e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:04/April-2024 Impact Factor- 7.868 www.irjmets.com

IMPROVING THE VERIFICATION EFFICIENCY WITH VIRTUAL SEQUENCES USING UNIVERSAL VERIFICATION METHODOLOGY

Rupadevi N*[1], Balaji S*[2], Gomathi P*[3], Jeeva V*[4], Sangeetha R*[5]

*[1]Assitant Professor, Electronics And Communication Engineering, Adhi College Of Engineering And Technology, Kanchipuram, Tamil Nadu, India.

*[2,3,4,&5]Student, Electronics And Communication Engineering, Adhi College Of Engineering And Technology, Kanchipuram, Tamil Nadu, India.

ABSTRACT

The verification of complex digital designs is further compounded when dealing with specific components like FIFO (First-In-First-Out) memory modules. These modules are integral to data processing and communication systems. Within the Universal Verification Methodology (UVM) framework, virtual sequences play a vital role in addressing these challenges. By explores the master-slave concept using virtual sequences for FIFO module verification. virtual sequences for FIFO module verification, we enable the creation of intricate test scenarios specifically to emulate the read and write operations of FIFO memory modules automating the verification process to enhance productivity, accuracy, and reduce time-to-market. Thus, while UVM provides a robust framework for system-level verification, the application of virtual sequences to FIFO modules is crucial for ensuring the reliability and functionality of digital designs.

Keywords – FIFO, Virtual Sequence, Master-Slave, UVM.

I. INTRODUCTION

This project focuses on the development and implementation of a comprehensive verification environment utilizing the Universal Verification Methodology (UVM) framework. The primary objective is to verify complex digital designs, with a specific emphasis on the operation of FIFO (First-In-First-Out) memory modules. To address the challenges associated with intricate data storage and retrieval mechanisms inherent in FIFO memory modules, we employ a master-slave concept. The master component is responsible for write operations, while the slave component handles read operations. Leveraging virtual sequences within the UVM framework, to automate the verification process, enhancing productivity, accuracy, and reducing time-to-market. The master component generates sophisticated test scenarios tailored to emulate write operations, the slave component focuses on creating diverse read scenarios. Together, these components ensure a holistic verification approach. Key components of the verification environment include the construction of systematic test scenarios. These scenarios are meticulously designed to cover both read and write operations of FIFO memory modules, ensuring thorough verification coverage and reliability

II. LITERATURE SURVEY

[1] In this paper FIFO consists of dual port RAM. This dual port RAM allows simultaneous access of read and write port synchronous FIFO architecture. The read and write process of FIFO is performed on a same clock. For every positive edge of the clock the data is written in to the FIFO, when the write enable is high and the FIFO is not full. When the read enable is high the data is read out for every positive edge of the clock and FIFO is not empty and all the output signals is set to zero when reset is given. In this work Synchronous FIFO is designed using Verilog and verified using UVM and simulation is carried out in Questa Sim tool.
[2] This paper proposes the design of a priority-based master slave communication system using SPI Protocol that enables the system to operate using interrupts. The design mainly emphasizes on priority-based communication where the slaves will generate an interrupt over a newly defined interrupt pin when some data transfer needs to happen. When the master receives an interrupt from the slave it establishes communication with one slave at a time based on the priority and the priority to each slave is assigned by the arbiter or priority control block. In SPI protocol, usually the master always initiates the communication. The master generates clock signal which is of a frequency the slave devices support

III. METHODOLOGY

2.1 Existing System

The current system employs the Universal Verification Methodology (UVM) framework to verify the FIFO memory module. Verification is achieved through the utilization of regular sequences, predefined sequences of events or transactions. However, while regular sequences can support constrained random verification, they possess limitations regarding flexibility and dynamic generation capabilities. Specifically, they offer constrained modeling abilities for higher-level scenarios based on specifications and constraints. Additionally, engineers must manually define these regular sequences, which can be time-consuming and less adaptable to design changes. Furthermore, a single sequencer oversees both write and read operations, which may result in synchronization challenges and decreased efficiency within the verification process.

2.2 Proposed System

We introduce a comprehensive approach to FIFO memory module verification within the UVM framework, integrating virtual sequences and a master-slave concept. Virtual sequences serve as dynamic generators of test scenarios, offering unparalleled flexibility in modeling scenarios based on constraints and specifications. By supporting constrained random verification, virtual sequences ensure more thorough coverage of scenarios, addressing potential verification gaps present in the existing system. Furthermore, the adoption of a master-slave architecture enhances the efficiency and synchronization of the verification environment. With a master sequencer managing write operations and a slave sequencer handling read operations, the system ensures synchronized and efficient operation, reducing the risk of synchronization issues and improving overall verification efficiency.

By automating scenario generation, both virtual sequences and the master-slave architecture minimize manual effort, allowing verification engineers to focus on higher-level tasks and ensuring quicker adaptation to design changes or updates. Overall, the proposed system provides enhanced flexibility, efficiency, and coverage compared to the existing system, ensuring the integrity and reliability of the FIFO memory module verification process.

2.3 Features

- **Enhanced Flexibility:** The proposed system offers enhanced flexibility in scenario generation based on constraints and specifications. High data transmission rates of up to 10Gbps can be achieved.
- **Automation:** Automated scenario generation reduces manual effort and increases verification efficiency. IOT has low implementation and maintenance costs.
- **Scalability:** The proposed system is scalable and adaptable to changes in the design, improving overall flexibility.
- **Comprehensive Verification:** Virtual sequences enable comprehensive scenario coverage, reducing the risk of verification gaps
- **Improved Coverage:** The dynamic nature of virtual sequences ensures coverage of a wide range of scenarios, enhancing overall verification quality.

2.3 Architecture

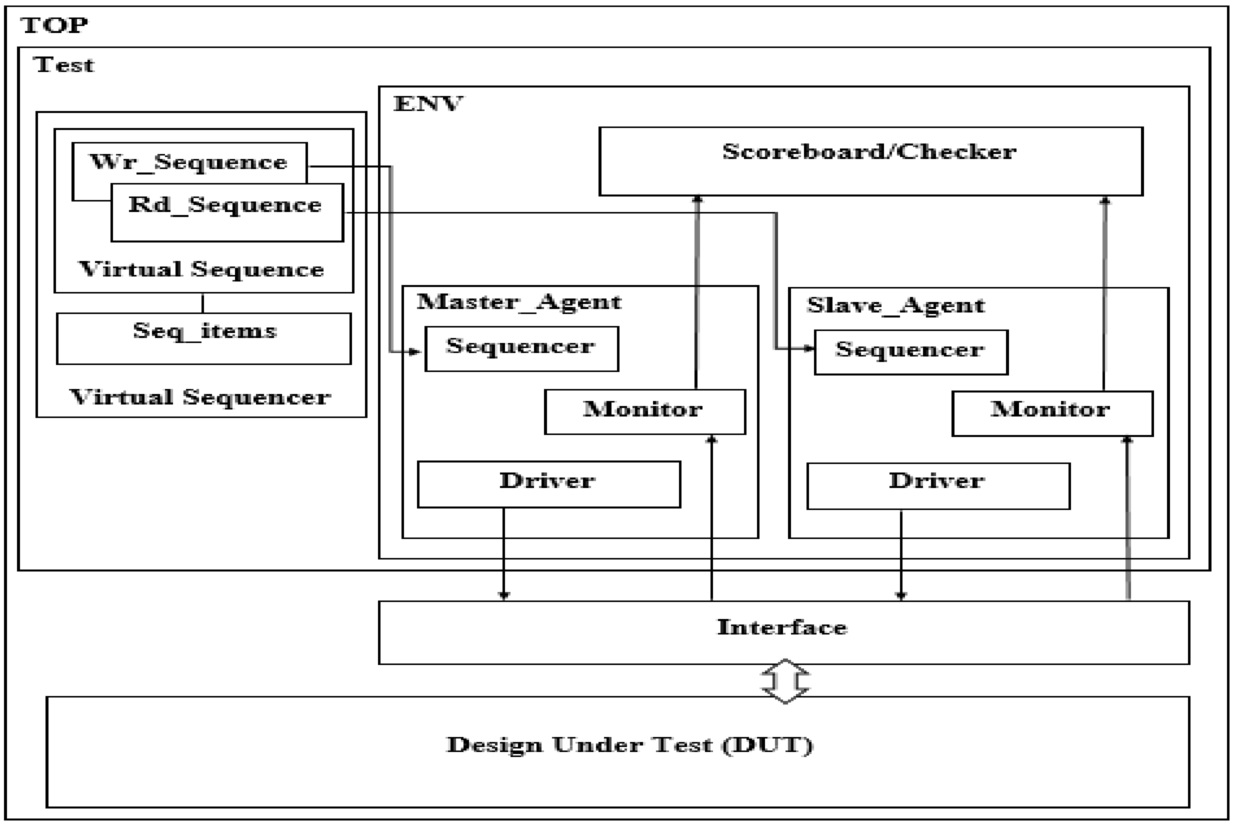


Figure 1: Architecture

IV. MODELING AND ANALYSIS

3.1 Synchronous FIFO Design

A synchronous FIFO (First-In-First-Out) is a fundamental building block in digital design, offering efficient data storage and retrieval mechanisms synchronized to a common clock signal. Unlike asynchronous FIFOs, which operate without a clock and can introduce timing uncertainties, synchronous FIFOs synchronize read and write operations to the rising edge of a clock signal, ensuring precise data transfer and coordination. When data is written into the FIFO, it is captured by the input register and transferred to the storage elements. when data is read from the FIFO, it is retrieved from the storage elements and presented at the output register. The read and write pointers track the current locations within the FIFO, allowing for efficient data movement and management. The "write" operation involves inserting data into the FIFO, typically at the tail end, ensuring that newer data is added after existing entries. Upon a write operation, the module advances its tail pointer, indicating the availability of new data for subsequent read operations. Conversely, the "read" operation retrieves data from the FIFO, typically from the head end, adhering to the FIFO principle of retrieving the oldest data first. This operation advances the FIFO's head pointer, indicating the consumption of data and making room for new entries.

The isFull functionality determines whether the FIFO has reached its maximum capacity, indicating that further write operations may result in overflow. The isEmpty functionality checks if the FIFO is devoid of any data, signaling that read operations may lead to underflow if attempted.

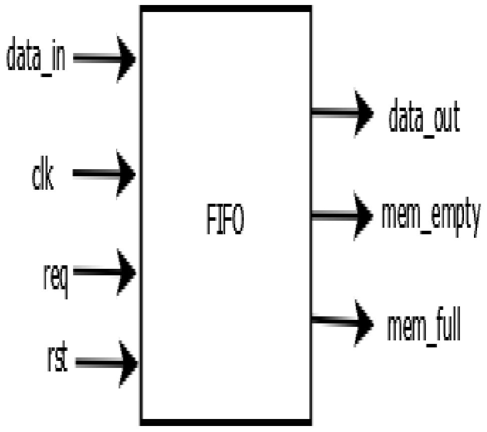


Figure 2: Block Diagram of FIFO

3.2 Master-Slave Concept

The Master-Slave concept is a fundamental architecture widely used in various applications, including digital design verification. In this architecture, the Master and Slave components work collaboratively to manage and execute tasks efficiently.

Master: The Master manages the overall flow of operations, ensuring synchronization and coordination among multiple Slave components. It also handles data transfer, storage, and retrieval, directing the Slave components to perform read and write operations as required. Furthermore, the Master monitors the execution of tasks and manages

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:04/April-2024 Impact Factor- 7.868 www.irjmets.com

error handling, ensuring that any issues or discrepancies are addressed promptly. It takes on the responsibility of initiating tasks or operations, directing the Slave components to execute specific sequences or transactions.

Slave: The Slave component acts as the executor or subordinate in the Master-Slave architecture. It executes the tasks or operations initiated by the Master, performing the required sequences or transactions. The Slave component processes data as directed by the Master, performing read and write operations, data manipulation, or other specified tasks. It reports the results of task execution back to the Master, providing status updates, data verification, or error reports as required.

3.3 UVM Component Class

All the infrastructure components in a UVM environment derive from the uvm component class and build a hierarchy which includes: sequencers, drivers, monitors, coverage collectors, scoreboards, environments and tests. In the following subsections the components that conform a UVM.

Test: It serve as pivotal components within the verification environment, encapsulating specific scenarios or test cases aimed at verifying distinct aspects of the design. Each test delineates the stimulus to be applied to the Device Under Test (DUT), along with any pertinent constraints governing the test scenario.

Environment: The environment serves as the overarching framework that encapsulates the entire verification infrastructure. Comprising a diverse array of components such as testbenches, agents, monitors, and other verification IP, the environment forms the backbone of the verification ecosystem. At the heart of the environment lies its ability to manage the overall functionality of the testbench, including stimulus generation, data analysis, and result reporting. By providing a cohesive and structured framework, the environment fosters collaboration among various verification components and facilitates the rigorous validation of the design under test (DUT).

Agent: Agents play a crucial role in interfacing with the Design Under Test (DUT) and facilitating communication within the verification environment. An agent represents a functional block responsible for abstracting the communication protocol and providing a standardized interface for interacting with the DUT. It typically encompasses components such as drivers, monitors, sequencers, and virtual sequences, which collectively ensure efficient data exchange and synchronization between the testbench and the DUT.

The Master Agent serves as the primary controller and coordinator for write operations. It takes charge of initiating and managing write transactions to the DUT, orchestrating the generation and transmission of data based on predefined or dynamically generated sequences.

The Slave Agent specializes in executing read operations within the verification environment. It interacts with the DUT to perform read operations, retrieve data, and validate the correctness of the DUT's responses.

Scoreboard: Scoreboard as a vital component responsible for ensuring the correctness and integrity of the Design Under Test (DUT). Its primary function revolves around comparing the expected behavior of the DUT with its actual performance, typically based on transaction-level data gathered from monitors embedded within the testbench. The scoreboard verifies that the DUT produces the correct outputs in response to the provided inputs, thus validating its functionality against specified requirements and protocol standards. In essence, scoreboards act as vigilant watchdogs, meticulously detecting any errors, discrepancies, or protocol violations that may arise during the verification process.

Driver: Drivers play a pivotal role in facilitating communication between the testbench and the Design Under Test (DUT). Acting as the interface between sequencers and the DUT's interface, drivers are tasked with translating abstract transactions generated by sequencers into tangible signals or transactions compatible with the DUT's interface protocol. This process involves handling timing, sequencing, and synchronization aspects to ensure the accurate delivery of stimulus to the DUT. In the context of master and slave drivers, each assumes distinct responsibilities tailored to their respective roles within the verification environment.

The Master Driver, for instance, takes charge of generating write transactions based on predefined test scenarios encapsulated in virtual sequences. It orchestrates the delivery of these transactions to the DUT's interface, meticulously synchronizing their timing with the DUT's clock and other interfaces to ensure seamless operation.

The Slave Driver focuses on extracting read data from the DUT's interface and preparing it for further verification. Alongside managing timing and synchronization aspects, the Slave Driver verifies the correctness and data integrity of read operations by comparing the retrieved data with the expected values.

Monitor: It serve as vigilant observers tasked with scrutinizing the interface between the Design Under Test (DUT) and the testbench. Their primary function involves capturing input and output transactions traversing the interface, extracting pertinent information, and relaying it to other verification components for analysis. It play a pivotal role in various verification tasks, including protocol checking, coverage collection, and debugging, thereby enhancing the overall verification process.

The Master Monitor, for instance, diligently observes the write transactions dispatched by the Master Driver, ensuring their correct initiation and synchronization with the DUT. Additionally, it monitors interface signals and handshakes between the Master Driver and the DUT to guarantee seamless communication and synchronization throughout the write operations. Concurrently, it provides real-time status updates on write operations, facilitating prompt error detection and resolution.

The Slave Monitor focuses on monitoring read transactions facilitated by the Slave Driver, verifying their proper initiation and synchronization with the DUT. It meticulously validates the read data captured by the Slave Driver, ensuring its conformity with expected values and adherence to the prescribed protocol. Moreover, the Slave Monitor vigilantly monitors interface signals and handshakes between the Slave Driver and the DUT, ensuring robust communication and synchronization during read operations.

Sequences: Sequences play a crucial role in defining high-level scenarios or sequences of transactions to be executed within the testbench environment. These sequences serve as comprehensive encapsulations, integrating stimulus generation, constraint application, and transaction sequencing logic into a cohesive unit. By abstracting away implementation details and focusing on the essential interactions between components, sequences offer a convenient and efficient means to model complex behaviors at a higher level of abstraction.

Sequencers: Sequencers play a vital role in the coordination and management of transaction flow between the testbench and the Device Under Test (DUT) within the Universal Verification Methodology (UVM) framework. Responsible for orchestrating the execution of sequences, sequencers receive requests from sequences, generate corresponding transactions, and deliver them to the appropriate interface or agent for execution. This process ensures proper synchronization and sequencing of transactions, facilitating efficient communication between various components of the verification environment.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:04/April-2024 Impact Factor- 7.868 www.irjmets.com

Master Sequencers serve as the primary controllers for initiating and sequencing write transactions sent by the Master Driver to the DUT. They meticulously manage the flow and ordering of write sequences, whether predefined or dynamically generated, to ensure adherence to the specified protocol. Additionally, Master Sequencers synchronize with the Master Monitor to validate transaction initiation, perform data validation, and detect any errors during write operations.

Slave Sequencers oversee the initiation and sequencing of read transactions executed by the Slave Driver to retrieve data from the DUT. Similar to their Master counterparts, Slave Sequencers manage the flow and ordering of read sequences to uphold protocol compliance. They also synchronize with the Slave Monitor to validate transaction initiation, conduct data validation, and detect errors during read operations.

Virtual sequences: virtual sequences to orchestrate the execution of sequences across multiple interfaces or agents, enabling comprehensive verification of system-level behaviour. By decoupling from specific implementation details, virtual sequences promote reusability and adaptability, making them well-suited for handling diverse verification tasks across different projects or iterations of the design.

Virtual Sequencer: Acting as intermediaries between virtual sequences and sequencers tied to specific interfaces or agents, virtual sequencers facilitate seamless communication and coordination across different parts of the testbench. By decoupling virtual sequences from low-level implementation details, virtual sequencers promote modularization and reusability of testbench components, enabling the construction of flexible and adaptable verification environments. This hierarchical structure fosters a more efficient and scalable verification process, allowing for the integration of diverse verification components and the handling of complex system-level scenarios.

3.4 UVM Phases

In Universal Verification Methodology (UVM), the simulation process is organized into distinct phases to ensure a synchronized mechanism throughout the verification flow.

Build Time Phase: The Build Time Phase operates at zero simulation time and follows a top-down execution style. During this phase, the testbench components are constructed, configured, and initialized. It involves setting up the test environment, including creating and configuring agents, drivers, monitors, and sequencers.

Run Time Phase: This phase is characterized by the execution of test scenarios, where stimulus is applied to the DUT, and responses are monitored and analyzed. Test cases run sequentially or concurrently, depending on the testbench architecture and the requirements of the verification plan. The Run Time Phase spans from the beginning to the end of the simulation, encompassing the execution of all designated test scenarios.

Clean Up Phase: The Clean-Up Phase occurs after the completion of the Run Time Phase. During this phase, the results of the simulation, including data from the scoreboard and coverage analysis, are collected and processed. Any necessary post-processing tasks, such as generating reports or performing additional analysis, are carried out in this phase.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:04/April-2024 Impact Factor- 7.868 www.irjmets.com

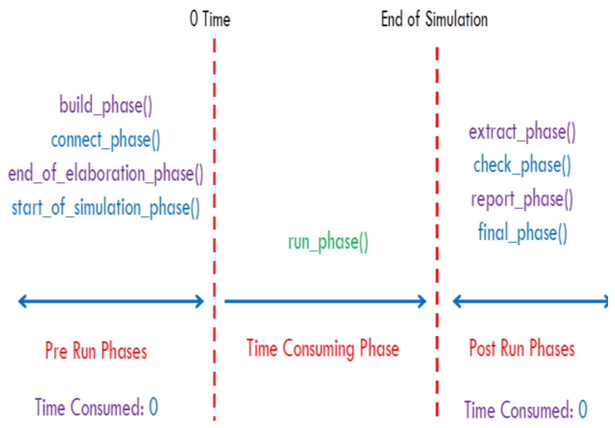


Figure 3: Execution of UVM Phases

3.5 TLM Ports and Exports

Ports: Ports can be categorized based on their primary function and directionality. For instance, an analysis port primarily serves as an output from an agent or a driver to transmit data transactions to other components in the testbench. On the other hand, blocking put and get ports are typically bidirectional, allowing a component to both send and receive data transactions.

Exports: An analysis export, for instance, is commonly used as an input for a scoreboard component, receiving data transactions from an analysis port. Exports are crucial for establishing the communication link between different components in the testbench, ensuring that data flows seamlessly and efficiently.

3.6 Virtual Sequence on Virtual Sequencer

The Virtual Sequence operates on a Virtual Sequencer of type 'uvm_sequencer'. This Virtual Sequencer serves as a bridge between the Virtual Sequence and the Sequencers belonging to the Agents. Its role entails overseeing the execution of the Virtual Sequence and assigning the execution of Sub-Sequences to the relevant Agent's Sequencers. This setup facilitates the organization and management of test scenarios, enhancing the flexibility and modularity of the verification environment.

The Hybrid Approach combines both regular and virtual sequences, along with their corresponding sequencers, to leverage the distinct benefits of each method. Regular sequences provide precise control over deterministic tests, allowing meticulous management of test scenarios. Conversely, virtual sequences autonomously generate random scenarios, enhancing test coverage and uncovering subtle bugs that may evade deterministic testing. By integrating these approaches, the verification environment gains versatility, adeptly accommodating a wide array of test scenarios and ensuring comprehensive verification coverage.

3.7 Verification Methodology

Verification methodology refers to the systematic approach and set of techniques used to verify the correctness, functionality, and performance of digital designs. It encompasses various processes, tools, and best practices aimed at ensuring that the design meets its specifications and requirements

Assertion-Based Verification (ABV): It employs assertions, which are statements or properties describing expected behavior or conditions within a design. These assertions are integrated directly into the RTL (Register Transfer Level) code or written in separate languages like System Verilog Assertions (SVA) or Property Specification Language (PSL). During simulation, assertions are continually evaluated, and if any assertion fails, indicating a violation of the specified condition, an error message is generated. ABV is instrumental in identifying design bugs, protocol violations, and corner-case scenarios by systematically verifying desired properties throughout simulation.

Constrained Random Verification (CRV): It is a methodology where random stimuli for the Device Under Test (DUT) are generated within specified constraints set by the verification engineer. These constraints limit the randomness of the stimulus, ensuring that it aligns with realistic scenarios and design specifications. By generating random stimuli, CRV explores a broad spectrum of possible input scenarios, encompassing both typical and corner-case behaviors. Coverage metrics are employed to gauge the effectiveness of CRV, ensuring that the verification environment thoroughly exercises all pertinent aspects of the design.

V. RESULTS AND DISCUSSION

The assertion-based verification phase conducted on the FIFO memory module yielded critical insights into its functionality and correctness. Through meticulous testing, the integrity of data was rigorously examined, with assertions validating whether the read data from the FIFO matched the expected values. waveform data generated during the simulation provided visual insights into the signals and transactions occurring within the FIFO. This facilitated debugging efforts and further elucidated the module's operation. To conduct the simulation, the project utilized the Synopsis VCS 2023.03 simulator, which was executed within the EDA Playground environment. This combination of advanced verification techniques, detailed reports, and waveform analysis played a pivotal role in ensuring the reliability and functionality of the FIFO memory module within the project's scope

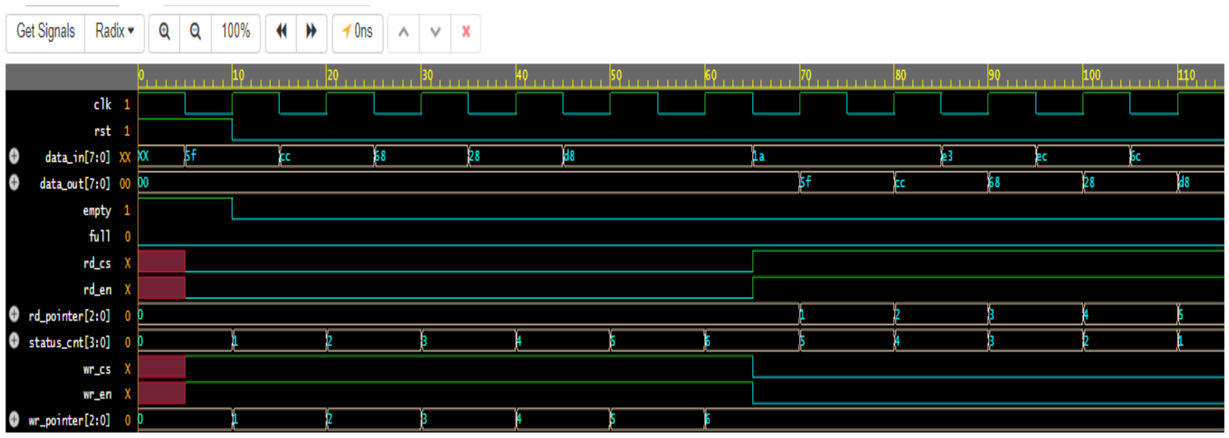
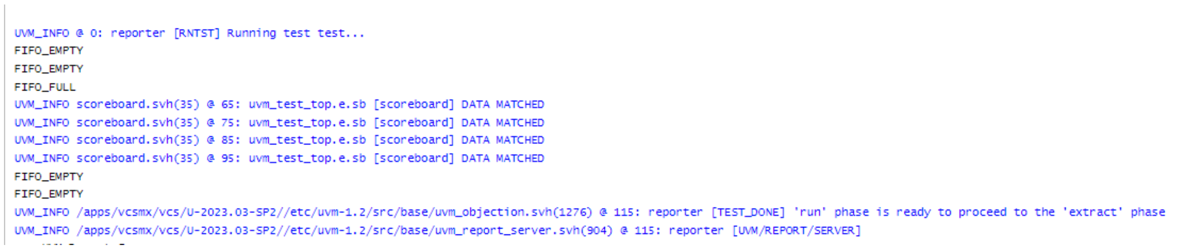


Figure 4: Output Waveform



```
UVM_INFO @ 0: reporter [RNTST] Running test test...
FIFO_EMPTY
FIFO_EMPTY
FIFO_FULL
UVM_INFO scoreboard.svh(35) @ 65: uvm_test_top.e.sb [scoreboard] DATA MATCHED
UVM_INFO scoreboard.svh(35) @ 75: uvm_test_top.e.sb [scoreboard] DATA MATCHED
UVM_INFO scoreboard.svh(35) @ 85: uvm_test_top.e.sb [scoreboard] DATA MATCHED
UVM_INFO scoreboard.svh(35) @ 95: uvm_test_top.e.sb [scoreboard] DATA MATCHED
FIFO_EMPTY
FIFO_EMPTY
UVM_INFO /apps/vcsmx/vcs/U-2023.03-SP2//etc/uvm-1.2/src/base/uvm_objection.svh(1276) @ 115: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
UVM_INFO /apps/vcsmx/vcs/U-2023.03-SP2//etc/uvm-1.2/src/base/uvm_report_server.svh(904) @ 115: reporter [UVM/REPORT/SERVER]
```

Figure 5: Data verification output

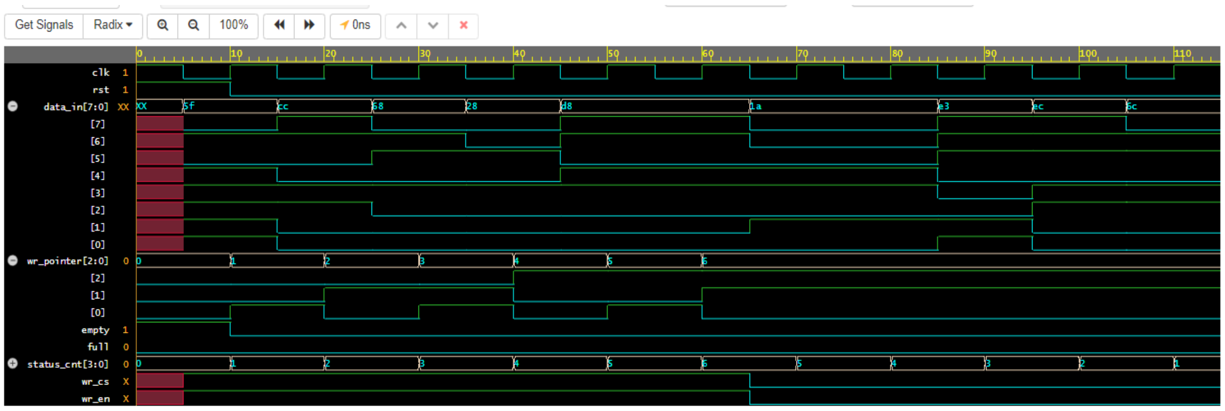


Figure 6: FIFO Write Operation

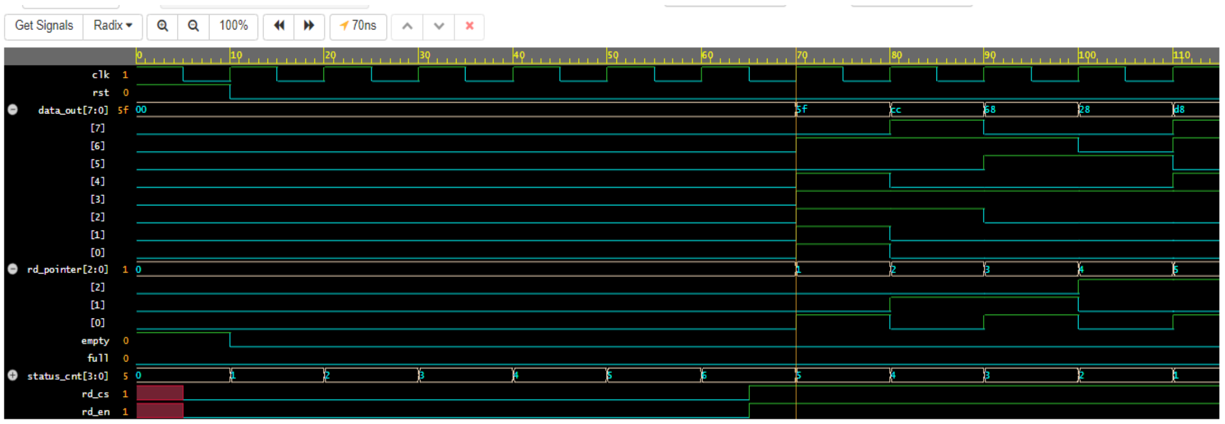


Figure 7: FIFO Read Operation

VI. CONCLUSION

The project successfully implemented and verified a FIFO memory module using advanced verification methodologies within the Universal Verification Methodology (UVM) framework, integrating a master-slave concept to partition responsibilities between agents. By leveraging techniques such as virtual sequences and assertion-based verification, combined with the master-slave architecture, the project showcased robust functionality and correctness of the FIFO module in a multi-agent environment. This approach not only enhanced the scalability and efficiency of the verification process but also facilitated parallel and coordinated verification, allowing for more complex and realistic test scenarios. The adoption of modern verification techniques and the master-slave concept underscored the project's commitment to ensuring the integrity and reliability of digital designs, paving the way for more efficient, scalable, and successful semiconductor development in the future

ACKNOWLEDGEMENTS

At this moment we would like to add a few heartful thanks for the people who made this project successful. We would like to take this pleasant moment to thank our management for creating a good environment to carry our project. Very special thanks to our Honorable Chairman **Dr. SARAN RAJ S** and CEO **Smt. SUJATHA MARAN**, formotivating us to carry out project successfully.

With deep sense of gratitude, we thank our beloved Principal of our institution **Dr. DEVARAJU A M.Tech., Ph.D.**, who has always beena constant source of inspiration and has been rendering his kind cooperation to complete our project.

No words of gratitude will sufficient for the unquestioning support extended to us by our Head of the Department and Internal Guide**Dr. DINESH BABU K, M.E, Ph.D.**, for being ever supporting force during our project work.

We wish to express our deep sense of gratitude to our project coordinator **Mrs. RUPADEVI N M.E.** for his able guidance and useful suggestions, which help us in completing the project work in time.

We also establish our grateful thanks to all our **FACULTY MEMBERS** of our Electronics and communication EngineeringDepartment for their able support and guidance.

Finally, yet importantly, we would like to express our heartfelt thanks to our beloved parents for their blessings, our friends and classmates for their help and wishes for the successful completion of this project.

VII. REFERENCES

[1] Apoorva H M and Dr. Kiran Bailey titled "UVM based Design Verification of FIFO" Published by International Journal of Engineering Research & Technology (IJERT) Vol. 9 Issue 06, June-2020

[2] Deepika, Jayanthi K Murthy 2020," Interrupt Enabled Priority Based Master Slave Communication using SPI Protocol" International Journal of Innovative Technology and Exploring Engineering (IJITEE).

[3] Abhishek Jain and Richa Gupta titled "Expanding the UVM Register Model towards Automation and Simplicity of Use" International Journal of Advanced Research in Computer Science Volume 8, No. 3, March – April 2017.

[4] Clifford E. Cummings and Janick Bergeron titled "Using UVM Virtual Sequencers & Virtual Sequences" World Class System Verilog & UVM Training DVCon 2016

[5] Josep Sans i Prats titled "Verification of a microprocessor's memory pipeline with UVM" Final Master Thesis Master in Innovation and Research in Informatics June 2022

[6] Bidisha Kashyap and Ravi V titled "Universal Verification Methodology Based Verification of UART

Protocol” 2020 Journal of Physics: Conference Series 1716 (2021) 012040

[7] Khaled Fathy and Khaled Salah titled “An Efficient Scenario Based Testing Methodology Using UVM” 2016 17th International Workshop on Microprocessor and SOC Test and Verification

[8] Juan Francesconi, J. Agustin Rodriguez, Pedro M. Julian titled “UVM Based Testbench Architecture for Unit Verification” 2014 Argentine School of Micro-Nanoelectronics, Technology and Applications.

[9] Shumaila Qamar, Wasi Haider Butt, Muhammad Waseem Anwar, Farooque Azam and Muhammad Qasim Khan, “A Comprehensive Investigation of Universal Verification Methodology(UVM) Standard for Design Verifiction,” ICSCA 2020, February 18–21, 2020, Langkawi, Malaysia