

Report Lab 4

Parham Gilani – 400101859

Sadra Khanjari – 400101107

Lab 1:

This is the Complex Multiplier code that I have used in my project.

```
module ComplexMultiplier(  
    input wire clk,  
    input wire reset,  
    input wire [15:0] inputMultiplier1,  
    input wire [15:0] inputMultiplier2,  
    output reg [33:0] multiplicationResult  
);  
  
    wire [7:0] real_1, real_2, imag1, imag2 ;  
    wire [15:0] multrr,multri,multir,multii;  
    wire signed [16:0] adderr,adderi;  
    wire [33:0] result;  
  
    assign real_1 = inputMultiplier1[15]? ~inputMultiplier1[15:8] + 1 : inputMultiplier1[15:8];  
    assign imag1 = inputMultiplier1[7]? ~inputMultiplier1[7:0] + 1 : inputMultiplier1[7:0];  
    assign real_2 = inputMultiplier2[15]? ~inputMultiplier2[15:8] + 1 : inputMultiplier2[15:8];  
    assign imag2 = inputMultiplier2[7]? ~inputMultiplier2[7:0] + 1 : inputMultiplier2[7:0];  
  
    assign multrr = (inputMultiplier1[15] ^ inputMultiplier2[15])? ~(real_1 * real_2) + 1 : real_1 * real_2;  
    assign multri = (inputMultiplier1[15] ^ inputMultiplier2[7])? ~(real_1 * imag2) + 1 : real_1 * imag2;  
    assign multir = (inputMultiplier1[7] ^ inputMultiplier2[15])? ~(imag1 * real_2) + 1 : imag1 * real_2;  
    assign multii = (inputMultiplier1[7] ^ inputMultiplier2[7])? ~(imag1 * imag2) + 1 : imag1 * imag2;  
  
    assign adderr = $signed(multrr) - $signed(multii);  
    assign adderi = $signed(multir) + $signed(multri);  
  
    assign result = {adderr, adderi};  
  
    always @(posedge clk ) begin  
        if (!reset) multiplicationResult <= result;  
        else multiplicationResult <= 0;  
    end  
endmodule
```

After synthesizing no errors was found.

gn

Implementation Simulation

Design Overview

Summary

IOB Properties

Module Level Utilization

Timing Constraints

Pinout Report

Clock Report

Static Timing

Errors and Warnings

Parser Messages

Synthesis Messages

Translation Messages

Map Messages

Place and Route Messages

Timing Messages

Bitgen Messages

All Implementation Messages

Detailed Reports

Synthesis Report

Translation Report

Map Report

Place and Route Report

Post-PAR Static Timing Report

Power Report

Bitgen Report

Secondary Reports

Design Properties

Enable Message Filtering

Optional Design Summary Contents

Show Clock Report

Show Failing Constraints

Show Warnings

Show Errors

View: Implementation Simulation

Hierarchy

lab1

xc6slx45-3csg324

ComplexMultiplier (ComplexMultiplier.v)

No Processes Running

Processes: ComplexMultiplier

Design Summary/Reports

Design Utilities

User Constraints

Synthesize - XST

View RTL Schematic

View Technology Schematic

Check Syntax

Generate Post-Synthesis Simulation Model

Implement Design

Generate Programming File

Configure Target Device

Analyze Design Using ChipScope

ComplexMultiplier Project Status

Project File:	lab1.xise	Parser Errors:	No Errors
Module Name:	ComplexMultiplier	Implementation State:	Synthesized
Target Device:	xc6slx45-3csg324	Errors:	No Errors
Product Version:	ISE 14.7	Warnings:	8 Warnings (8 new)
Design Goal:	Balanced	Routing Results:	
Design Strategy:	Ylimn Default (unlocked)	Timing Constraints:	
Environment:	System Settings	Final Timing Score:	

Device Utilization Summary (estimated values)

Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	168	27288	0%
Number of fully used LUT-FF pairs	0	168	0%
Number of bonded IOBs	68	218	31%
Number of BUFG/BUFGCTRLs	1	16	6%
Number of DSP48A1s	4	58	6%

Detailed Reports

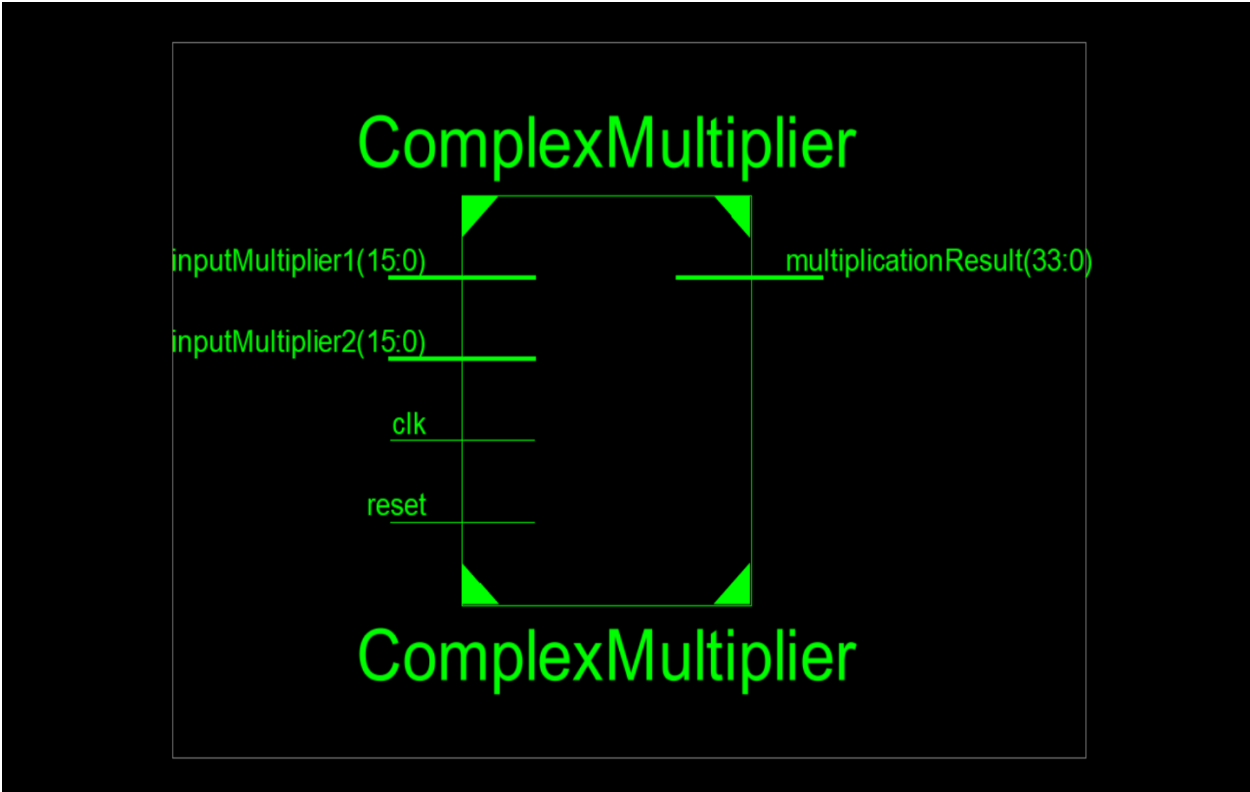
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Sun Jun 2 21:24:37 2024	0	8 Warnings (8 new)	0
Translation Report					
Map Report					
Place and Route Report					
Power Report					
Post-PAR Static Timing Report					
Bitgen Report					

Secondary Reports

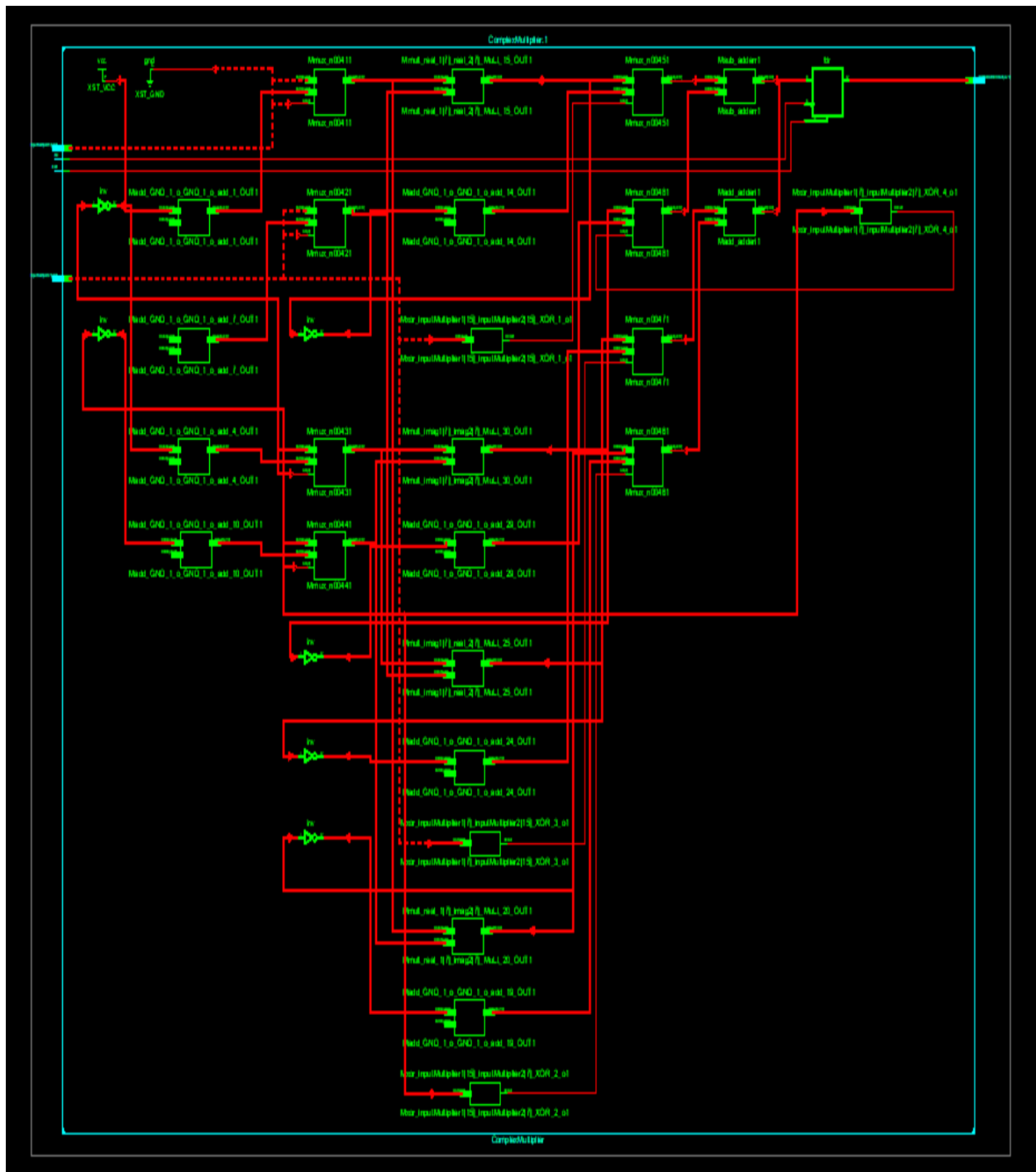
Report Name	Status	Generated
-------------	--------	-----------

Date Generated: 06/02/2024 - 21:26:20

And the RTL Schematic of the code is this:



And this is the RTL circuit details:



Lab 2:

The testbench related to the last Verilog code is shown blow.

```
`timescale 1ns/1ns

module ComplexMultiplier_tb();

    wire [33:0] out;
    reg clk, reset;
    reg [15:0] input1, input2;

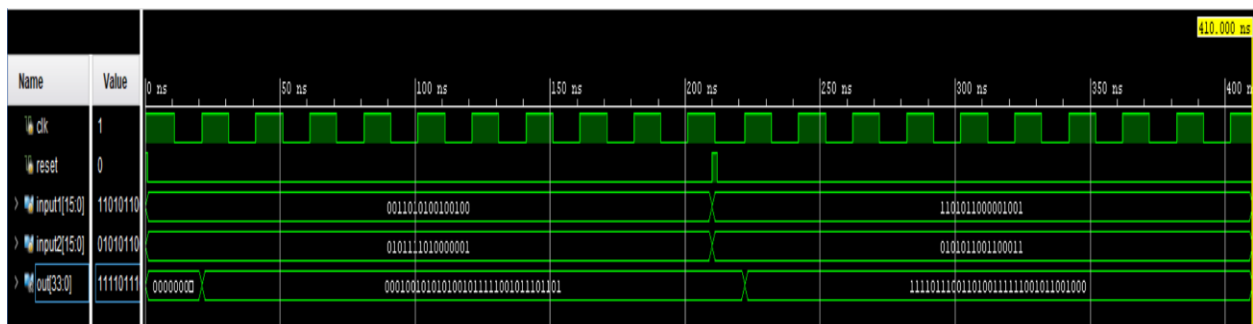
    always @(clk) begin
        if (reset) #1 reset<=0;
        #10 clk <= ~clk;
    end

    initial begin
        clk = 1;
        input1 = $random();
        input2 = $random();
        reset = 1;
        #200;
        $display("input1 = %b" , input1);
        $display("input2 = %b" , input2);
        $display("out = %b" , out);
        #10
        input1 = $random();
        input2 = $random();
        reset = 1;
        #200;
        $display("input1 = %b" , input1);
        $display("input2 = %b" , input2);
        $display("out = %b" , out);
        $stop;
    end

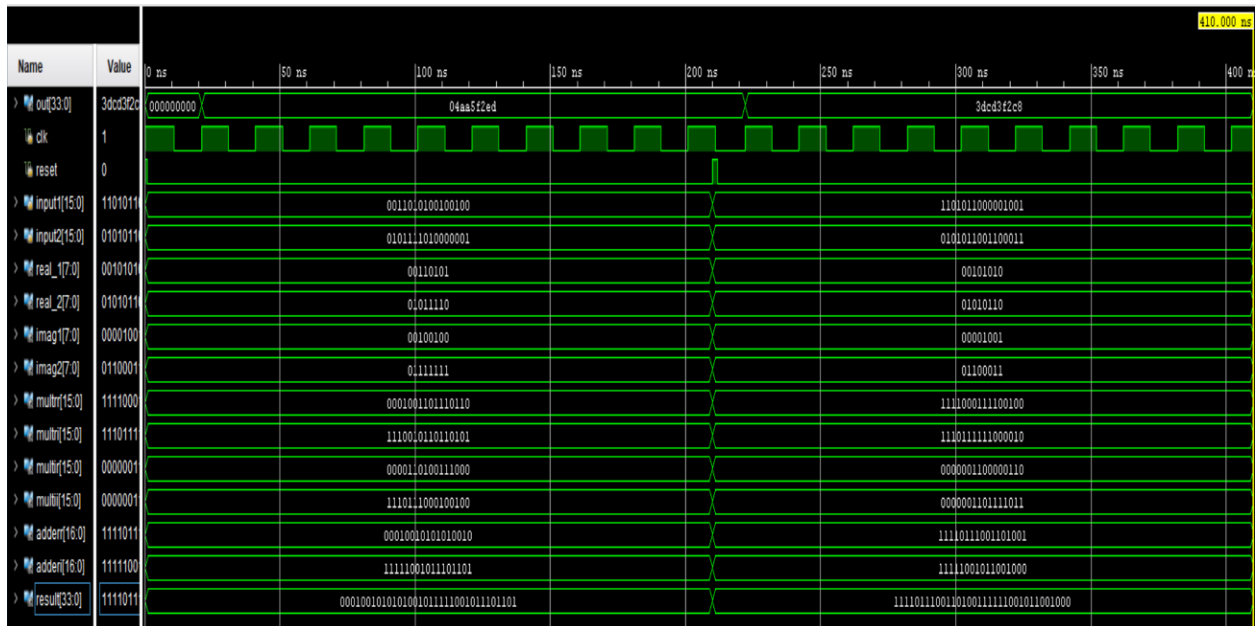
    ComplexMultiplier uut (
        .clk(clk),
        .reset(reset),
        .inputMultiplier1(input1),
        .inputMultiplier2(input2),
        .multiplicationResult(out)
    );

endmodule
```

And the wave form of it is this.

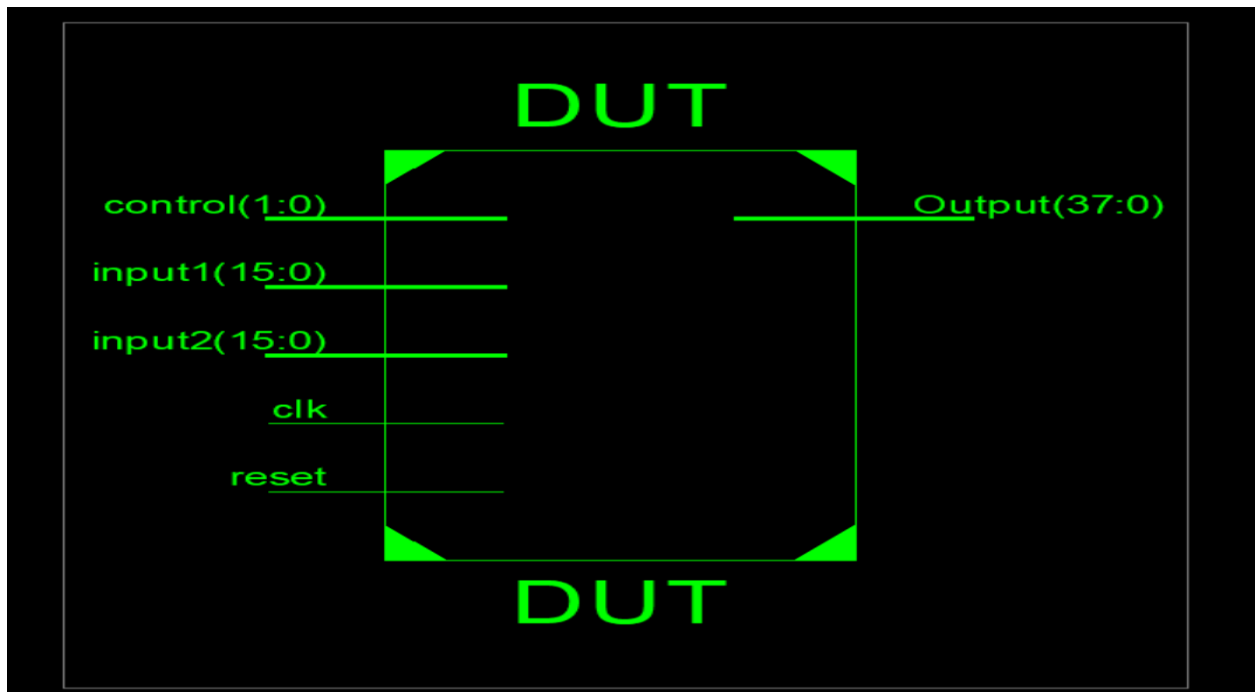


But this wave form contains all wires and registers in the module.



Lab 3:

In this part we must use our previous codes and use several IP cores like CORDIC and Multiply Adder to be instantiated in DUT code. Just like before I have got the RTL Schematics of the code.



Name	Value	0 ns	200 ns	400 ns	600 ns	800 ns	1,000 ns	
Output[37:0]	10	0		5		0	10	
clk	1							
reset	0							
control[1:0]	0	0						
input1[15:0]	6	3		6				
input2[15:0]	8	4		8				
test[7:0]	-1	-1						

[illegible][illegible]

Name	Value	0 ns	100 ns	200 ns	300 ns	400 ns	500 ns	600 ns	645.000 ns	700 ns	
Output[37:0]	00000000000000000001										
clk	1										
reset	0										
control[1:0]	11										
input1[15:0]	0000000011111111										
input2[15:0]	0000000000000001										
test[7:0]	11111111										

Gets the third Complex Multiplier output.

Lab 4:

In this section we must add constraints to the project. After I saved the constraint file into “DUT.ucf” and close the constraint editor.

```
#Created by Constraints Editor (xc6slx45t-csg324-3) - 2024/06/03
NET "clk" TNM_NET = clk;
TIMESPEC TS_clk = PERIOD "clk" 30 ns HIGH 50% INPUT_JITTER 60 ps;
NET "control<0>" OFFSET = IN 20 ns VALID 20 ns BEFORE "clk" RISING;
INST "Output<0>" TNM = Output1_Group;
INST "Output<1>" TNM = Output1_Group;
INST "Output<2>" TNM = Output1_Group;
INST "Output<3>" TNM = Output1_Group;
INST "Output<4>" TNM = Output1_Group;
INST "Output<5>" TNM = Output1_Group;
INST "Output<6>" TNM = Output1_Group;
INST "Output<7>" TNM = Output1_Group;
INST "Output<8>" TNM = Output1_Group;
INST "Output<9>" TNM = Output1_Group;
INST "Output<10>" TNM = Output1_Group;
INST "Output<11>" TNM = Output1_Group;
INST "Output<12>" TNM = Output1_Group;
INST "Output<13>" TNM = Output1_Group;
INST "Output<14>" TNM = Output1_Group;
INST "Output<15>" TNM = Output1_Group;
INST "Output<16>" TNM = Output1_Group;
INST "Output<17>" TNM = Output1_Group;
INST "Output<18>" TNM = Output1_Group;
INST "Output<19>" TNM = Output1_Group;
INST "Output<20>" TNM = Output1_Group;
INST "Output<21>" TNM = Output1_Group;
INST "Output<22>" TNM = Output1_Group;
INST "Output<23>" TNM = Output1_Group;
INST "Output<24>" TNM = Output1_Group;
INST "Output<25>" TNM = Output1_Group;
INST "Output<26>" TNM = Output1_Group;
INST "Output<27>" TNM = Output1_Group;
INST "Output<28>" TNM = Output1_Group;
INST "Output<29>" TNM = Output1_Group;
INST "Output<30>" TNM = Output1_Group;
INST "Output<31>" TNM = Output1_Group;
INST "Output<32>" TNM = Output1_Group;
INST "Output<33>" TNM = Output1_Group;
INST "Output<34>" TNM = Output1_Group;
INST "Output<35>" TNM = Output1_Group;
INST "Output<36>" TNM = Output1_Group;
INST "Output<37>" TNM = Output1_Group;
TIMEGRP "Output1_Group" OFFSET = OUT 20 ns AFTER "clk";

NET "Clk" LOC = L16;
NET "Reset" LOC = A10;
```

This was the constraint file which I have edited and add 2 lines and the end of it which locates the location of constraints in UCF file.

Now we want to enter constraints using an XST file. We create a new text file in ISE and write this code in it and save it as a .XCF file.

```
BEGIN MODEL "DUT"
NET "Clk" clock_signal = yes;
END;
```

THE END