

Sharif University of Technology Department of Computer Engineering

# System on Chip Desing Projects RTL Design Using IP Cores

Mahbod Afarin

November 2016

# **1- Overview**

Our goal is to design a **complex ALU** in two different ways: as a **regular design** and as an **IP core**. Ultimately, we will compare the two implementations in terms of **area**, **delay**, and **power consumption**.

This ALU operates on matrices, where each matrix element is **16 bits** wide: **8 bits for the real part** and **8 bits for the imaginary part** of the complex number. As a result, this project consists of **three main parts**:

- 1. **Regular Design** A custom implementation of the ALU.
- 2. IP Core Design Using a pre-designed and optimized IP core.
- 3. **Comparison** Evaluating and comparing both designs.

As we know, **IP cores** are typically developed by expert engineers and are expected to be highly optimized in every aspect. The objective of this project is to **demonstrate the efficiency and optimization** of the IP core implementation.

For this project, we use the **Verilog** language and the **Xilinx ISE** software for simulation and synthesis. Additionally, we target the **Virtex-7 FPGA** for implementation.

# 2- Non-IP Core Design for Complex ALU

Our goal is to design an **ALU for complex numbers**. This ALU performs **three operations**: **addition**, **subtraction**, and **multiplication**.

- The **addition and subtraction** operations are handled within the Complex\_ALU module.
  - When OP = 00, the ALU performs **addition**.
  - When OP = 01, it performs **subtraction**.
  - When OP = 10, it performs **multiplication**.
- The **multiplication** operation is carried out in a separate module called complex\_mul, which is instantiated inside the Complex\_ALU module.

In the next section, we will **replace the complex\_mul module with an IP Core** to evaluate its performance and compare the results.

# 2-1- Functional Verification of the Complex ALU Without IP Core

For this purpose, we use the **ISE software**. After writing the **testbench**, we apply the following **input values** to the circuit and observe the corresponding **output results**.

$$A = \begin{bmatrix} 2+4j & 1+3j & 2+4j \\ 3+j & 2+2j & 6+7j \\ 7+2j & 2+7j & j \end{bmatrix}$$

$$B = \begin{bmatrix} 7+4j & 1+4j & 3+2j \\ 2+j & 1+7j & 2+2j \\ 2+3j & 3+2j & 1 \end{bmatrix}$$

### 2-1-1- Addition Operation Verification

The result of adding the two matrices above is as follows.

$$\begin{bmatrix} 2+4j & 1+3j & 2+4j \\ 3+j & 2+2j & 6+7j \\ 7+2j & 2+7j & j \end{bmatrix} + \begin{bmatrix} 7+4j & 1+4j & 3+2j \\ 2+j & 1+7j & 2+2j \\ 2+3j & 3+2j & 1 \end{bmatrix} = \begin{bmatrix} 9+8j & 2+7j & 5+6j \\ 5+2j & 3+9j & 8+9j \\ 9+5j & 5+9j & 1+1j \end{bmatrix}$$

The simulation results are shown below, confirming the correctness of the addition operation.

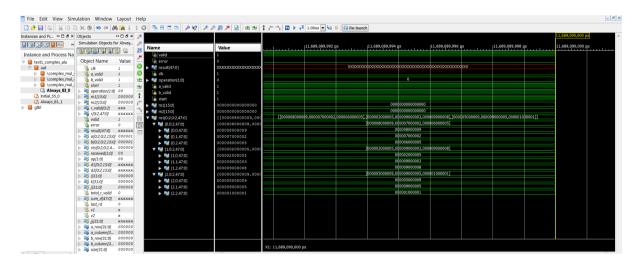


Figure 1: Simulation of the ALU Without IP Core for the Addition Operation

### 2-1-2- Subtraction Operation Verification

The result of subtracting the two matrices above is as follows.

$$\begin{bmatrix} 2+4j & 1+3j & 2+4j \\ 3+j & 2+2j & 6+7j \\ 7+2j & 2+7j & j \end{bmatrix} - \begin{bmatrix} 7+4j & 1+4j & 3+2j \\ 2+j & 1+7j & 2+2j \\ 2+3j & 3+2j & 1 \end{bmatrix} = \begin{bmatrix} -5 & -j & -1+2j \\ 1 & 1-5j & 4+5j \\ 5-j & -1+5j & -1+j \end{bmatrix}$$

The simulation results are shown below, confirming the correctness of the subtraction operation.

🔜 File Edit View Sim	ulation Window	Layout H	elp														- 8 ×
🗋 🌶 🖬 😓 🕺 🛍 🛙	≥×® ≈ ≈ 0	4 <b>3</b> 34↓î	0 5 8	🗆 🕾 🥬 🌮 J	8 🖉 🏓 🗟 🛬 🕿	10	1 🖬 🕨 🕫	1.00us 🔻 🔙 🛙	🗔 Re-launch								
Instances and Pr •• 🗆 & ×	Objects	+□∂× ,	÷													46,131,470,000 ps	~
	Simulation Objects f		8		Malua			46,131,469,992 ps		46,131,469,994 p	ns	46,131,469,996	ns	46,131,469,998	ns	46,131,470,000 ps	
Instance and Process Na		15 🔛 🖇	Name		Value						F		F		7	Provide States	
	Object Name	Value	<ul> <li>Us valid</li> <li>Us error</li> </ul>		0												
V sti_complex_alu			Image: Second		*				20000000	000000000000000000000000000000000000000			000				
uut	Us cik	1		ntevol	1				100000000				001				
complex_mul_	b_valid		b b 🐝 ope								1						
complex mul	1 start		r 🖡 👘 ope		1						-						
G Always 83 0	operation[1:0]		b_va		-												
C Initial_55_0	m1(15:0)	000000	la b_va		1												
C Always_81_1	▷ ➡ m2[15:0]	000000	<ul> <li>Stat</li> <li>Stat</li> <li>m1f</li> </ul>		± 000000000000000000000000000000000000					0000	00000000000000000						
⊳ 📒 glbl	▷ ➡ r_valid[0:2]	xxx	1 🖡 📷 m1(		000000000000000000000000000000000000000						000000000000000000000000000000000000000						
	▷ N 10:2,47:0]	XXXXXX		0:2,0:2,47:0]	110000000000000000000000000000000000000	Thomas	000000000000000000000000000000000000000	000000011111111				000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000101111111		
	La valid	0		0:2,0:2,47:0] 10.0:2.47:01	[00000000000000000000000000000000000000		1000000000000000	00000001111111			fffff000000.0000			00000000000000000	00010111111		
	error	0		[0,0:2,47:0]	000000fffffb						000000000000000000000000000000000000000						
	result[47:0] a(0:2,0:2,15:0)	XXXXXX		0,0,47:0]	ffffff000000						11111000000						
	▷ ➡ b(0:2,0:2,15:0)			[0,1,47:0]	000002ffffff						000002ffffff						
	res[0:2,0:2,10:0]			[1,0:2,47:0]	[000002111111 [0000000000001,fff1					000000000000000000000000000000000000000	ffffb000001.0000	050000041					
	received[1:0]				0000000000001, 1111						000000000000000000000000000000000000000	03000004					
	▷ ➡ op/1:01	01		[1,0,47:0]							ffffb000001						
	d1[0:2,15:0]	XXXXXX		[1,1,47:0]	fffffb000001 000005000004						00005000004						
	▷ N d2[0:2,15:0]	XXXXXX		<b>[1,2,47:0]</b>							00003000004	1.66666					
	⊳ 📷 i(31:0)	000000		[2,0:2,47:0]	[ffffff000005,0000						fffff00005	40000					
	⊳ 💕 k[31:0]	000000		[2,0,47:0]	ffffff000005						000005ffffff						
	⊳ 📸 j[31:0]	000000		[2,1,47:0]	000005ffffff						000005000						
	15 total_r_valid	0		[2,2,47:0]	000001ffffff						0000111111						
	▷ ➡ sum_d[47:0]	xxxxxx 0															
	15 vi	x															
	14 v2	x															
	▷ → jj[31:0]	XXXXXX															
	> a_row[31:0]	000000															
	▷ ➡ a_column(3	000000															
	b_row[31:0]	000000															
	b_column(3	000000				¥1• 4	6,131,470,000 p	-									
	b ize[31:0]	000000				A1. 4	0,101,470,000 p.										

Figure 2: Simulation of the ALU Without IP Core for the Subtraction Operation

# 2-1-3- Multiplication Operation Verification

The result of multipication the two matrices above is as follows.

[2 + 4 <i>j</i>	1 + 3 <i>j</i>	2 + 4 <i>j</i> ן	[7 + 4 <i>j</i>	1 + 4 <i>j</i>	3 + 2 <i>j</i> ך	[-11 + 57 <i>j</i>	-36 + 38 <i>j</i>	-4 + 28j
3 + j	2 + 2j	$6+7j \times$	2+j	1 + 7j	2 + 2j =	= 10 + 57 <i>j</i>	-9 + 62 <i>j</i>	13 + 24 <i>j</i>
[7 + 2j]	2 + 7 <i>j</i>	j ]	L2 + 3 <i>j</i>	3 + 2 <i>j</i>	1 ]	$= \begin{bmatrix} -11 + 57j \\ 10 + 57j \\ 35 + 60j \end{bmatrix}$	-50 + 54 <i>j</i>	7 + 39 <i>j</i>

The simulation results are shown below, confirming the correctness of the multipication operation.

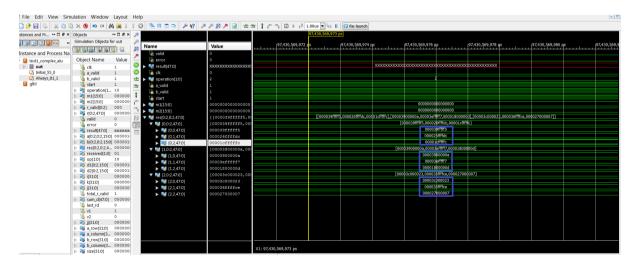


Figure 3: Simulation of the ALU Without IP Core for the Multiplication Operation

# 2-2- Synthesis of the ALU Without IP Core and Analysis of Its Power, Delay, and

### Area

In this section, we aim to obtain the **power**, **delay**, and **area** results for the **non-IP core design**.

# 2-2-1- Power Analysis

To analyze **power consumption**, we use the **XPower Analyzer** tool in **Xilinx ISE**. First, we need to generate a **VCD (Value Change Dump)** file. To do this, we add the following code snippet to the **testbench**, which enables the creation of the VCD file.

```
// The following code will generate a VCD file containing
// all of the nets in the instance t.uut. "t" is the
module name of the
// testfixture, "uut" is the instance name
// of the design being tested.
    initial begin
    $dumpfile ("invchn26.vcd"); // Change filename as
    appropriate.
    $dumpvars(1, t.uut);
    end
```

As a result, the **VCD file** is generated in the project's storage directory.

Next, from the Tools menu, we select XPower Analyzer.

Then, under the **File** tab, we choose **Open Design** and provide the appropriate input files to proceed with the analysis.

퉳 Open Design	×							
Required								
Design file	NCD File							
E_Tutorial/Examples/Full_Adder/Full_Adder.ncd								
Optional								
Settings file	XMA File							
I/Examples/Full_Adder/Ful	I_Adder_summary.xml							
Physical Constraints file	PCF File							
E_Tutorial/Examples/Full_	Adder/Full_Adder.pcf							
Simulation Activity file	VCD/SAIF File							
Strip Path Path i	inVCD/SAIF File							
Automatically run analy	vsis upon opening design							
Analysis Effort Level H	igh							
	OK Cancel							

Figure 4: Specifying the Required Inputs in the Open Design Section

The power consumption results in this case are shown in Figure 5. As illustrated, the dynamic power consumption is 0.002 W, the static power consumption is 0.177 W, and the total power consumption is 0.179 W.

	xmahbad Desktop/SC/Complex,ALU_withoutp/Comlex,ALU.ad [Table View]	x
E File Edit View Tools Help		) 2 X
🖻 🔂 🖉 🕲 🗎		
	A B CD E F G H IJ K L M N	
View	Device On-Chip Power (W) Used Available Utilization (%) Supply Summary Total Dynamic Quescent	
Color Servers     Color Servers     Color Active Relates     Control A	Concernance         Concernance	
Delaak Calculated	The Power Analysis is up to date. () Pisce mouse over the samelek for more detailed IRAM attractor.	
Views	S Table View	
Matched 41% (34/83) Matched 16% (13/81)		*
×		1
Console Report Warning E		
Ready		

Figure 5: Power Consumption Results of the System Without IP Core

# 2-3- Delay Analysis

To analyze the circuit's **delay**, we use the **delay report**.

This is shown in Figure 6.

As indicated in the figure, the **critical path delay** is **2.537 ns**.

ISE Project Navigator (P.20131013) - C:\Users\mahbod\Desktop	Complex_ALU_withoutip\Complex_AL	U_withoutip.xise - [Co	mlex_ALU.	syri 👘 🕐
File Edit View Project Source Process Tools Window	v Layout Help			
○ ※日間は、X ○ ○ X ら ○ N ※ アクタタク	8 🔺 🗟 🗄 🗉 🗣 🖉 🕨	x 📌 💡		
Design ↔ □ # × []				
Image: Wew: ● 御 Implementation ● @ Simulation       Image: Hierarchy       Image: Image		(frequency: 39 / destination	4.151MH ports:	z) 118 / 24
Complex_mul (complex_mulv)	Source: j_1 Destination: j_1	37ns (Levels o (FF) (FF) rising rising	I TOĞIÇ	= 12)
	Cell:in->out fa	Gate nout Delay		Logical Name (Net Name)
	FDR:C->Q LUT1:I0->O MUXCY:S->O MUXCY:CI->O MUXCY:CI->O MUXCY:CI->O MUXCY:CI->O MUXCY:CI->O	$\begin{array}{cccc} 3 & 0.236 \\ 1 & 0.043 \\ 1 & 0.238 \\ 1 & 0.013 \\ 1 & 0.013 \\ 1 & 0.013 \\ 1 & 0.013 \\ 1 & 0.013 \\ 1 & 0.013 \end{array}$	0.362 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000	j_1 (j) Mccmpar GND 1 o j(31] LessThan 107 o cy<0> rt (Mccmpar GND 1 o j(31] LessThan 107 o cy<0> rt) Mccmpar GND 1 o j(31] LessThan 107 o cy<0> (Mccmpar GND 1 o j(31] LessThan 107 o cy<0> Mccmpar GND 1 o j(31] LessThan 107 o cy<1> (Mccmpar GND 1 o j(31] LessThan 107 o cy<1> Mccmpar GND 1 o j(31] LessThan 107 o cy<2> (Mccmpar GND 1 o j(31] LessThan 107 o cy<2> Mccmpar GND 1 o j(31] LessThan 107 o cy<2> (Mccmpar GND 1 o j(31] LessThan 107 o cy<3) Mccmpar GND 1 o j(31] LessThan 107 o cy<3> (Mccmpar GND 1 o j(31] LessThan 107 o cy<3> Mccmpar GND 1 o j(31] LessThan 107 o cy<4> (Mccmpar GND 1 o j(31] LessThan 107 o cy<4> Mccmpar GND 1 o j(31] LessThan 107 o cy<4> (Mccmpar GND 1 o j(31] LessThan 107 o cy<4> Mccmpar GND 1 o j(31] LessThan 107 o cy<4> (Mccmpar GND 1 o j(31] LessThan 107 o cy<5>
No Processes Running     Processes: Comlex ALU	MUXCY:CI->O INV:I->O MUXCY:CI->O	8 0.151 1 0.054 0 0.013	0.339	<pre>Mcompar_GND 1 o j[31] LessThan 107 o cy&lt;6&gt; (GND 1 o j[31] LessThan 107 o inv) Mcompar_GND 1 o j[31] LessThan 107 o cy&lt;6&gt; inv1 INV 0 (GND 1 o j[31] LessThan 107 o inv inv) Mcompar_GND 1 o j[31] LessThan 107 o cy&lt;6&gt; inv1 INV 0 (GND 1 o j[31] LessThan 107 o inv inv)</pre>
Processes Commex/ALD     Processes Commex/Reports     Design Utilities     Create Schematic Symbol     View Command Line Loa File	MUXCY:CI->O XORCY:CI->O LUT6:I5->O FDR:D	0 0.013 1 0.262 1 0.043 -0.000	0.350	<pre>Mcount j_cyc0&gt; (Mcount j_cyc0&gt;) Mcount_j_xorcl&gt; (Mcount_j1) j_l_glue_set (j_glue_set) j_1</pre>
View Command Line Log File     View HDL Instantiation Template     View HDL Instantiation Template     View Constraints	Total	2.537ns		ns logic, 1.430ms route) logic, 56.3% route)

Figure 6: Delay Analysis of the Circuit Without IP Core

# 2-4- Area Analysis

To evaluate the **area**, we refer to the **number of components** used in the circuit, as shown in the figure below. These component counts are organized in **Table 1**.

File Edit View Project Source Process Tools	Winde	ow Layout Help	
। अन्त्र 🖉 🖓 🖏 🖉 🖓 🖄 🖗 🗐 🗑 🖉		B 🗟 💫 🕾 🗄 🖬 🖳 🖋 🖗 🕨 🗶 🗶 👂	
esign ↔ □ 문 ×			
View:		HDL Synthesis Report Macro Statistics # Multipliers	: 12
■ □ xc7vx330t-2ffg1157		8x8-bit multiplier	: 12
		# Adders/Subtractors	. 12
Complex_ALU (Complex_ALU.V)		17-bit adder	: 3
		17-bit subtractor	: 3
		32-bit adder	: 2
		# Registers	: 14
		1-bit register	: 6
		144-bit register	: 2
		2-bit register	: 2
		32-bit register	: 2
	1	48-bit register	: 2
		# Comparators	: 3
		2-bit comparator greater	: 1
		32-bit comparator greater	: 2
	-	# Multiplexers	: 110
		1-bit 2-to-1 multiplexer	: 101
		16-bit 3-to-1 multiplexer	: 6
	i.	32-bit 2-to-1 multiplexer	: 1
No Processes Running		48-bit 2-to-1 multiplexer	: 2
Processes: Comlex ALU	1		

Figure 7: Area Analysis of the Circuit Without IP Core

Components	Numbers of the component				
8x8-bit multiplier	12				
17-bit adder	3				
17-bit subtractor	3				
32-bit adder	2				
1-bit register	6				
144-bit register	2				
2-bit register	2				
32-bit register	2				
48-bit register	3				
2-bit comprator grater	1				
32-bit comprator grater	2				
1-bit 3-to-1 multiplexer	101				
16-bit 3-to-1 multiplexer	6				
32-bit 2-to-1 multiplexer	1				
48-bit 2-to-1 multiplexer	2				

### Table1 : Number of Components in the Circuit Without IP Core

# 3- IP Core-Based Design for Complex ALU

In this design, we replaced the **complex number multiplier module** with an **IP Core**.

# 3-1- Functional Verification of the IP Core-Based Design

For this purpose, we use **Xilinx ISE**. After writing the **testbench**, we apply the following **input values** to the circuit and observe the corresponding **output**.

$$A = \begin{bmatrix} 2+4j & 1+3j & 2+4j \\ 3+j & 2+2j & 6+7j \\ 7+2j & 2+7j & j \end{bmatrix}$$
$$B = \begin{bmatrix} 7+4j & 1+4j & 3+2j \\ 2+j & 1+7j & 2+2j \\ 2+3j & 3+2j & 1 \end{bmatrix}$$

### **3-1-1-** Analysis of the Addition Operation

The result of adding the two matrices above is as follows.

 $\begin{bmatrix} 2+4j & 1+3j & 2+4j \\ 3+j & 2+2j & 6+7j \\ 7+2j & 2+7j & j \end{bmatrix} + \begin{bmatrix} 7+4j & 1+4j & 3+2j \\ 2+j & 1+7j & 2+2j \\ 2+3j & 3+2j & 1 \end{bmatrix} = \begin{bmatrix} 9+8j & 2+7j & 5+6j \\ 5+2j & 3+9j & 8+9j \\ 9+5j & 5+9j & 1+1j \end{bmatrix}$ 

The simulation results are shown below, confirming the **correctness of the addition operation**.

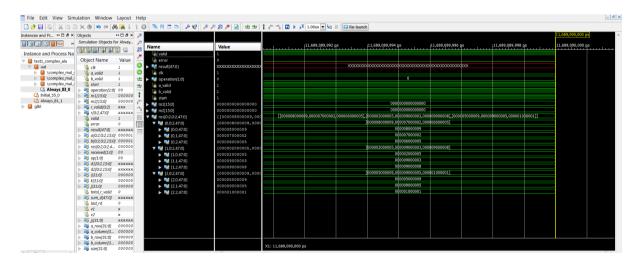


Figure 8: Simulation of the ALU Using IP Core for the Addition Operation

### 3-1-2- Analysis of the Substraction Operation

The result of subtracting the two matrices above is as follows.

$$\begin{bmatrix} 2+4j & 1+3j & 2+4j \\ 3+j & 2+2j & 6+7j \\ 7+2j & 2+7j & j \end{bmatrix} - \begin{bmatrix} 7+4j & 1+4j & 3+2j \\ 2+j & 1+7j & 2+2j \\ 2+3j & 3+2j & 1 \end{bmatrix} = \begin{bmatrix} -5 & -j & -1+2j \\ 1 & 1-5j & 4+5j \\ 5-j & -1+5j & -1+j \end{bmatrix}$$

The simulation results are shown below, confirming the correctness of the subtraction operation.

🔜 File Edit View Sim	ulation Window	Layout H	elp														- 8 ×
🗋 🌶 🖬 😓 🕺 🛍 🛙	≥×® ≈ ≈ 0	4 <b>3</b> 34↓î	0 5 8	🗆 🕾 🥬 🌮 J	8 🖉 🏓 🗟 🛬 🕿	10	1 🖬 🕨 🕫	1.00us 🔻 🔙 🛙	🗔 Re-launch								
Instances and Pr •• 🗆 & ×	Objects	+□∂× ,	÷													46,131,470,000 ps	~
	Simulation Objects f		8		Malua			46,131,469,992 ps		46,131,469,994 p	ns	46,131,469,996	ns	46,131,469,998	ns	46,131,470,000 ps	
Instance and Process Na		15 🔛 🖇	Name		Value						F		F		7	Provide States	
	Object Name	Value	<ul> <li>Us valid</li> <li>Us error</li> </ul>		0												
V sti_complex_alu			Image: Second		*				20000000	000000000000000000000000000000000000000			000				
uut	Us cik	1		ntevol	1				100000000				001				
complex_mul_	b_valid		b b 🐝 ope								1						
complex mul	1 start		r 🖡 👘 ope		1						-						
G Always 83 0	operation[1:0]		b_va		-												
C Initial_55_0	m1(15:0)	000000	la b_va		1												
C Always_81_1	▷ ➡ m2[15:0]	000000	<ul> <li>Stat</li> <li>Stat</li> <li>m1f</li> </ul>		± 000000000000000000000000000000000000					0000	00000000000000000						
⊳ 📒 glbl	▷ ➡ r_valid[0:2]	xxx	1 🖡 📷 m1(		000000000000000000000000000000000000000						000000000000000000000000000000000000000						
	▷ N 10:2,47:0]	XXXXXX		0:2,0:2,47:0]	110000000000000000000000000000000000000	Thomas	000000000000000000000000000000000000000	000000011111111				000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000101111111		
	La valid	0		0:2,0:2,47:0] 10.0:2.47:01	[00000000000000000000000000000000000000		1000000000000000	00000001111111			fffff000000.0000			00000000000000000	00010111111		
	error	0		[0,0:2,47:0]	000000fffffb						000000000000000000000000000000000000000						
	result[47:0] a(0:2,0:2,15:0)	XXXXXX		0,0,47:0]	ffffff000000						11111000000						
	▷ ➡ b(0:2,0:2,15:0)			[0,1,47:0]	000002ffffff						000002ffffff						
	res[0:2,0:2,10:0]			[1,0:2,47:0]	[000002111111 [0000000000001,fff1					000000000000000000000000000000000000000	ffffb000001.0000	050000041					
	received[1:0]				0000000000001, 1111						000000000000000000000000000000000000000	03000004					
	▷ ➡ op/1:01	01		[1,0,47:0]							ffffb000001						
	d1[0:2,15:0]	XXXXXX		[1,1,47:0]	fffffb000001 000005000004						00005000004						
	▷ N d2[0:2,15:0]	XXXXXX		<b>[1,2,47:0]</b>							00003000004	1.66666					
	⊳ 📷 i(31:0)	000000		[2,0:2,47:0]	[ffffff000005,0000						fffff00005	40000					
	⊳ 💕 k[31:0]	000000		[2,0,47:0]	ffffff000005						000005ffffff						
	⊳ 📸 j[31:0]	000000		[2,1,47:0]	000005ffffff						000005000						
	15 total_r_valid	0		[2,2,47:0]	000001ffffff						0000111111						
	▷ ➡ sum_d[47:0]	xxxxxx 0															
	15 vi	x															
	14 v2	x															
	▷ → jj[31:0]	XXXXXX															
	> a_row[31:0]	000000															
	▷ ➡ a_column(3	000000															
	b_row[31:0]	000000															
	b_column(3	000000				¥1• 4	6,131,470,000 p	-									
	b ize[31:0]	000000				A1. 4	0,101,470,000 p.										

Figure 9: Simulation of the ALU With IP Core for the Subtraction Operation

### **3-1-3-** Multiplication Operation Verification

The result of multipication the two matrices above is as follows.

[2 + 4 <i>j</i>	1 + 3 <i>j</i>	2 + 4 <i>j</i> ן	[7 + 4 <i>j</i>	1 + 4 <i>j</i>	3 + 2 <i>j</i> ן	[-11 + 57 <i>j</i>	-36 + 38 <i>j</i>	-4 + 28j
3 + <i>j</i>	2 + 2 <i>j</i>	$6+7j \times$	2+j	1 + 7 <i>j</i>	2 + 2j =	10 + 57 <i>j</i>	-9 + 62 <i>j</i>	13 + 24 <i>j</i>
[7 + 2j]	2 + 7 <i>j</i>	j ]	L2 + 3 <i>j</i>	3 + 2 <i>j</i>	1 ]	$\begin{bmatrix} -11 + 57j \\ 10 + 57j \\ 35 + 60j \end{bmatrix}$	-50 + 54 <i>j</i>	7 + 39 <i>j</i>

The simulation results are shown below, confirming the correctness of the multipication operation.

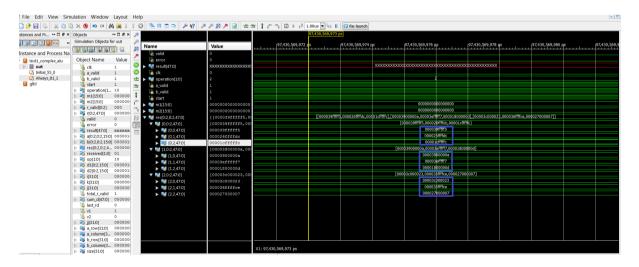


Figure 10: Simulation of the ALU With IP Core for the Multiplication Operation

### 3-2- Analysis of Power, Area, and Delay for the IP Core-Based Design

In this section, we aim to obtain the **power**, **delay**, and **area** results for the **IP core-based design**.

# **3-2-1-** Power Analysis

We follow the same procedure as described in the previous section. The results are shown in **Figure 11**. As seen in Figure 11, the **total power consumption** of the circuit is **0.178 W**. As expected, the **IP-based design consumes less power** than the non-IP design. This confirms our expectation regarding the **efficiency and optimization** of the IP core implementation.

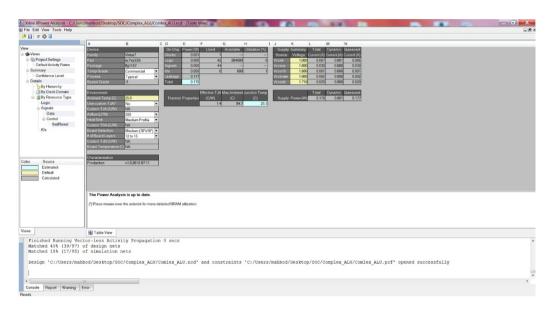


Figure 11: Power Consumption of the Circuit Using IP Core

# 3-2-2- Delay Analysis

Using the same method described in the previous section, we obtain the **delay**. As shown in the figure below, the **critical path delay** is **2.313 ns**. This delay is **lower than that of the non-IP design**, further confirming our expectation regarding the **superior efficiency** of the **IP core-based implementation**.

ISE Project Navigator (P.20131013) - C:\Users\mahbod\Deskto		se - [Comlex_	ALU.syr]		
File Edit View Project Source Process Tools Wind					
🗋 🏓 🖓 😓 🔏 🙆 🖄 🖄 🖄 🖉 🖉 🖉 🖉	P 🗟 🍌 🕾 🗄 🖬 🗣 🎤 🕅	? 🕨 🗶 📌	<b>9</b>		
Design ↔ □ # × 🕻					
👔 View: 💿 🇱 Implementation 💮 🐖 Simulation 🗛	Timing constraint: I				
A Hierarchy	Clock period: 2.31				
G - Complex ALU	Total number of pa				21706 / 1033
R = 2 xc7vx330t-2ffg1157					
a Scrwssol-2rig1157	Delay: Source:	2.313ns	(Levels o	of Logic	= 12)
· · · · · · · · · · · · · · · · · · ·	Destination:	j_1 (FF)			
mc - complex_mul (complex_mul.xco)	Source Clock:	j_1 (FF) clk risi			
complex_array_mul (complex_array_mul.v) d mc - complex mul (complex mul.xco)	Destination Clock:				
mc - complex_mul (complex_mul.xco)     g complex array sub add (complex array sub a	Descination Clock:	CIK FISI	ng		
Complex_array_sub_add (complex_array_sub_a	Data Path: j 1 to	4 1			
	Daca racii. j_1 co	1_1	Gate	Net	
	Cell:in->out	fanout	Delav		Logical Name (Net Name)
	ccii.in your				
	FDR:C->O	9	0.236	0.395	j 1 (j 1)
	LUT1:10->0	i	0.043		Mcompar GND 1 o j[31] LessThan 107 o cy<0> rt (Mcompar GND 1 o j[31] LessThan 107 o cy<0> rt)
	MUXCY:S->O	ĩ	0.238		Mcompar GND 1 o 1311 LessThan 107 o cv<0> (Mcompar GND 1 o 1311 LessThan 107 o cv<0>)
	MUXCY:CI->O	1	0.013		Mcompar GND 1 o j[31] LessThan 107 o cy<1> (Mcompar GND 1 o j[31] LessThan 107 o cy<1>)
	MUXCY:CI->O	1	0.013		Mcompar GND 1 o 1[31] LessThan 107 o cy<2> (Mcompar GND 1 o 1[31] LessThan 107 o cy<2>)
	MUXCY:CI->O	1	0.013		Mcompar GND 1 o 1[31] LessThan 107 o cv<3> (Mcompar GND 1 o 1[31] LessThan 107 o cv<3>)
	MUXCY:CI->O	1	0.013	0.000	Mcompar GND 1 o j[31] LessThan 107 o cy<4> (Mcompar GND 1 o j[31] LessThan 107 o cy<4>)
4 III F	MUXCY:CI->O	1	0.013	0.000	Mcompar GND 1 0 1[31] LessThan 107 0 cy<5> (Mcompar GND 1 0 1[31] LessThan 107 0 cy<5>)
No Processes Running	MUXCY:CI->O	7	0.013	0.384	Mcompar GND 1 o j[31] LessThan 107 o cy<6> (GND 1 o j[31] LessThan 107 o inv)
Real Processes Kurning	LUT2:11->0	1	0.043	0.000	Mcount j lut<0> (Mcount j lut<0>)
Processes: Comlex_ALU	MUXCY:S->O	0	0.238	0.000	Mcount j cy<0> (Mcount j cy<0>)
E Design Summary/Reports	XORCY:CI->O	1	0.262		Mcount_j_xor<1> (Mcount_j1)
🕱 🗉 🞾 Design Utilities	LUT5:14->0	1	0.043	0.000	j_1_glue_set (j_1_glue_set)
-  User Constraints	FDR:D		-0.000		j_1
CAA Synthesize - XST					
View RTL Schematic	Total		2.313ns		ns logic, 1.129ns route)
View Technology Schematic				(51.24	logic, 48.8% route)

Figure 12: Critical Path Delay of the IP Core-Based Circuit

# 3-2-3- Area Analysis

Using the same method described in the previous section, we obtain the **area** of the circuit. The results are organized in **Table 2**. By comparing **Table 1** (non-IP design) and **Table 2** (IP-based design), it is clear that the **IP core-based design occupies less area**, which further confirms the **optimization and efficiency** of using an IP core.

ISE Project Navigator (P.20131013) - C:\Users\mahbod\D	ISE Project Navigator (P.20131013) - C:\Users\mahbod\Desktop\Complex_ALU\Complex_ALU.xise - [Comlex_ALU.syr]									
File Edit View Project Source Process Tools V	Vindo	w Layout Help								
🔯 🔍 🔍 👘 🖉 🔏 🕼 👔 😸 🕼 🔚	<u> </u>									
Design ↔ □ 🗗 ×	(Ĵ									
📑 View: 🧿 🔯 Implementation 🔘 🌌 Simulation	A									
J Hierarchy	-	HDL Synthesis Report								
€ Complex_ALU		Macro Statistics								
□ □ xc7vx330t-2ffg1157		# Adders/Subtractors	: 2							
		32-bit adder	: 2							
mc - complex_mul (complex_mul.xco)		# Registers	: 14							
Complex_array_mul (complex_array_mul.v)		1-bit register	: 6							
_ time complex@new(complex@newco)		144-bit register	: 2							
complex_array_sub_add (complex_array_sub_add)		2-bit register	: 2							
		32-bit register	: 2							
		48-bit register	: 2							
		# Comparators	: 3							
		2-bit comparator greater	: 1							
		32-bit comparator greater	: 2							
	i i	# Multiplexers	: 110							
		1-bit 2-to-1 multiplexer	: 101							
		16-bit 3-to-1 multiplexer	: 6							
		32-bit 2-to-1 multiplexer	: 1							
	1	48-bit 2-to-1 multiplexer	: 2							
4 III +										

Figure 13: Area Analysis of the IP Core-Based Circuit

Components	Numbers of the component
32-bit adder	2
1-bit register	6
144-bit register	2
2-bit register	2
32-bit register	2
48-bit register	3
2-bit comprator grater	1
32-bit comprator grater	2
1-bit 3-to-1 multiplexer	101
16-bit 3-to-1 multiplexer	6
32-bit 2-to-1 multiplexer	1
48-bit 2-to-1 multiplexer	2

### Table2 : Sorted Area Results of the IP Core-Based Circuit

### 4- Comparison of Power, Delay, and Area Results: IP Core vs. Non-IP Core Design

In **Table 2**, we compare the **power**, **delay**, and **area** of the **non-IP** and **IP core-based** designs. All three metrics—**power consumption**, **critical path delay**, and **area**—are improved in the **IP-based design**. The reason for this improvement is that the IP core was designed by expert engineers at **Xilinx**, who have extensive knowledge and experience in creating highly **optimized and efficient circuits** from all aspects.

### Table3 : Comparison of Area, Delay, and Power Parameters Between IP Core-Based and Non-IP Core Designs

Parameters	Without IP	With IP
Power	0.179W	0.178W
Delay	2.537ns	2.313ns
Area	Bigger Area	Smaller Area