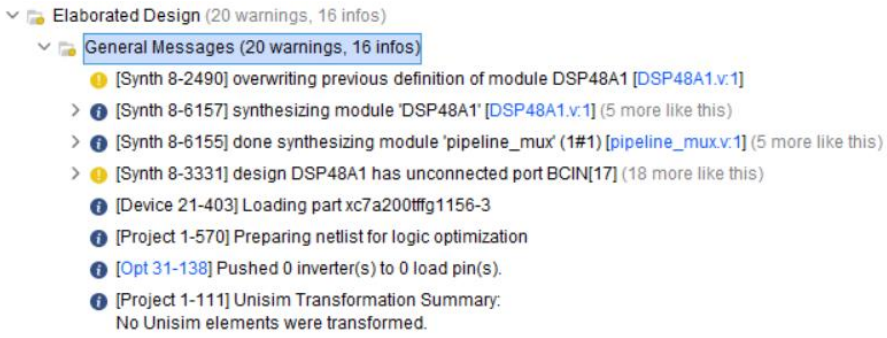


Constraint file:

```
# 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]


## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]
```

Elaboration:

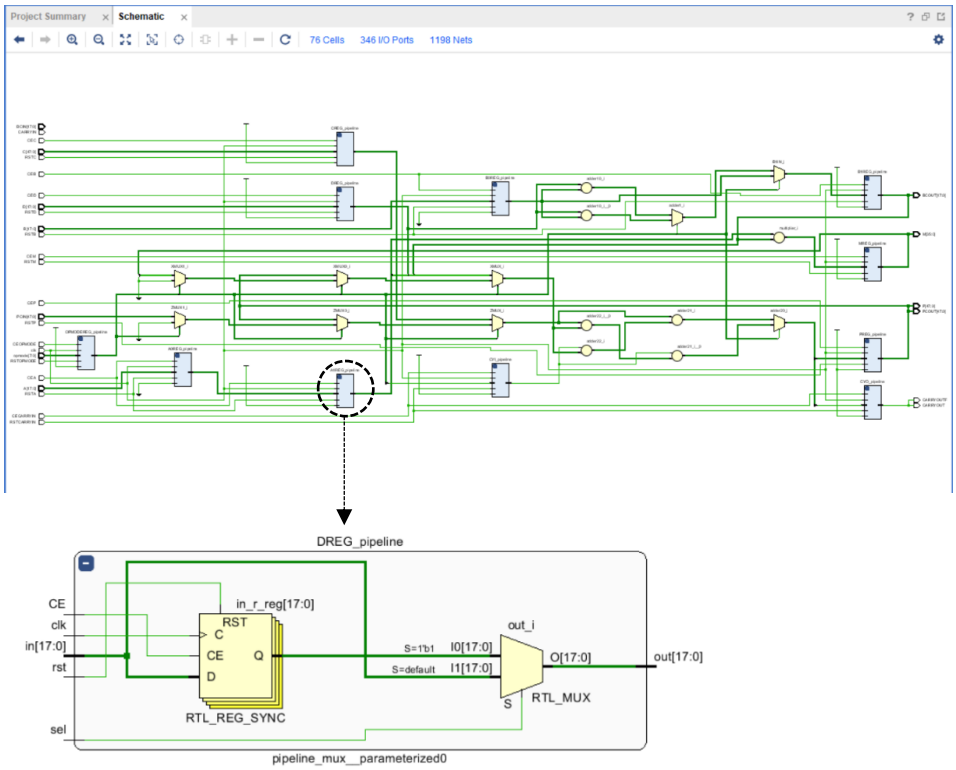
- Messages



Regarding the warning messages:

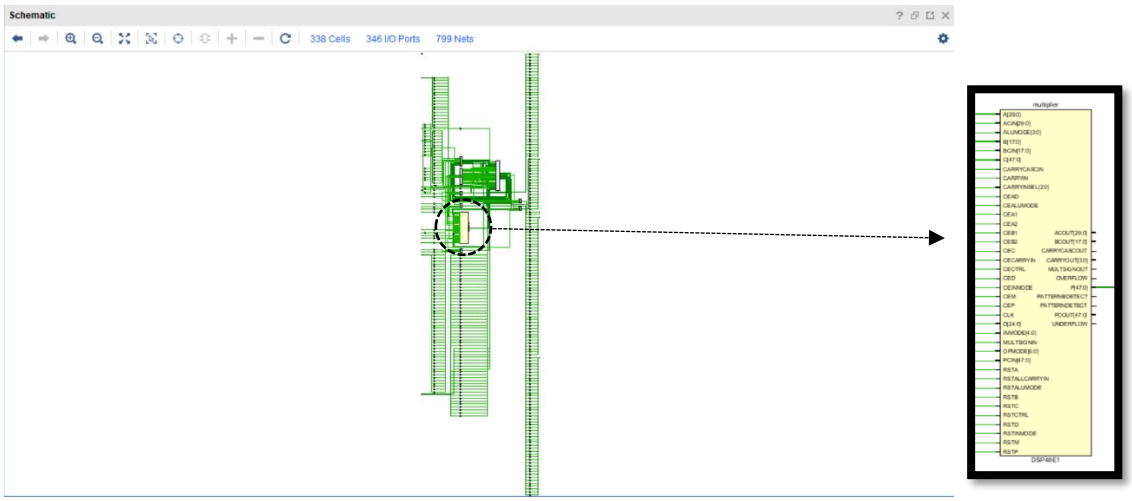
- Since our module name is the same as the VIvado module, it warns us about overwriting the same module
- As for the (BCIN) this occurred because the port was not driven since we designed a single stage slice

- Schematic

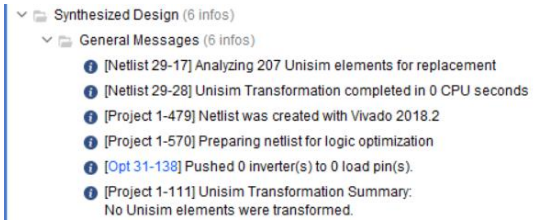

Project Summary
Schematic
76 Cells
346 I/O Ports
1198 Nets
DREG_pipeline
CE
clk
in[17:0]
rst
sel
in_r_reg[17:0]
RST
C
CE
D
Q
RTL_REG_SYNC
S=1'b1
S=default
I0[17:0]
I1[17:0]
out_i
O[17:0]
S
RTL_MUX
out[17:0]
pipeline_mux__parameterized0

Synthesis:

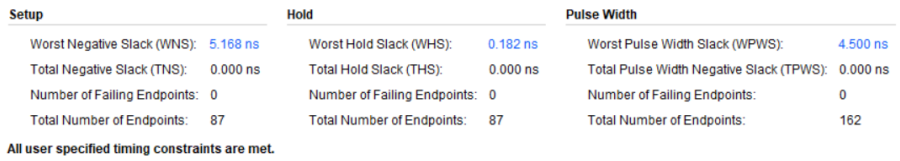
- Schematic



- Messages



- 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

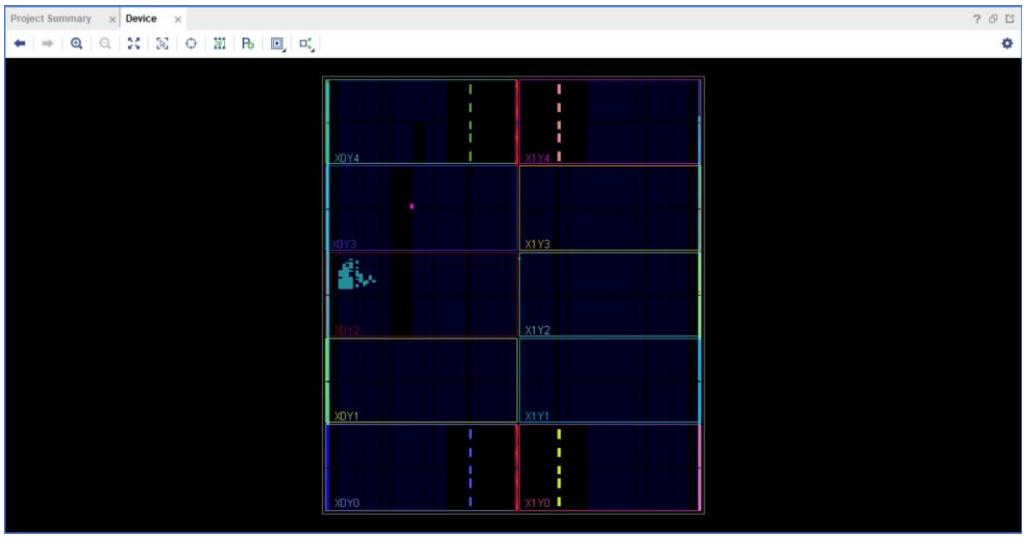
All user specified timing constraints are met.

- Utilization report

Name	Slice LUTs (134600)	Slice Registers (269200)	DSPs (740)	Bonded IOB (500)	BUFGCTRL (32)
N DSP48A1	230	160	1	327	1
A1REG_pipeline (pipel...	0	18	0	0	0
B1REG_pipeline (pipel...	0	18	0	0	0
CREG_pipeline (pipeli...	0	48	0	0	0
CYI_pipeline (pipeline...	1	1	0	0	0
CYO_pipeline (pipelin...	0	1	0	0	0
DREG_pipeline (pipeli...	0	18	0	0	0
OPMODEREG_pipelin...	228	8	0	0	0
PREG_pipeline (pipeli...	0	48	0	0	0

Implementation:

- Device



- Messages

Implemented Design (9 infos)
- General Messages (9 infos)
  - [Netlist 29-17] Analyzing 207 Unisim elements for replacement
  - [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
  - [Project 1-479] Netlist was created with Vivado 2018.2
  - [Project 1-570] Preparing netlist for logic optimization
  - [Timing 38-478] Restoring timing data from binary archive.
  - [Timing 38-479] Binary timing data restore complete.
  - [Project 1-856] Restoring constraints from binary archive.
  - [Project 1-853] Binary constraint restore complete.
  - [Project 1-111] Unisim Transformation Summary:
    No Unisim elements were transformed.

- Timing report

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.813 ns	Worst Hold Slack (WHS):	0.266 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.

- 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)
DSP48A1	229	179	108	229	52	1	327	1
A1REG_pipeline (pipel...	0	18	5	0	0	0	0	0
B1REG_pipeline (pipel...	0	36	16	0	0	0	0	0
CREG_pipeline (pipeli...	0	48	18	0	0	0	0	0
CYI_pipeline (pipeline...	1	1	1	1	1	0	0	0
CYO_pipeline (pipelin...	0	2	2	0	0	0	0	0
DREG_pipeline (pipeli...	0	18	11	0	0	0	0	0
OPMODEREG_pipelin...	228	8	70	228	0	0	0	0
PREG_pipeline (pipeli...	0	48	12	0	0	0	0	0