Design And Simulation Of AXI4 Lite Protocol Using Verilog

Kazi Nikhat Parvin, K Harshitha, A Akshitha, L Manasa

[1]Associate professor, Department of ECE, Bhoj Reddy Engineering College for Women, India

[2,]B.Tech Student, Department of ECE, Bhoj Reddy Engineering College for Women, India

ABSTRACT

The Advanced eXtensible Interface 4 Lite (AXI4 Lite) protocol serves as a simplified yet efficient method for interconnecting digital systems, facilitating register based communication between master and slave devices. This mini project endeavors to design and implement AXI4 Lite protocol using Verilog. The project's primary objective is to develop a robust and functional AXI4 Lite master and slave interface capable of handling read and write transactions, adhering to the protocol's specifications.

The implementation process involves breaking down the AXI4 Lite protocol into its fundamental components, including address decoding, data multiplexing, and control signal management. A hierarchical modular design approach is adopted, facilitating the systematic development and integration of individual modules.Verilog is employed as the hardware description language, providing a platform for concise and efficient representation of digital hardware components.Thorough testing and verification methodologies are employed to validate the functionality and correctness of the implemented AXI4 Lite interface. Simulation using industry standard Verilog simulators allows for comprehensive analysis of the interface's behavior under various operating conditions and transaction scenarios.

Additionally, the design is synthesized, enabling real world validation and performance evaluation.The mini project aims not only to provide a working implementation of the AXI4 Lite protocol but also to serve as a learning experience for digital design enthusiasts, offering insights into the intricacies of protocol based communication.

1-Introduction

The AXI4 Lite (Advanced eXtensible Interface) is a subset of the AXI4 protocol, designed by ARM for simpler, lightweight communications between IP cores in a SoC (System on Chip). The protocol is used primarily for low throughput communication and provides a simple register like interface for peripherals and processors. In this project, we focus on designing and simulating the AXI4 Lite protocol using Verilog HDL (Hardware Description Language), which allows the creation of a functional model of the communication interface.

The Advanced eXtensible Interface (AXI) protocol is a part of the ARM AMBA (Advanced Microcontroller Bus Architecture) family, widely used in modern System on Chip (SoC) designs to facilitate communication between different components like processors, memory, and peripherals. The AXI4 Lite is a lightweight subset of the AXI4 protocol, optimized for low complexity and low throughput applications. Unlike the full AXI4, which supports burst transactions and complex interleaving, AXI4 Lite provides a simpler interface that enables point to point, register based communication with only single data transfers at a time.

2- Literature Survey

In the foundational document by ARM, the AXI4 Specification outlines the protocol's features and describes how it provides high performance, low latency communication between system components. The AXI4 family, including AXI4 Full, AXI4 Lite, and AXI4 Stream, is designed to be highly flexible, scalable, and reusable across multiple IP cores in a SoC. AXI4 Lite, as a lightweight variant, supports non burst single transactions, making it particularly suited for memory mapped peripherals and control registers. In this specification, key characteristics such as addressing, data transfer protocols, and timing requirements are described in detail, forming the core technical framework for AXI4 Lite designs.

The implementation of the AXI4 Lite protocol in FPGA based systems has been a popular area of study. For example, in their work, Jones and Smith (2017) explore the integration of AXI4 Lite in Xilinx FPGAs, using the Vivado Design Suite. Their study focuses on how the protocol can be synthesized into programmable logic for use in custom peripheral designs. Through their simulation and verification efforts, they demonstrate that AXI4 Lite can be successfully implemented with minimal resource utilization, making it a viable choice for FPGA based projects. The paper also emphasizes the importance of simulation tools such as ModelSim for validating AXI4 Lite behavior and ensuring protocol compliance, as incorrect timing or signal transitions can lead to system failures.

Various studies have explored optimization techniques for AXI4 Lite implementations. A notable study by Wu and Zhao (2019) investigates methods to reduce the latency and resource consumption of the protocol while maintaining its simplicity. They propose the use of pipelining in the data and control path of the AXI4 Lite interface to enhance throughput and reduce latency without compromising the lightweight nature of the protocol. Additionally, their research delves into power saving strategies, demonstrating that AXI4 Lite can be further optimized for low power applications by reducing the clock frequency and utilizing power gating techniques.

Another important aspect of AXI4 Lite discussed in the literature is its role in IP core reusability. A study by Patel and Singh (2018) examines how the AMBA AXI4 standard, particularly AXI4 Lite, has led to the standardization of IP core interfaces, allowing IP cores to be reused across different platforms and designs. This is crucial in modern SoC design, where time to market is a key factor, and the ability to reuse verified IP blocks significantly speeds up development. Their research also emphasizes that by adhering to the AXI4 Lite specification, designers can ensure interoperability between IP blocks from different vendors, reducing the risk of integration issues.

The importance of verification in AXI4 Lite designs is discussed in detail by Lee and Park (2021), who focus on the challenges of verifying AXI4 Lite interfaces in large SoC environments. Their research highlights that although AXI4 Lite is simpler than its full counterpart, thorough verification is still required to ensure correct functionality.

They propose a verification framework that includes comprehensive testbenches for modeling the behavior of both the master and slave interfaces, ensuring that all possible edge cases, such as incorrect data transfers or signal glitches, are tested before implementation. The study also emphasizes the role of simulation tools, such as ModelSim and Vivado, in identifying design errors early in the development process.

3-AXI4 LITE PROTOCOL

In this chapter we will discuss about the process of designing and simulation of AXI4 Lite protocol using Verilog.

The convention essentially creates the standards about how various modules on a chip speak with one another, essentially it requires a handshake like technique prior to all transmissions. Having a convention, for example, this permits a genuine "framework" as opposed to an "assortment" of modules to be set up as the convention interfaces and gives a viable medium to move information between the current segments on the chip.

To go more inside and out, the interface works by setting up correspondence among master and slave gadgets. Among these two gadgets (or more if utilizing an AXI Core IP) exists five separate channels: Read Address, Write Address, Read Data, Write Data, and Write Response. Each channel has its own special signals just as comparable signs existing among every one of the five. The substantial and prepared signs exist for each channel as they take into consideration the handshake cycle to happen for each channel.

For communicating any sign (address/information/reaction/and so forth) the significant channel source gives a functioning legitimate sign and a similar channel's objective should give a functioning prepared sign. After the two signs are dynamic, transmission might happen from that channel. As expressed over, the transmission of control signals/address and information are done in discrete stages, and subsequently a location should consistently be moved between gadgets before the handshake interaction can happen for the comparing data move. On account of composing data, the response channel is utilized toward the fulfillment of the data move.

There it is. The convention is that simple! Obviously there are extra alternatives that the convention gives that up the intricacy fairly, like burst move, QoS, Protections, and others. These alternative

are essentially additional signs existing on the various channels that take into account extra usefulness, for general use nonetheless, the above portrayal conveys the idea on how this interface by and large functions.

By working with the master and slave gadgets, the AXI convention works across five tends to that incorporate peruse and compose address, peruse and compose information, and compose reaction. Since each channel has its own one of a kind sign, it can send the handshake reaction continuously so it tends to be gotten and placed into request. That way, the channel that has need will be reacted to first, etc. The source should give a legitimate sign and one that gets an appropriate reaction from the beneficiary.

By having the transmission acted in independent stages, it permits the exchange of data to be acted in a precise way. This implies that a handshake or arrangement is arrived from the outset, at that point the data is moved from the source to the beneficiary. Also, that is the way the AXI convention attempts to move data between various

sources without obstruction.

Working

The AXI4 Lite protocol project focuses on designing and simulating a simplified version of the AXI4 protocol using Verilog HDL. The primary objective is to implement the AXI4 Lite interface, which enables communication between a master (usually a processor or controller) and a slave (such as memory, peripherals, or registers) for low throughput, low latency data transfers. The system's working is divided into several stages, involving design, simulation, and verification of the protocol. Here's an overview of how the project works:

Master Slave Communication:

In the AXI4 Lite system, communication occurs between a master and a slave over a shared bus. The master initiates the transactions, either reading from or writing to the slave device. The communication takes place through distinct signal groups, including:

- **Address Signals**: Carry the address of the target location in the slave.
- **Data Signals**: Carry the data to be written or the data read from the slave.
- **Control Signals**: Indicate the validity of the address and data, control handshakes, and transaction completion.

The protocol operates on a simple request response model, with the master sending the address and control information, followed by the data (in the case of a write), and the slave responding with data (for reads) or an acknowledgment (for writes).

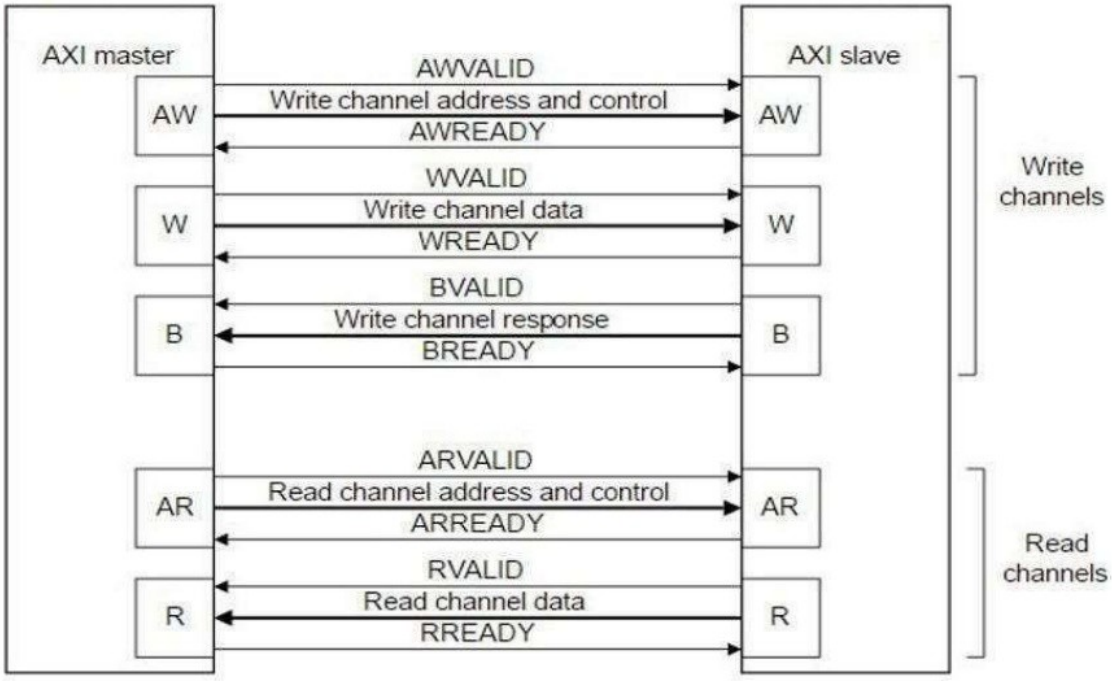

Fig 3.2.1 Block Diagram

The block diagram depicts the communication protocol between an AXI master and an AXI

slave using the AXI4 interface. This diagram illustrates how data flows during both write and read transactions, highlighting the various control and data signals involved in the AXI protocol. It is essential to understand how these signals operate to ensure efficient communication between the components in a system on chip (SoC) architecture.

4- SOFTWARE REQUIREMENTS

In this chapter we will discuss software requirements for Design and Simulation of AXI4 Lite Protocol using Verilog.

VIVADO

Programming assignments in this course will almost exclusively be performed in VIVADO, is a comprehensive software suite for synthesizing and analyzing hardware description language (HDL) designs, succeeding the older Xilinx ISE. It offers enhanced features for system on chip (SoC) development and high level synthesis, streamlining the design process. With a complete redesign of the design flow, VIVADO incorporates modern methodologies to improve efficiency and performance. It supports the integration of complex designs and advanced optimization techniques, making it essential for FPGA and SoC development. Typical areas of use include:

- Digital Circuit Design
- System on Chip (SoC) Development
- High Level Synthesis
- Embedded Systems
- Prototyping and Testing

The first time you start VIVADO, the desktop appears with the default layout, as shown in Figure 1. The VIVADO desktop consists of the following parts:

- Design Workspace: The main area for creating and managing design projects
- Sources Pane: Displays the hierarchy of design files, allowing you to view and edit HDL files.
- IP Integrator: A graphical interface for integrating and configuring IP cores into your design.
- Output Window: Shows messages, warnings and errors during synthesis, simulation, and implementation.
- Diagram View: Provides a visual representation of your design when using the IP integrator.

The VIVADO editor (Figure 2) allows you to create and modify HDL files with features such as syntax highlighting, code folding, and error checking. You can access it directly from the Sources Pane by double clicking on a file. The editor supports both VHDL and Verilog languages, enabling efficient design entry and modifications.

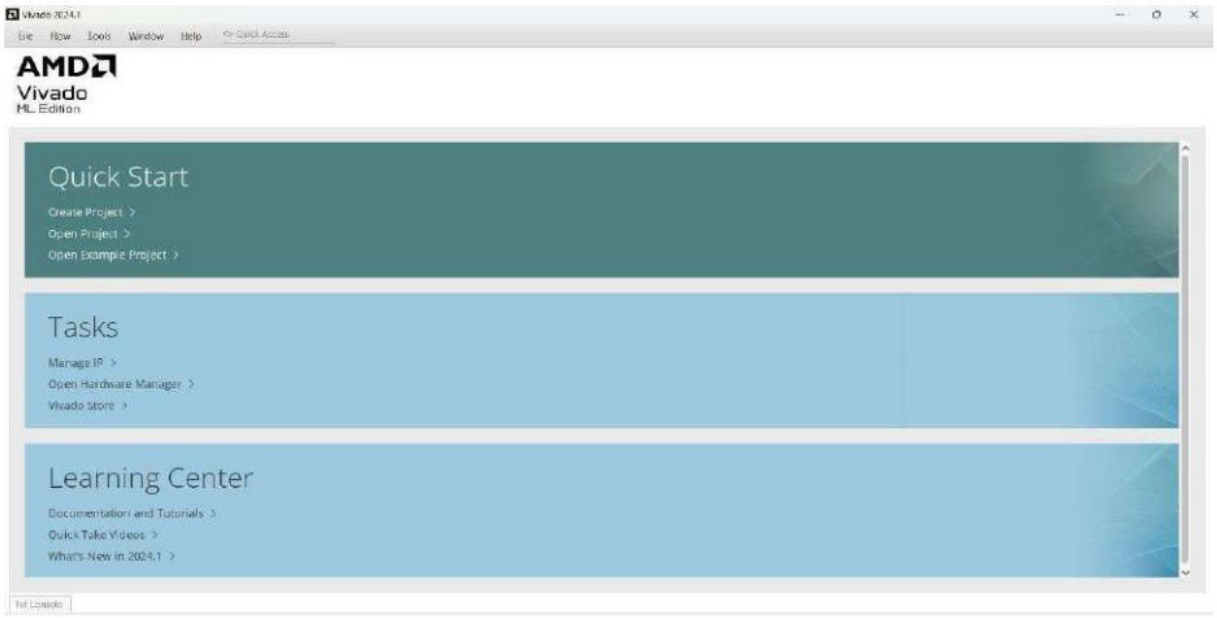

Fig 4.2.1 : VIVADO Desktop(default layout

5- RESULTS AND DISCUSSION

5.1 Comparative Analysis

AXI4 Lite is a simplified version of the Advanced eXtensible Interface (AXI) protocol,primarily designed for low bandwidth control and status register accesses. It is a part of the ARM AMBA (Advanced Microcontroller Bus Architecture) protocol family. Here's a comparison between AXI4 Lite and previous protocols, including its own predecessors within the AMBA family, such as AHB (Advanced High performance Bus) and APB (Advanced Peripheral Bus):

AXI4 Lite vs. AHB

1. Bus Architecture:

AXI4 Lite: Supports a point to point connection with separate read and write channels, allowing for simultaneous read and write operations. It has a more flexible interconnect structure.

AHB: Uses a single bus architecture for both read and write transactions, which can lead to contention when multiple masters try to access the bus.

2. Complexity:

AXI4 Lite: Simpler in terms of implementation and use, as it has fewer signals and does not support burst transactions. It is designed for low throughput applications.

AHB: More complex, supporting burst transfers and higher bandwidth applications.

1. Address Write Channel

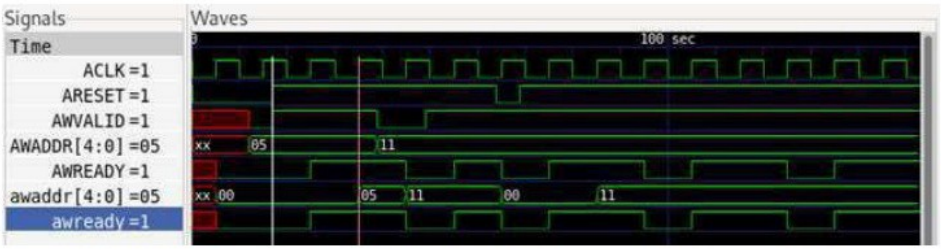

Figure 5.2.1: Address write channel with required signals

Figure 4.1 shows that write address channel simulation on axi slave module. In that first, we give ACLK and ARESET signals as per specification. Note In this simulation upper case related to slave input or output port and lower case monitor signal which is the latch inside slave. First, we provide AWADDR and AWVALID signals as input and depend on input AWREADY asserted high. After one clock cycle awaddr accepts that input address and latch into the slave as shown in fig.4.1 When the reset was low then awaddr was also goes starting address in this case zero.

2. Data Write Channel

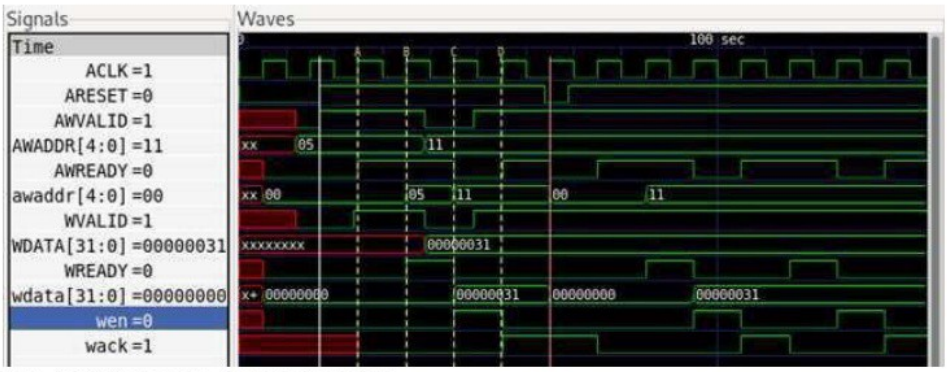

Figure 5.2.2: Data write channel simulation

Above fig. 5.2, show the simulation of data writes channel at a slave. In that first perform write address and put some address on slave latch. In the first case, the address should be 0x05 which shows with a white marker. At marker B 0x05 latch into a slave. In axi don't have share bus terminology that's why along with the write address we put write data. For that first slave input, WVALID becomes asserted one and WDATA put some value here 0x31. A marker, A WVALID becomes asserting high. After the very next clock cycle, WREADY also becomes asserting high at marker B. As soon as WREADY becomes high next clock cycle wdata latch data at marker C. When data latch into slave wen signal also asserted high. This signal uses to lock data and provide acknowledgment for memory. After the next clock cycle data store in memory of the array and wack is

asserted high. Wen and wack both are internal signals for checking the working of write data on the memory array.

3. Write Response Channel

Figure 5.2.3: Write Response channel simulation

Above fig.4.3 show that the write response channel at the slave in that BREADY signal always asserted high from the master side. When slave latches some data in wdata then write transaction was successful complied and BVALID also asserted high at the white marker. When both BVALID and BREADY asserted high it means handshaking complied and all required data and address latch into a slave.

3. Master and slave connect back to back

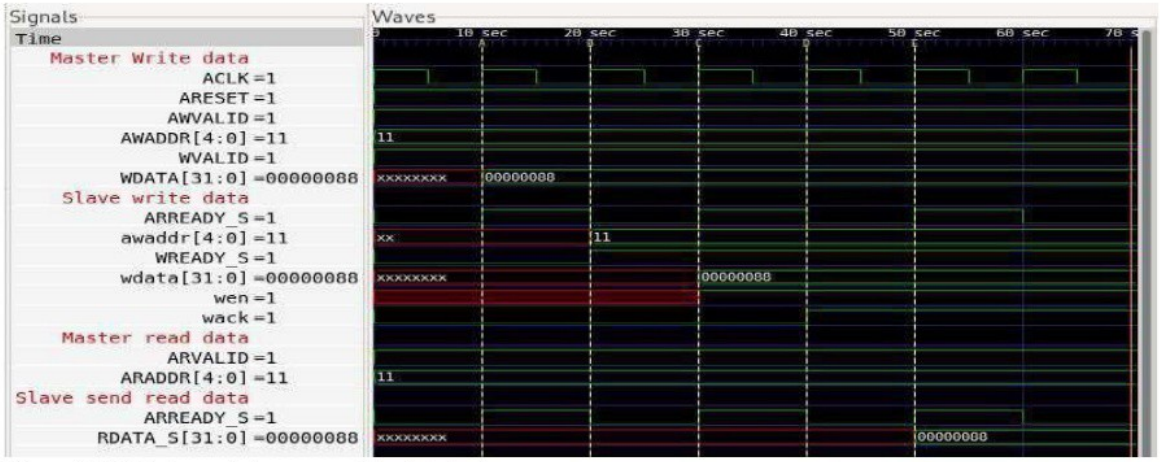

Figure 5.2.4: Master and slave connect back to back

Above simulation done for checking master and slave back to back connection. In the above fig. 4.4 Position A show master insert AWADDR = 0x11, and WDATA = 0x88 with respective valid signals. Marker B signifies slave get address form AWADDR and latch into awaddr. Marker C signifies WDATA also latch into wdata. Before both information latch, respective ready signal asserted high so handshaking was ensured before feathering any process. At master side also asserted ARADDR and ARVALID with 0x11 and 0x1 respective. It shows that I want to write and read at the same location or address. For a replay of a read, channel slave put related data by given address on RDATA_S which is indicating by marker E.

5.2 Synthesis

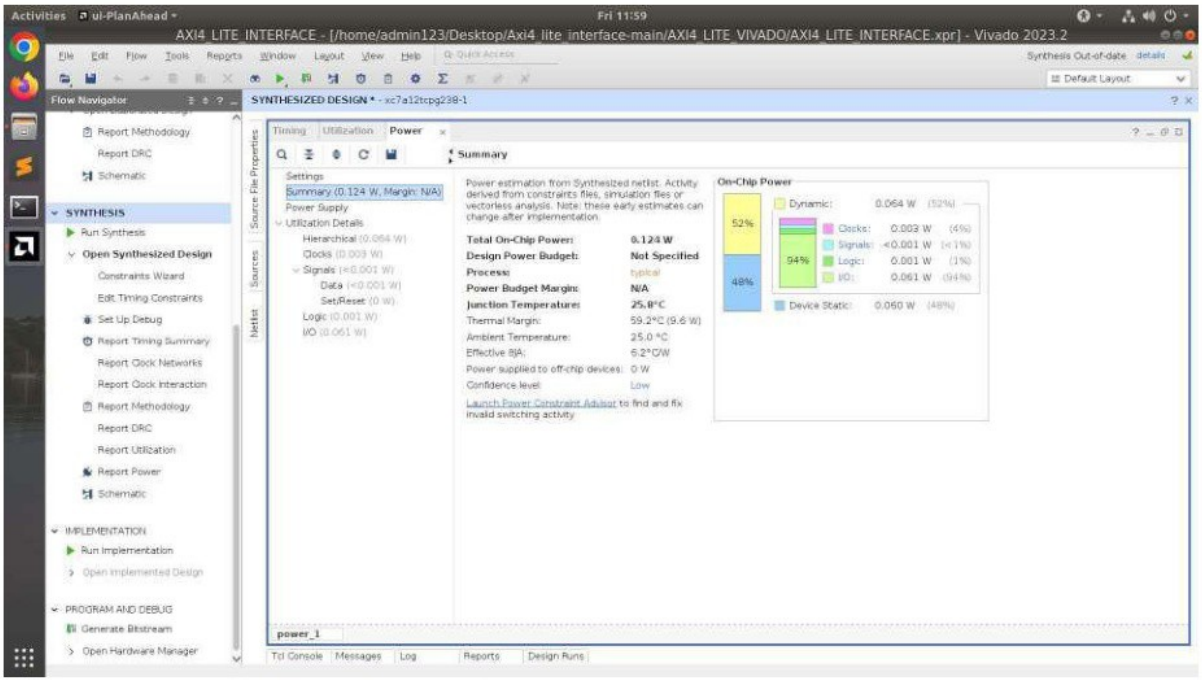

Fig 5.3.1Synthesized Power Utilization

Utilization Details

- Hierarchical**:** 0.064 W
- Clocks**:** 0.003 W
- Signals, Data, Set/Reset, Logic**:** Each consumes very little power.
- I/O**:** 0.061 W, indicating most of the dynamic power is used by the I/O elements.

Interpretation

- The largest contributor to dynamic power is the I/O, consuming 94% of dynamic power, which suggests the design is I/O intensive.
- The static power contributes almost half of the total power, indicating that the device's inherent leakage is significant.

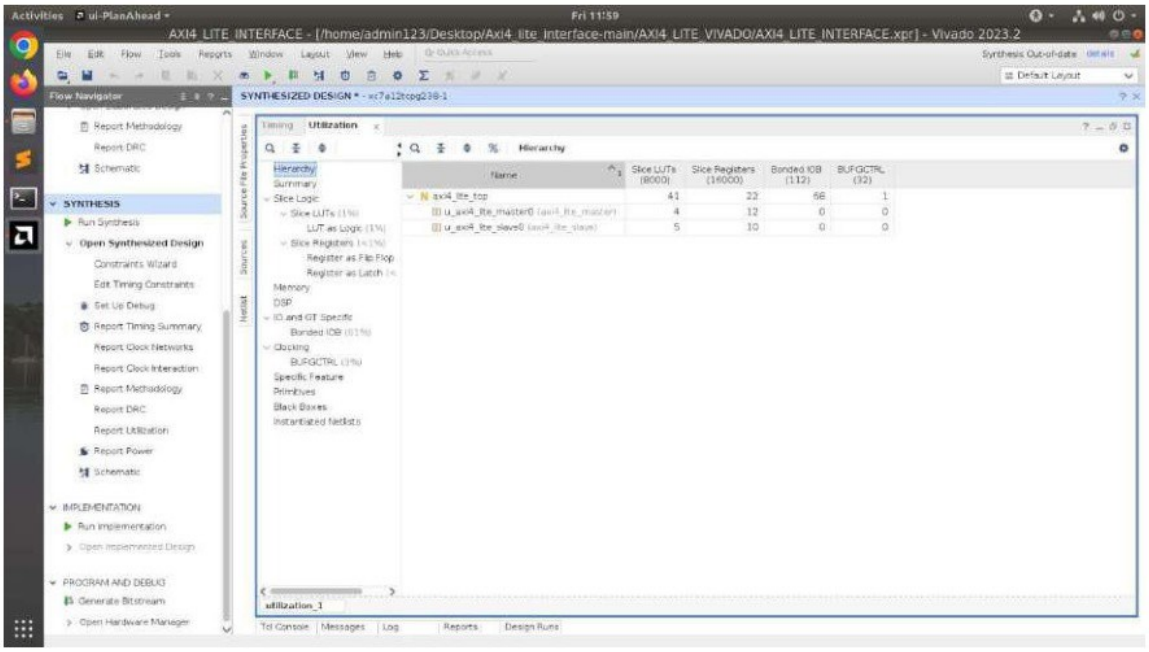

Fig5.3.2 Hierarchy of Resource Utilization

- Slice LUTs (8000)

These represent the Look Up Tables (LUTs) used in the design. LUTs are the basic logic elements in an FPGA.

- The design uses a total of 41 out of 8000 available LUTs:
  - axi4_lite_top: 41 LUTs
  - u_axi4_lite_master0: 4 LUTs
  - u_axi4_lite_slave0: 5 LUTs

In this report, we started out with a conversion about AXI Protocol and AMBA protocol significance, along with its versions variant. In the present digital era, we need faster memories and **CONCLUSION**

In this project, we successfully designed and simulated the AXI4 Lite protocol using Verilog

enormous speed for communicating with different devices. For computing, those requirements need heterogeneous peripherals on SoCs along with communication protocol. To full fill these requirements with AMBA AXI.

The work focused on design an AXI Lite 32 bit master and slave using Verilog. And verified design by applying test cases. For Low memory peripheral requirement full fill by using memory array at slave side. Here, we conclude with the synthesis of AXI LITE master and slave Memory mapped peripheral along with simulation waveform. At the end, we performing effective data transactions in back to back master slave connection.

HDL, focusing on efficient communication between a master and slave device in a system on chip (SoC) or FPGA environment. The project explored the essential aspects of low latency and

low bandwidth data transfers, implementing reliable handshake mechanisms to ensure proper synchronization between transactions. The use of Verilog allowed for precise control over signal management, timing, and resource allocation. Through the structured design, simulation, and verification process, the AXI4 Lite protocol was optimized for minimal power consumption and efficient use of resources.

The successful completion of the project demonstrates a thorough understanding of the AXI4 Lite protocol's functionality and highlights the importance of custom protocol design for low power and embedded systems. This project can serve as a foundation for further exploration of on chip communication protocols, emphasizing the adaptability and scalability of AXI4 Lite for a wide range of applications.

REFERENCES

[1]"AXI4 Lite Protocol Implementation and Custom IP Design" Xiao Qiu, Zhao Wei, IEEE 2023.

[2]Understanding the AXI Protocol: A Quick Introduction(2023) AnySilicon.

[3]"Design and Verification of Advanced Microcontroller Bus Architecture (AMBA) Protocols" Rajthilak S., Mohankumar N, IEEE 2022.

[4]Design and Verification of AMBA AXI3 Protocol for High Speed Communication (2022) (IJRASET).2022 Smart Technologies, Communication and Robotics (STCR) published. IEEE 13 January 2023.

[5]"Implementation of Advanced High Performance Bus to Advanced Peripheral Bus Bridge"Chinta Sai Manasa, Navya Mohan, J. P. Anita, IEEE 2021.

[6]Jakub Szefer, "AXI4 Lite Interface Development", EENG 428 / ENAS 968 Cloud FPGA. [Accessed: November 14, 2019].

[7]Verification of AMBA AXI on chip communication protocol,Nikhil Gaikwad, Vijay N Patil 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 1 5, 2018.