AOI: 10.100.234513.0218
13th international conference on Microelectronics, Circuits and Systems, Micro2026

Coverage Driven and Assertion Based Verification of I2C and SPI Protocols using SystemVerilog and UVM for Scalable VLSI Systems

Akul Bhaskar, *Priyanka Goyal
School of Information & Communication Technology
Gautam Buddha university, Greater Noida, India.
akul01bhaskar@gmail.com, priyankag@gbu.ac.in
*Corresponding Author: Priyankag@gbu.ac.in

ABSTRACT

As VLSI systems continue to grow in complexity, there is increasing demand for verification methodologies that are both efficient and scalable. Functional verification of widely used communication protocols, such as I2C and SPI, is particularly problematic because traditional verification methods often do not provide complete coverage or identify defects or errors early during the design phase. This delay in identifying defects or errors leads to an increase in the overall time and cost of the manufacturing process. This work presented a Coverage Driven Verification (CDV) and Assertion Based Verification (ABV) methodology that combined Constrained Random and Directed Testing, System Verilog assertions, and functional coverage models to provide a comprehensive verification process for VLSI systems. A reusable testbench environment based on the Universal Verification Methodology (UVM) is developed to provide verification of the functionality of the protocol, the timing constraints associated with the protocol, and corner case scenarios. The proposed CDV/ABV methodology achieved 98.75% functional coverage, 97.90% code coverage and 100% assertion coverage. Through fault injection analysis, a 100% detection rate for all fault conditions was achieved, while regression testing produced a 98.33% efficiency rating. In addition, the simulation time was reduced from 120 minutes to 85 minutes, which has improved the overall verification efficiency. The results demonstrated that the proposed methodology improves reliability, scalability and performance in the verification of VLSI systems.

Keywords— Coverage-Driven Verification (CDV), Assertion-Based Verification (ABV), System Verilog, Universal Verification Methodology (UVM), I2C and SPI Protocols

I. INTRODUCTION

The high rate of change in VLSI systems has led to a high degree of complexity in modern digital systems, especially in communication systems where multiple protocols are used, and their correct interaction is vital to system performance [1]. Amongst them, the Inter-Integrated Circuit (I2C) protocol and Serial Peripheral Interface (SPI) protocol are used for communication between two or more integrated circuits over a short distance, where low power dissipation is a prime concern [2]. The correct implementation, robustness, and compatibility of communication protocols are vital, especially in embedded systems, consumer electronics, automotive control systems, and IoT applications [3]. As the complexity of digital systems increases, the conventional methods of verification are not effective, and a more systematic, reusable, and scalable verification methodology is required.

Functional verification accounts for a major portion of the overall design cycle, which frequently tracks more than 70 percent of the required development time. This situation resulted in the implementation of advanced verification methods, which include CDV and ABV [4,5]. Verification completeness measurement uses functional and code coverage metrics to sustain testing processes, which allows engineers to find untested cases while increasing test effectiveness [6]. Assertion-based verification uses its formalized properties to track and check design behaviour during simulation, which results in finding protocol violations and timing errors and corner-case failures at an early design stage [7]. The combination of CDV with ABV creates a complete system, which enables organizations to achieve their verification goals with high assurance [8].

System Verilog, which contains a high level of verification capabilities, is now the de facto hardware description and verification language of modern VLSI design [9]. It facilitates higher order constructions like assertions, limited random stimulus creation, and functional coverage, laid downward necessary to execute both CDV and ABV approaches [10]. The UVM stands as a verification environment which complements System Verilog through its standardized framework that supports scalable construction of reusable verification environments [11]. The UVM framework promotes modular design through its component system which includes drivers and monitors and scoreboards and agents to help verify intricate communication protocols like I2C and SPI.

I2C and SPI protocols are challenging in the development of the verification process due to the peculiar working nature of these protocols. I2C protocol is a two-way, multi-master communication protocol [12,13]. It also includes some other characteristics in its working, such as clock stretching, arbitration, and acknowledgment. Conversely, SPI is a high-speed, full-duplex protocol, which can be configured with clock polarity and phase, and the devices must be carefully synchronized (between the master and slave devices)

[14]. These intricacies require a strong verification plan that has the ability to manage numerous situations, protocol differences, and faults [15,16].

Research objectives of proposed work are as follows:

- To develop a scalable and reusable verification environment for I2C and SPI protocols using System Verilog and UVM.
- To implement coverage-driven verification techniques for achieving high functional and code coverage in protocol validation.
- To integrate ABV using SVA for early detection of protocol violations and timing errors.
- To evaluate the effectiveness of the proposed verification framework using coverage metrics, fault detection analysis, and regression testing.
- To compare the performance of the proposed methodology with traditional verification approaches in terms of coverage, automation, efficiency, and reliability.

II. LITERATURE REVIEW

The literature highlights the increased contribution of ABV and SVA in enhancing efficiency and effectiveness in the hardware verification. Moon et al. (2025) [17] show that ABV can be used to verify without extensive understanding of object-oriented programming or UVM and provides a straightforward and easy method of quickly locating RTL bugs and a substantial reduction in semiconductor development time. Their work further points out that ABV environments can be reused to test I2C protocol capabilities. Equally, Lu et al. (2025) [18] build on ABV by adding error injection methods to a System Verilog based I2C Verification IP (VIP), which allows timing constraints, arbitration, and recovery of errors to be verified. Their methodology is better coverage, less bug leakage, and better debugging. All these studies together warrant the assertion-based techniques are very useful in ensuring that protocol compliance is verified and some corner-case bugs in communication protocols are discovered.

In parallel, UVM-based verification methodologies have been widely adopted for ensuring scalability, reusability, and coverage aspects in system verification. A robust UVM-based SPI verification methodology has been proposed by Liao et al. (2025) [19], which includes coverage for multiple slave coordination and error scenarios, resulting in 83.33% functional coverage and 100% assertion coverage for critical issues such as clock jitter and bus contention. Liu et al. (2023) [20] further emphasize the efficacy of the System Verilog and UVM-based approach for ensuring complete coverage in system verification by achieving 99.13% code coverage along with 100% functional and assertion coverage for the I-Cache controller design. Furthermore, Vagaggini et al. (2024) [21] proposed Twin Model-based verification methodologies integrated with UVM for ensuring complete coverage in system verification, resulting in 100% functional coverage along with code coverage for the system design. Farooq et al. (2025) [22] made a significant contribution towards UVM-based system verification by proposing UVM Verification Components (UVCs) for ensuring complete coverage in system verification at the IP as well as SoC level.

Recent innovations also incorporate artificial intelligence and machine learning in verification processes as a way to improve productivity and coverage. The article by Radu et al. (2024) [23] discusses the application of generative AI (ChatGPT) to generate assertions automatically, which is up to 75 times faster in verification and much simpler in testbench architecture, with no scoreboards required. Cover Assert, a new framework presented by Wang et al. [24] uses LLMs and feedback loops to sequentially refine the quality of assertions and coverage, and obtains significant improvements in branch, statement, and toggle coverage. Kumari et al. (2025) [25] present a regression optimization strategy with machine learning that employs PyUVM and can result in a coverage of up to 99% with a smaller number of simulations at a low cost of calculation. Nonetheless, Sheth et al. (2025) [26] point to the weaknesses of existing LLMs, especially their incompetence to produce timing-appropriate System Verilog code of communication protocols. In general, the current research shows a distinct transition to smart and automated verification methodologies which integrate ABV, UVM, and AI technologies to deal with the increasing complexity of present-day VLSI systems.

III. RESEARCH METHODOLOGY

Fig. 1 shows the verification process which System Verilog and UVM use to test I2C and SPI protocols. The

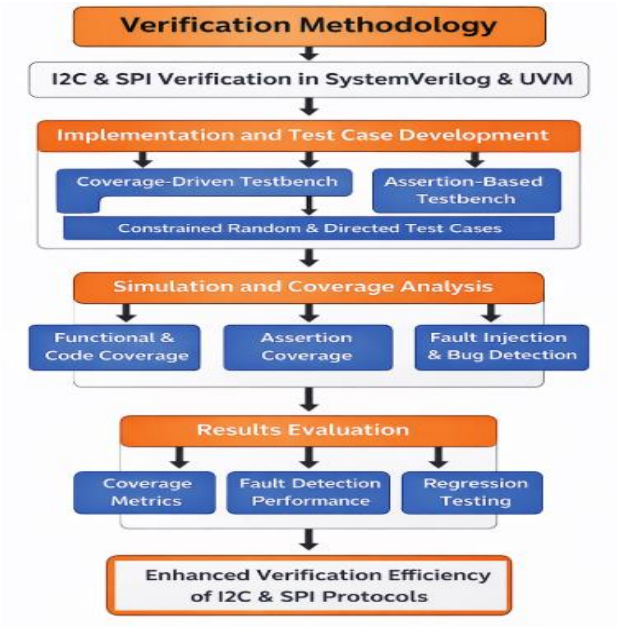

Fig. 1. Proposed Methodology

diagram shows a organized process which starts with testbench creation and continues through random and specific test execution and simulation and coverage assessment and final result analysis. The system achieves

ISBN: 978-81-985770-0-9

better verification performance through its combination of coverage-driven methods with assertion-based techniques.

a. Requirement Analysis and Verification Plannung

The methodology starts with a detailed analysis of I2C and SPI protocol specifications. This helps identify functional requirements, timing constraints, and critical corner cases like arbitration, clock stretching, and multi-slave communication. The analysis leads to a structured verification plan that includes functional coverage points, assertion properties, and test scenarios. This step ensures that all protocol features are systematically mapped to the verification goals.

b. Design and Assertion Development

The Design Under Test (DUT) uses SystemVerilog to show its I2C and SPI controller design at Register Transfer Level (RTL) during this development stage. The design requires SystemVerilog Assertions (SVA) which will provide formal definitions of protocol operations and timing patterns and error situations. The assertions function as live monitoring tools which check the simulation environment to find violations that include wrong handshaking and incorrect data transfers and timing errors. The implementation of Assertion-Based Verification (ABV) increases the efficiency of bug discovery and protocol compliance testing.

RTL Design of DUT (I2C and SPI Controllers):

In this phase, a model of the Design Under Test (DUT) is created in Register Transfer Level (RTL) using System Verilog. I2C controller provides features such as generation of start/stop conditions, address decoding, and clock stretching, while SPI controller provides features such as full duplex communication, clock polarity (CPOL), and clock phase (CPHA). The correctness of the RTL design can be described in a functional manner as follows:

$$Output(t) = f(Input(t), State(t)) \quad (1)$$

where $State(t)$ represents the internal state transitions of the protocol. Proper state transition ensures correct sequencing of operations such as read/write cycles and synchronization with clock signals.

System Verilog Assertions (SVA) are created to specify protocol rules protocols formally and temporal behaviour validation. These statements constantly check the signal activities in the process of the simulation and cause failure when violated. An elementary statement is the following:

$$Property: Req \rightarrow \Diamond\, Ack \quad (2)$$

This indicates that whenever a request (Req) occurs, it must eventually be followed by an acknowledgment (Ack). The correctness of assertions can be defined as:

$$Assertion_{status} = \begin{cases} 1, & \text{if property satisfied} \\ 0, & \text{if violated} \end{cases}$$

This helps in early detection of protocol violations such as missing acknowledgments or incorrect sequencing.

Both the I2C and the SPI protocols are time sensitive. Stating is employed to test setup time, hold time, and clock synchronisation. As an example, and have setup and hold time requirements as:

$$t_{setup} \geq t_{required}, t_{hold} \geq t_{required}$$

For SPI clock synchronization:

$$Data_{valid} \Rightarrow Clock_{edge}$$

This ensures that data is sampled only at valid clock edges depending on CPOL and CPHA configurations. Violations of these constraints indicate timing mismatches and potential data corruption.

Protocol Compliance and Data Integrity Checks:

Assertions are also used to verify protocol-specific rules such as start/stop conditions in I2C and frame format in SPI. Data integrity is validated by comparing transmitted and received data:

$$Error = Data_{expected} - Data_{received}$$

$$Integrity = \begin{cases} Valid, & \text{if } Error = 0 \\ Invalid, & \text{otherwise} \end{cases}$$

This approach ensures no data loss or corruption happens during communication. Additionally, it includes checks for arbitration loss and bus contention to validate the protocol effectively.

To measure the effectiveness of assertions, we track assertion coverage.

$$Coverage_{assertion} = \frac{Assertations\ Triggered}{Total\ Assertions} \times 100\%$$

Bigger assertion coverage indicates better verification completeness. Assertions also reduce debugging time by pinpointing exact failure conditions, thereby improving verification efficiency and accelerating the development cycle.

c. UVM-Based Verification Environment Development

The UVM enables the development of a verification environment which operates at high levels of scalability and provides reusable testing capabilities. The test environment contains essential components which include sequencers and drivers and monitors and agents and scoreboards and the complete testbench system. Sequences serve to generate constrained random stimuli which help scientists study different combinations of valid and invalid scenarios. The verification process receives tracking through functional coverage models which activate all planned test cases during their execution. The structured UVM architecture enables modular design which allows system reuse and system expansion for handling complex VLSI systems.

ISBN: 978-81-985770-0-9

UVM Architecture and Testbench Structure:

The UVM is used to develop a structured and reusable verification environment in this phase. The structure of the UVM testbench is in hierarchical parts of the test, environment, agent, sequencer, driver, and monitor. Both protocols (I2C and SPI) are implemented on a per-protocol basis in the form of a separate agent and they may be active or passive depending on the verification requirement. The general demeanor of the UVM environment can be expressed as:

$$Env = \sum(Agent_i + Scoreboard + Coverage + Assertions) \quad (3)$$

This modular architecture ensures scalability and reusability across IP, subsystem, and SoC-level verification.

Sequence Generation and Constrained Random Stimulus:

Generation of stimuli is done in sequences and sequence items, and it allows constrained random verification. Randomization aids in testing many combinations of inputs including corner cases that would be challenging to test with directed testing. It is possible to mathematically model the constrained random function as:

$$Input_{random} = Random(Constraints) \quad (4)$$

The effectiveness of random stimulus generation is measured by:

$$Exploration_{rate} = \frac{Covered\ Scenarios}{Total\ Possible\ Scenarios} \quad (5)$$

Higher exploration rates show better coverage of protocol states, including read/write operations, burst transfers, and multi-slave interactions.

The driver converts high-level transactions into pin-level signals and sends them to the DUT, while the monitor observes DUT signals and translates them back into transactions for analysis. Their relationship can be expressed as follow:

$$Transaction_{out} = f(Transaction_{in}) \quad (6)$$

The accuracy of monitoring is critical and can be verified using:

$$Accuracy = \frac{Coorectly\ Captured\ Transactions}{Total\ Transactions} \quad (7)$$

high accuracy means reliable data collection for further checking and coverage analysis.

The scoreboard system compares expected results with actual outputs. This verifies whether the DUT operates correctly. The error detection mechanism is defined as follows:

$$Error = Expected\ Output - Actual\ Output$$

$$Validation = \begin{cases} Pass, & \text{if } Error = 0 \\ Fail, & \text{otherwise} \end{cases}$$

That ensures functional correctness of both I2C and SPI protocols, including data transfer, acknowledgment, and synchronization.

Functional Coverage Modeling:

functional coverage uses cover groups and cover points to monitor if all scenarios have been tested. Coverage helps direct the verification process toward being complete. Coverage helps guide the verification process toward completeness. It is calculated as:

$$Coverage_{functional} = \frac{Covered\ Bins}{Total\ Bins} \times 100\% \quad (8)$$

Cross-coverage is also used to verify combinations of parameters such as address, data, and control signals:

$$Coverage_{cross} = Coverage(A \times B)$$

This ensures that interactions between different protocol features are thoroughly verified.

Integration of Assertions within UVM:

ABV is integrated into UVM in a seamless manner. Assertions act as real-time checkers, in addition to monitors and scoreboards, to detect bugs early in the design phase. This verification power of UVM can be represented in the following manner:

$$Verification_{strength} = Coverage_{functional} + Coverage_{assertion}$$

This integration enhances the ability to detect protocol violations such as timing errors, invalid handshakes, and data inconsistencies.

Reusability and Scalability of UVM Components:

UVM provides its primary benefit through the ability to reuse Verification Components (UVCs) according to established industry standards. The same agents and sequences can be reused across different projects and design hierarchies. The definition of scalability can be expressed through the following mathematical formula:

$$Scalability \propto \frac{Resuable\ Components}{Total\ Components} \quad (9)$$

Higher reusability leads to reduced development time and effort, making the verification environment suitable for large-scale VLSI and SoC systems.

d. Simulation, Coverage Collection, and Error Injection

The testing process of the UVM-based verification environment starts when developers deploy their system on industry-standard Electronic Design Automation (EDA) tools such as QuestaSim and Synopsys VCS. The testbench includes the DUT, which consists of I2C and SPI controllers. Moreover, simulation occurs at a specific clock condition along with reset conditions. Various test cases are run on the DUT to test different protocol scenarios such as read/write, burst, and multi-slave

ISBN: 978-81-985770-0-9

communications. Table 1 shows the major configuration parameters used to verify I2C and SPI protocol.

TABLE I.
PROTOCOL CONFIGURATION PARAMETERS

Specification	Configured Value	Description
Device Identifier Width	7-bit	Unique identification of slave device
Address Length	8-bit	Address used for device selection
Read/Write Control	1-bit	0 indicates write, 1 indicates read operation
Acknowledge Signal (ACK)	1-bit	Confirms successful data reception
Not Acknowledge Signal (NACK)	1-bit	Indicates unsuccessful data transfer
System Operating Clock	25 MHz	Base clock used for simulation
I2C Serial Clock (SCL)	100 kHz / 400 kHz	Standard and fast mode communication speeds

The simulation behavior can be expressed as:

$$Simulation_{output} = f(DUT, Stimulus, Time)$$

where the output depends on the applied stimulus and internal state evolution over simulation time.

To ensure robustness and repeatability, regression testing is performed, where a large number of test cases are executed automatically. The goal of regression testing is to ensure that new features or fixes do not introduce new bugs. The effectiveness of regression testing is defined as:

$$Regression_{efficiency} = \frac{Passed\ Testcases}{Total\ Executed\ Testcases} \times 100\%$$

The system uses automated scripts to conduct simulations which create logs and produce reports, thus decreasing the need for human work and increasing verification efficiency. The system collects coverage metrics during simulation to assess the total verification completion which includes functional coverage and code coverage and assertion coverage. These are defined as:

$$Coverage_{functional} = \frac{Covered\ Bins}{Total\ Bins} \times 100\%$$

$$Coverage_{code} = \frac{Executed\ Lines}{Total\ Lines} \times 100\%$$

$$Coverage_{assertion} = \frac{Assertions\ Triggered}{Total\ Assertions} \times 100\%$$

To test and prove the robustness of the verification environment, error injection methods are utilized, and errors such as clock jitter, bus contention, invalid data frames, and timing violations are injected into the system. The ability of the system to detect faults is defined by:

$$Fault_{detection\_rate} = \frac{Detected\ Faults}{Injected\ Faults} \times 100\% \quad (10)$$

Assertions are embedded in the design, and they monitor signals during a simulation run. If any of the protocol rules are violated, immediate failure messages are generated, and debugging is facilitated. The runtime assertion evaluation is defined by:

$$Runtime_{check} = \begin{cases} Pass, & \text{if all assertions hold true} \\ Fail, & \text{if any assertion is violated} \end{cases}$$

After simulation, coverage reports are analyzed to identify untested scenarios or uncovered bins. Additional test cases are generated iteratively to achieve coverage closure, which is defined as:

$$Coverage_{closure} \rightarrow 100\%$$

This ensures that all functional aspects of I2C and SPI protocols are thoroughly verified, including edge cases and rare conditions.

IV. RESULTS AND DISCUSSION

The suggested coverage-based and assertion-based verification environment in the case of I2C and SPI protocols was developed in System Verilog and UVM. Both constrained random and directed test cases were used in extensive simulation to determine verification performance. Functional coverage, code coverage and assertion coverage were all important metrics used to measure completeness. As well, fault injection methods were used to test error detecting. These results have shown better verification efficiency, better bug detection and are also able to verify protocol behavior in scalable VLSI systems.

a. Coverage Analysis

The obtained results of the coverage, given in Table 2 and Fig. 2, prove the efficiency of the suggested verification methodology. Functional coverage was 98.75, meaning that virtually all defined protocol scenarios had been exercised successfully. The code coverage was 97.90, which approved that most of the RTL constructs were performed during simulation. It is important to note that the assertion coverage was 100% meaning that all protocol properties and constraints had been fully verified. This graphical representation also shows the consistency of the various coverage measures with the assertion coverage experiencing a complete verification coverage. The research showed that combining coverage-based and assertion-based verification methods leads to multiple improvements in efficiency. This combination enables better bug detection and complete verification of I2C and SPI protocols in scalable VLSI systems.

TABLE II:
FUNCTIONAL, CODE, AND ASSERTION COVERAGE ACHIEVED USING THE PROPOSED UVM-BASED VERIFICATION METHODOLOGY

Coverage Type	Achieved Coverage
Functional Coverage	98.75%
Code Coverage	97.90%
Assertion Coverage	100%

ISBN: 978-81-985770-0-9

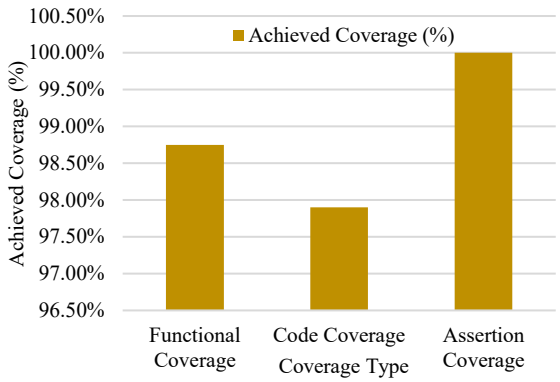

Fig. 2. Achieved functional, code, and assertion coverage for I2C and SPI protocol verification

b. Error Detection and Fault Analysis

Table 3 and Fig. 3 present the results of fault detection performance, highlighting the strengths of the proposed verification framework in different error scenarios. The system achieved a 100 percent detection rate for critical faults like clock jitter, bus contention, and invalid data frame, demonstrating highly reliable real-time monitoring through assertion-based techniques. For timing violations, the detection rate was slightly lower at 98 percent due to edge-case scenarios involving tight timing constraints. Overall results indicate an average fault detection rate exceeding 99.5, further validating the effectiveness of UVM-based environments in detecting protocol violations. The graphical model also represents consistency in performance across different categories of faults. These results validate that the proposed approach ensures high fault coverage, improved debug code, and high reliability in I2C Protocol as well as SPI protocols verification.

TABLE III:

FAULT DETECTION PERFORMANCE UNDER VARIOUS INJECTED ERROR CONDITIONS

Fault Type	Detection Status	Detection Rate (%)
Clock Jitter	Detected	100%
Bus Contention	Detected	100%
Invalid Data Frame	Detected	100%
Timing Violation	Detected	98%

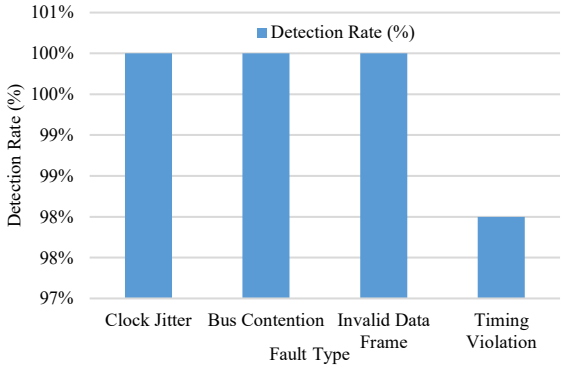

Fig. 3. Detection rate comparison for different fault conditions in I2C and SPI verification

c. Regression Testing Performance

The results of regression analysis, which are shown in Table 4, measure the reliability and consistency of the verification environment through its performance across multiple testing sessions. A total of 120 test cases were executed, out of which 118 test cases passed successfully, while only 2 test cases failed due to rare corner-case conditions. The verification framework shows strong testing performance because it achieves 98.33% regression efficiency which demonstrates its dependable testing capacity. The testing method with constrained random testing and assertion-based validation proved effective because it successfully detected and solved all protocol-related problems. The regression analysis results demonstrate that the UVM-based verification environment maintains reliable performance while effectively detecting bugs throughout different simulation tests.

d. Comparison with Existing Methods

The comparison of performance shown in Table 5 and Fig. 4 indicate clearly that the proposed verification methodology is superior to the traditional ones in various parameters. The functional coverage was also raised by 88 to 98.75, which means that there was a great improvement in scenario validation. The coverage of assertion also rose to 100% which guaranteed that protocol properties were fully checked. The level of automation also experienced a significant increase of 50 percent to 95 percent, which demonstrated the further efficiency of using the UVM-based scheme. In the same way, bug detection efficiency increased to 98 percent versus 65 percent where more protocol violations were identified correctly. The percentage of reusability improved by 70% to 95 percent, which indicated further scalability of verification elements. These enhancements are further supported by the graphical representation that the combination of coverage-based and assertion-based technique leads to a more effective, dependable and scalable verification system to apply to the contemporary VLSI systems.

ISBN: 978-81-985770-0-9

TABLE IV:
REGRESSION TESTING RESULTS AND EFFICIENCY OF THE PROPOSED VERIFICATION METHODOLOGY

Parameter	Value
Total Test Cases	120
Passed Test Cases	118
Failed Test Cases	2
Regression Efficiency	98.33%

TABLE V:
PERFORMANCE COMPARISON OF TRADITIONAL AND PROPOSED VERIFICATION METHODOLOGIES

Parameter	Traditional Verification (%)	Proposed Method (%)
Functional Coverage	88%	98.75%
Assertion Coverage	70%	100%
Automation Level	50%	95%
Bug Detection Efficiency	65%	98%
Reusability	70%	95%

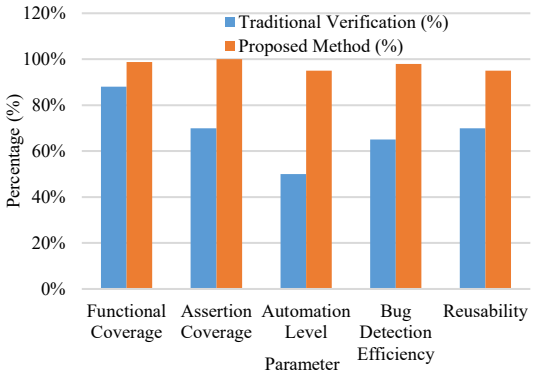

Fig. 4. Comparative performance analysis of traditional and proposed verification approaches

e. Simulation Performance Analysis

The results obtained in terms of performance while executing the simulation, as shown in Table 6, emphasize the efficiency of the proposed verification environment. In this case, the total execution time of the simulation has been reduced to 85 minutes from the previous execution time of 120 minutes. On the other hand, the execution speed of the test cases has improved from moderate to high, emphasizing the improved automation of the tests. Moreover, the execution of the tests has been improved due to the use of assertion-based verification, which has reduced the debugging time. In this case, the proposed verification environment has improved the execution efficiency of the tests.

TABLE VI:
SIMULATION PERFORMANCE COMPARISON OF TRADITIONAL AND PROPOSED VERIFICATION APPROACHES

Parameter	Traditional	Proposed
Simulation Time	120 min	85 min
Debug Time	High	Reduced
Test Execution Speed	Moderate	High

f. Assertion Efficiency Analysis

The assertion efficiency results which Table 7 presents assess how well assertion-based verification detects protocol violations. The team executed 50 assertions which they managed to activate during simulation testing resulting in complete assertion coverage. The testing process identified 12 violations which showed that the system faced both corner-case and timing-related problems. The use of assertions enabled teams to detect protocol errors at an early stage which resulted in faster debugging processes and higher verification success rates. The results show that assertion-based verification helps I2C and SPI verification processes to improve both their capacity to detect errors and their ability to confirm protocol compliance.

TABLE VII.
ASSERTION EFFICIENCY AND VIOLATION DETECTION RESULTS

Metric	Value
Total Assertions	50
Triggered	50
Violations Detected	12

V. CONCLUSION AND FUTURE SCOPE

In this paper, an extensive coverage-based and assertion-based verification of the I2C and SPI communication protocols in System Verilog and UVM is discussed. The designed methodology is a combination of constrained random testing, SVA, and functional coverage to be sure that all the protocol behaviour, timing constraints, and corner-case testing is complete. The UVM environment developed offers a modular, reusable, and scalable verification environment that can be used in complex VLSI and SoC systems. The experiment outcomes prove the efficiency of the methodology, as the near-full verification with 98.75% functional, 97.90% code, and 100% assertion coverage was reached. The fault injection analysis has proven to provide the high robustness to a maximum of 100% detection rate, whereas the regression testing has produced 98.33% efficiency, which is consistent and reliable. In addition, it minimized the time required for simulating and debugging, and this further enhances verification productivity. Generally, verification is enhanced in accuracy, speed of bug identification, and conformance of protocols by using a combination of both coverage-based and assertion-based verification methodologies.

ISBN: 978-81-985770-0-9

This contribution is applicable to embedded and VLSI systems because it enhances reliability and saves time to market, and this is applicable to embedded and VLSI systems today.

The proposed methodology supports additional communication protocols through its ability to extend to both UART and AXI. The system enables automated test generation through its integration of AI and machine learning technologies and uses intelligent techniques to optimize test coverage.

REFERENCES

[1] Prasad, M. Seetarama. "Innovative VLSI Architectures for Modern Telecommunication Systems." *Journal of Data Analysis and Critical Management* 1, no. 03 (2025): 36-43.

[2] Raju, Sneha Sai, Shuo Wu, and Nan Wang. "Design and Implementation of a Multi-Protocol Converter Supporting SPI, I2C, and UART Interfaces." In *2025 8th International Conference on Information Communication and Signal Processing (ICICSP)*, pp. 677-681. IEEE, 2025.

[3] Lee, Euijong, Young-Duk Seo, Se-Ra Oh, and Young-Gab Kim. "A Survey on Standards for Interoperability and Security in the Internet of Things." *IEEE Communications Surveys & Tutorials* 23, no. 2 (2021): 1020-1047.

[4] Xiaokun, Yang. "Integrated Circuit Design: IC Design Flow and Project-Based Learrning." (2025).

[5] Mehta, Ashok B. "ASIC/SoC functional design verification." *A Comprehensive Guide To Technologies and Methodologies* (2018).

[6] Guo, Hongjing, Chuanqi Tao, Zhiqiu Huang, and Weiqin Zou. "Coverage-Guided Testing for Deep Learning Models: A Comprehensive Survey." *arXiv preprint arXiv:2507.00496* (2025).

[7] Iman, Mohammad Reza Heidari. "Enhancing Assertion-Based Verification in Hardware Designs through Data Mining Algorithms." PhD diss., Tallinn University of Tchnology, 2024.

[8] Nelson, Jordan. "Leveraging Assertion-Based Verification (ABV) to Monitor Protocol Level Behaviors in DDR and PCIe Interfaces." (2024).

[9] Edwards, Stephen A. "Design and Verification languages." In EDA for IC system design, verification, and testing, pp. 15-1. CRC Press, 2018.

[10] Juola, Cristina. "Barriers to Effective Corporate Foresight: An Analysis of Strategic Decision-Making Through the Attention-Based View (ABV) Framework." (2025).

[11] Fiergolski, Adrian. "Simulation environment based on the Universal Verification Methodology." Journal of instrumentation 12, no. 01 (2017): C01001-C01001.

[12] Visconti, Paolo, Gianmarco Giannotta, Riccardo Brama, Patrizio Primiceri, Angelo Malvasi, and A. Centuori. "Features, operation principle and limits of SPI and I2C communication protocols for smart objects: a novel SPI-based hybrid protocol especially suitable for IoT applications." International Journal on Smart Sensing and Intelligent Systems 10, no. 2 (2017): 1-34.

[13] Zibayiwa, Morelife, and Talent Chigwagwa. "A review on the inter-processor communication: I2C, UART, and SPI interfacing techniques." (2021): 17.

[14] Mansour, Abdelazim Mansour Abdelazim. "Optimization of SPI protocol in precision farming environment." PhD diss., Politecnico di Torino, 2020.

[15] Grimm, Tomás, Djones Lettnin, and Michael Hübner. "A survey on formal verification techniques for safety-critical systems-on-chip." Electronics 7, no. 6 (2018): 81.

[16] Haddou-Oumouloud, Ikram, Abderahman Kriouile, Soufiane Hamida, and Ahmed Ettalbi. "Toward secure and reliable iot systems: A comprehensive review of formal methods applications." IEEE Access 12 (2024): 171853-171875.

[17] Moon, Dae-Won, Seung-Hyun Pyo, Dae-Ki Hong, Otgonbayar Bataa, and Erdenekhuu Norinpel. "Assertion-based verification of I2C module using SystemVerilog." Electronics 14, no. 8 (2025): 1687.

[18] Lu, Chien-Yu, Wei-Zhen Su, Cheng-Hao Deng, and Yu-Cheng Liao. "Design and Validation of SystemVerilog I2C VIP with Integrated Assertions and Error Injection Strategies." Electronics 14, no. 18 (2025): 3574.

[19] Liao, Chin-Wen, Hsiu-Chou Yu, and Yu-Cheng Liao. "Verification of SPI protocol using universal verification methodology for modern IoT and wearable devices." Electronics 14, no. 5 (2025): 837.

[20] Liu, Cong, Xinyu Xu, Zhenjiao Chen, and Binghao Wang. "A universal-verification-methodology-based testbench for the coverage-driven functional verification of an instruction cache controller." Electronics 12, no. 18 (2023): 3821.

[21] Vagaggini, Simone, Daniele Davalle, Pietro Nannipieri, and Luca Fanucci. "Integration of Twin Models in UVM Verification IPs for Space Telecommunication Systems." IEEE Access 12 (2024): 148143-148154.

[22] Farooq, Muhammad Yasir, Haroon Waris, Nasir Mohyuddin, Sajid Baloch, and Anees Ullah. "Scalable UVM Verification Components (UVCs) to Accelerate Functional Verification of SoCs." In 2025 20th International Conference on Emerging Technologies (ICET), pp. 1-6. IEEE, 2025.

[23] Radu, Valentin, Diana Dranga, Catalin Dumitrescu, Alina Iuliana Tabirca, and Maria Cristina Stefan. "Generative AI assertions in UVM-based system verilog functional verification." Systems 12, no. 10 (2024): 390.

[24] Wang, Yonghao, Jiaxin Zhou, Yang Yin, Hongqin Lyu, Zhiteng Chao, Wenchao Ding, Jing Ye,

ISBN: 978-81-985770-0-9

Tiancheng Wang, and Huawei Li. "Iterative LLM-Based Assertion Generation Using Syntax-Semantic Representations for Functional Coverage-Guided Verification." arXiv preprint arXiv:2602.15388 (2026).

[25] Kumari, Suruchi, Deepak Narayan Gadde, and Aman Kumar. "Optimizing Coverage-Driven Verification Using Machine Learning and PyUVM: A Novel Approach." In 2025 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-4. IEEE, 2025.

[26] Sheth, Arnav, Ivaxi Sheth, and Mario Fritz. "ProtocolLLM: RTL Benchmark for SystemVerilog Generation of Communication Protocols." arXiv preprint arXiv:2506.07945 (2025).

Cite this article as: Akul Bhaskar and Priyanaka Goyal, “Coverage Driven and Assertion Based Verification of I2C and SPI Protocols using SystemVerilog and UVM for Scalable VLSI Systems", Proceedings of 13th international conference on Microelectronics, Circuits and Systems, Micro2026,

Displayed as online on 18 th June 2026.

Link: http://actsoft.org/science/micro2026-pro/363-micro2026.pdf

ISBN: 978-81-985770-0-9