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.

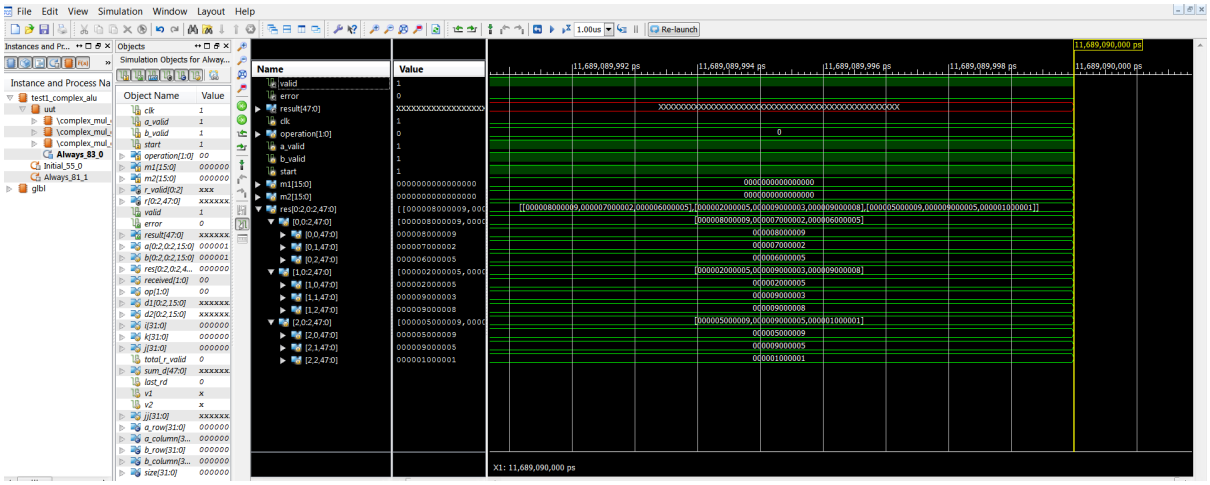

Figure 1: Simulation of the ALU Without IP Core for the Addition Operation

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.

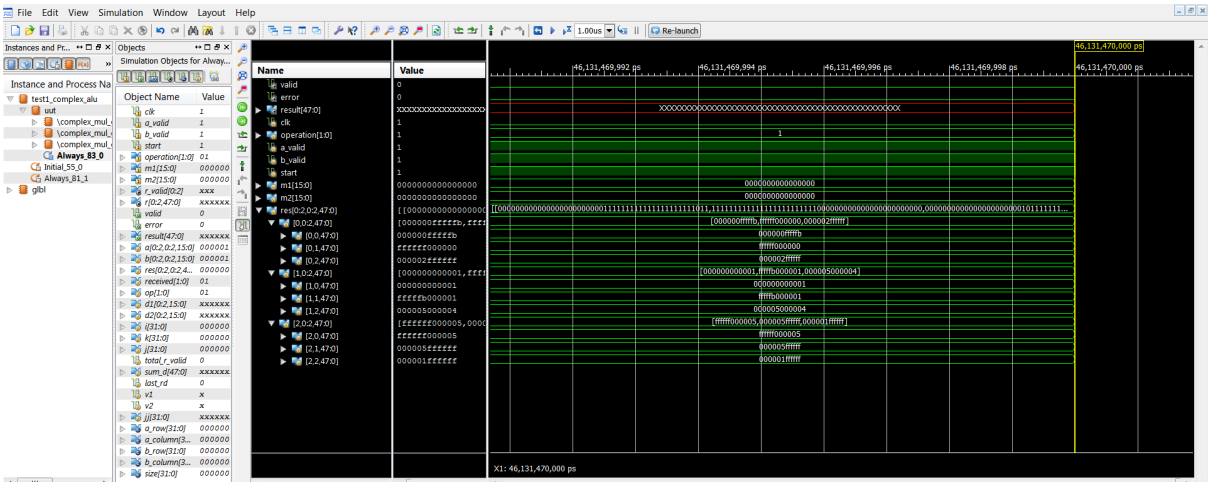

Figure 2: Simulation of the ALU Without IP Core for the Subtraction Operation

2-1-3- Multiplication Operation Verification

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

$$\begin{bmatrix} 2+4j & 1+3j & 2+4j \\ 3+j & 2+2j & 6+7j \\ 7+2j & 2+7j & j \end{bmatrix} \times \begin{bmatrix} 7+4j & 1+4j & 3+2j \\ 2+j & 1+7j & 2+2j \\ 2+3j & 3+2j & 1 \end{bmatrix} = \begin{bmatrix} -11+57j & -36+38j & -4+28j \\ 10+57j & -9+62j & 13+24j \\ 35+60j & -50+54j & 7+39j \end{bmatrix}$$

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

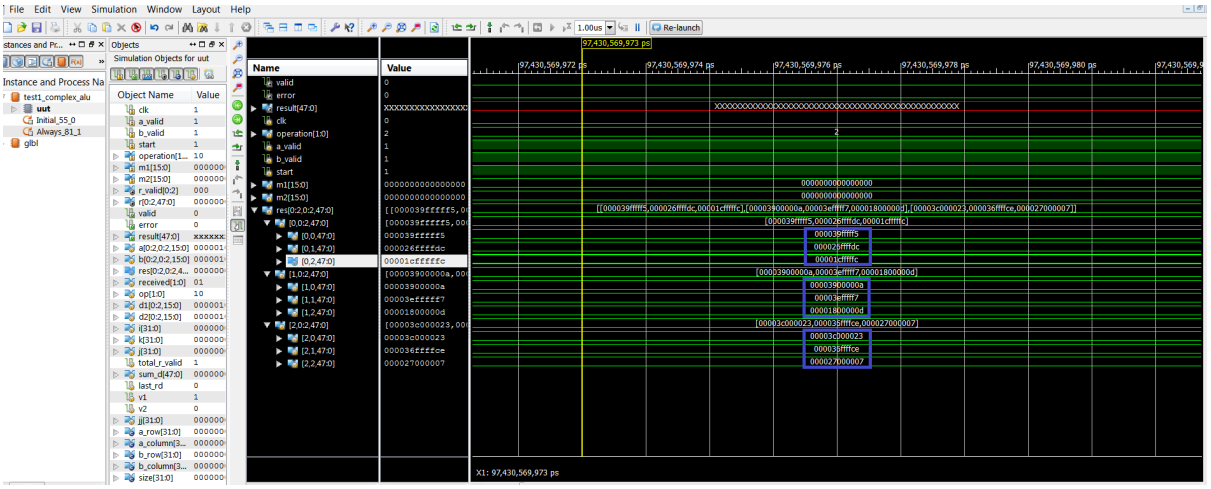

Figure 3: Simulation of the ALU Without IP Core for the Multiplication Operation

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.

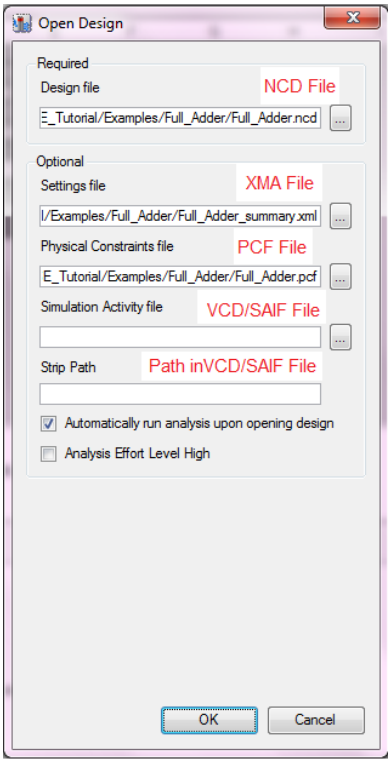

Figure 4: Specifying the Required Inputs in the Open Design Section

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**.

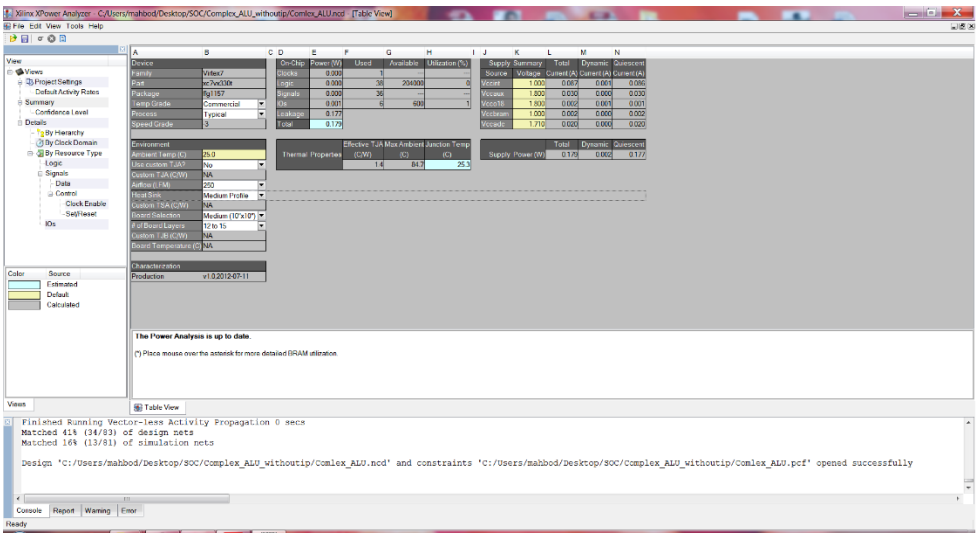

Figure 5: Power Consumption Results of the System Without IP Core

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**.

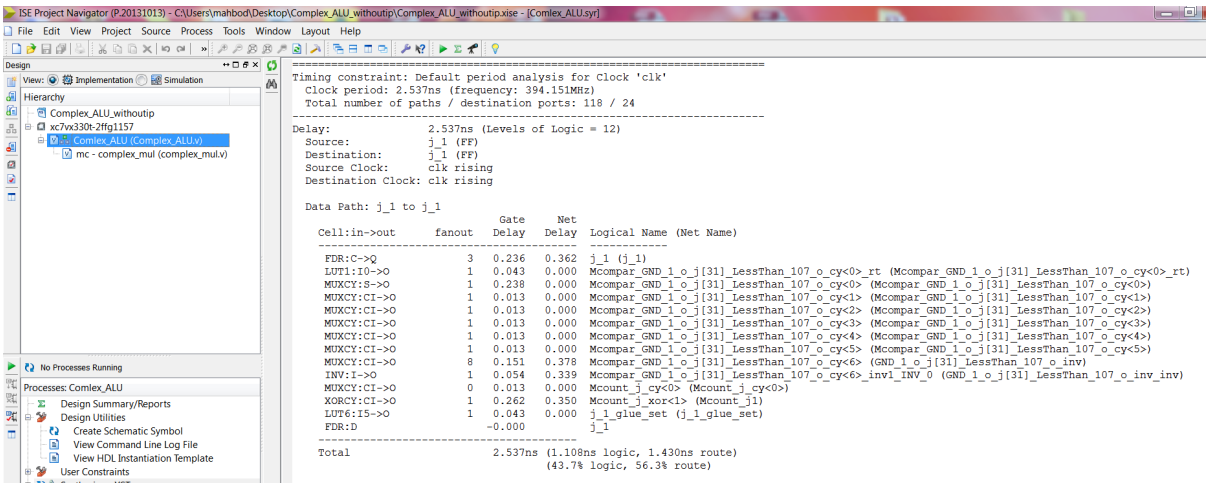

Figure 6: Delay Analysis of the Circuit Without IP Core

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**.

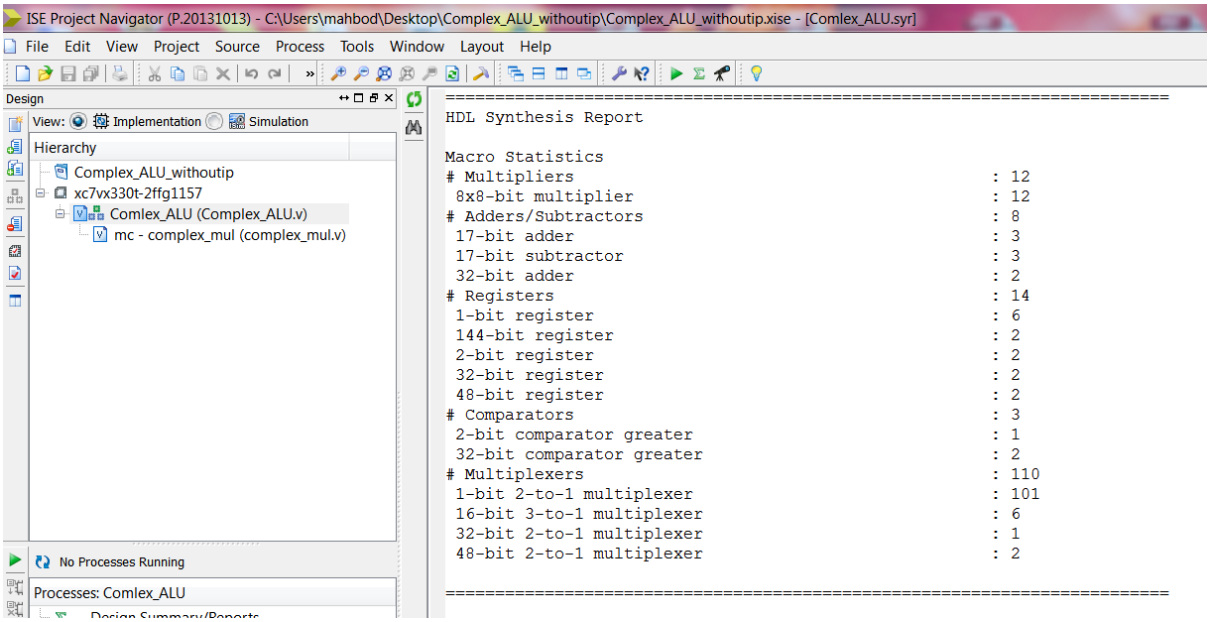

Figure 7: Area Analysis of the Circuit Without IP Core

Table1 : Number of Components in 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**

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**.

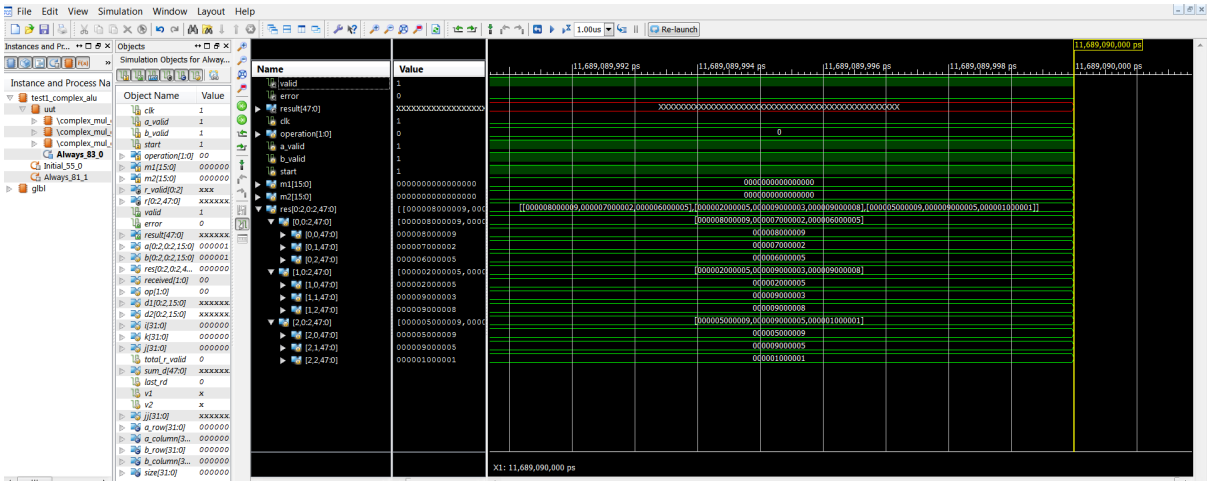

Figure 8: Simulation of the ALU Using IP Core for the Addition Operation

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.

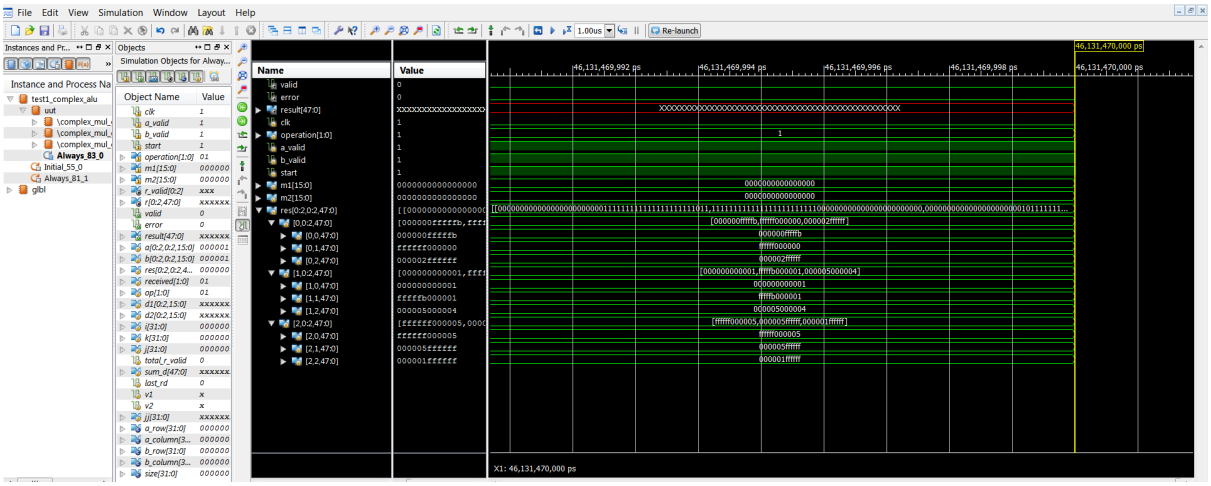

Figure 9: Simulation of the ALU With IP Core for the Subtraction Operation

3-1-3- Multiplication Operation Verification

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

$$\begin{bmatrix} 2+4j & 1+3j & 2+4j \\ 3+j & 2+2j & 6+7j \\ 7+2j & 2+7j & j \end{bmatrix} \times \begin{bmatrix} 7+4j & 1+4j & 3+2j \\ 2+j & 1+7j & 2+2j \\ 2+3j & 3+2j & 1 \end{bmatrix} = \begin{bmatrix} -11+57j & -36+38j & -4+28j \\ 10+57j & -9+62j & 13+24j \\ 35+60j & -50+54j & 7+39j \end{bmatrix}$$

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

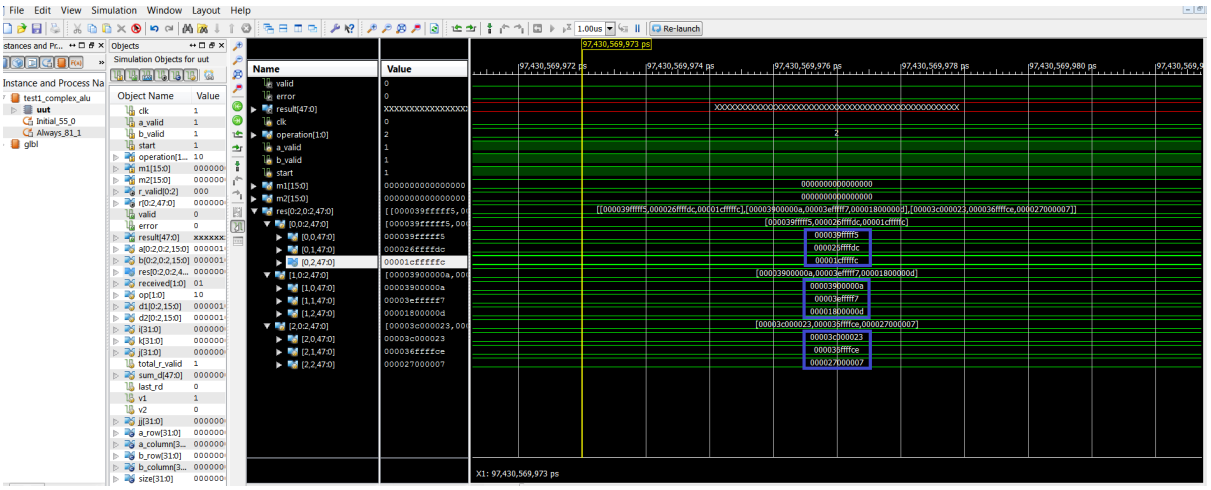

Figure 10: Simulation of the ALU With IP Core for the Multiplication Operation

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.

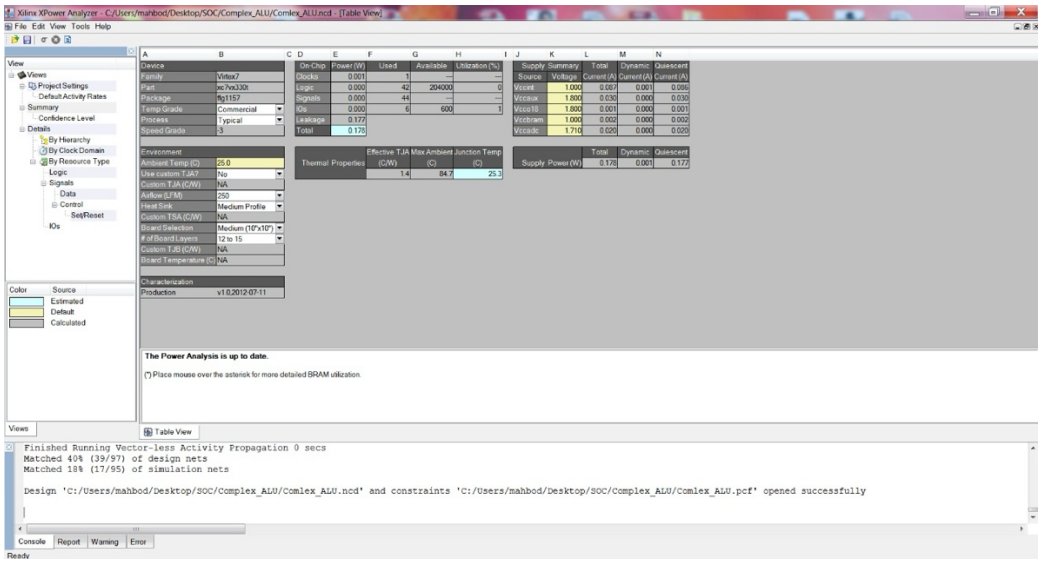

Figure 11: Power Consumption of the Circuit Using IP Core

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**.

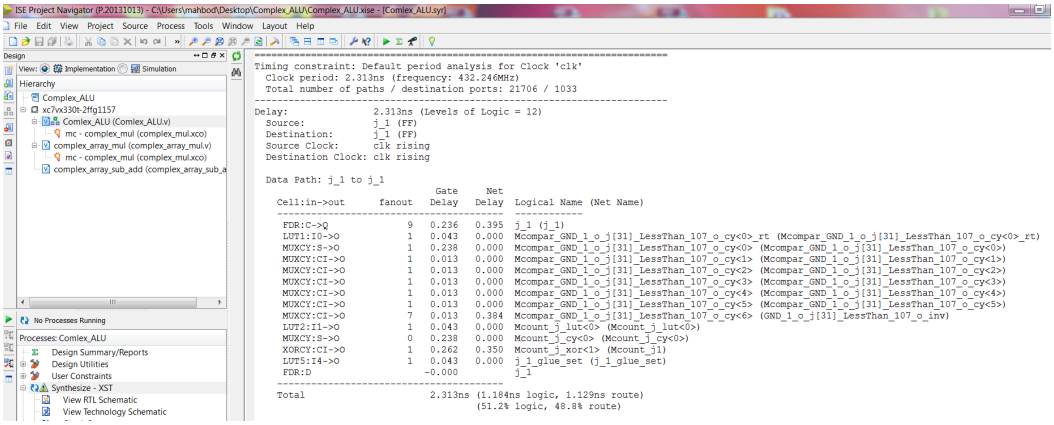

Figure 12: Critical Path Delay of the IP Core-Based Circuit

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.

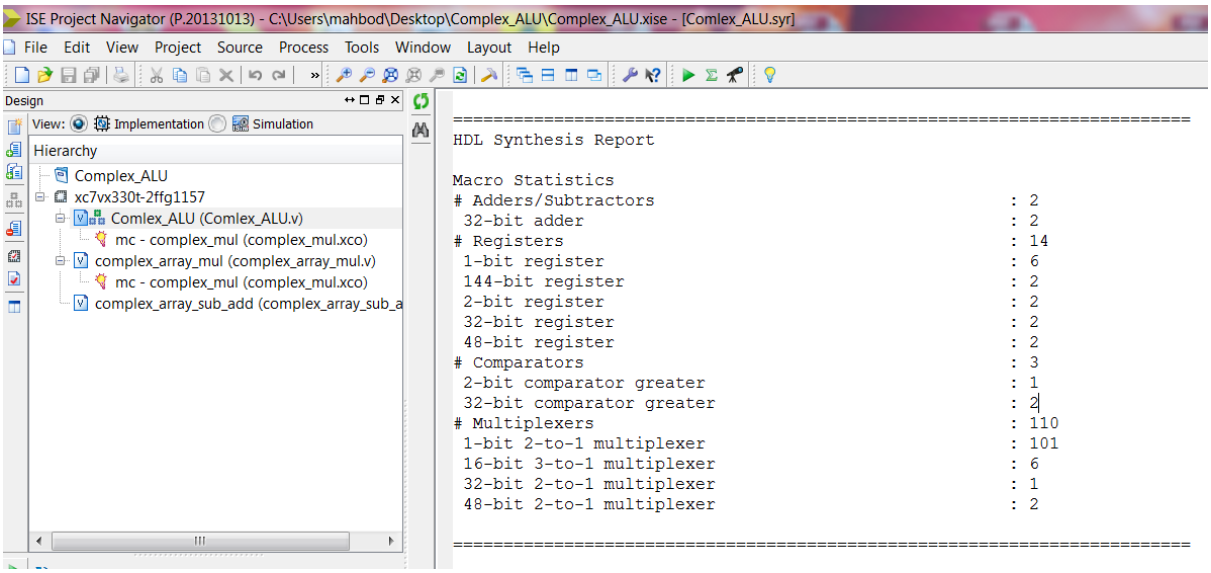

Figure 13: Area Analysis of the IP Core-Based Circuit

Table2 : Sorted Area Results 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**

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*