

Development of UVM-Based PCIe Non-FLIT Scoreboard and Protocol Compliance Checker

Varsha Kuruba¹, Dr. P. Ramana Reddy²

¹ Department of Electronics and Communication Engineering,
Jawaharlal Nehru Technological University College of Engineering,
Anantapur, Andhra Pradesh, India.
kurubavarsha8639@gmail.com

² Senior Professor, Department of Electronics and Communication Engineering,
Jawaharlal Nehru Technological University College of Engineering,
Anantapur, Andhra Pradesh, India.
prr.ece@jntua.ac.in

Abstract—Abstract— The increasing demand for higher data transfer rates in contemporary computing systems has significantly increased the complexity of the PCI Express (PCIe) protocol, creating substantial challenges for functional verification and debugging. This paper presents a UVM-based verification environment for PCIe Endpoint validation, with an enhanced scoreboard as the primary component for transaction-level data integrity and protocol compliance verification. The proposed environment models communication between a Root Port and an Endpoint using PIPE-based simulation and AXI-Stream interfaces, supporting Memory Read, Memory Write, Configuration Read/Write, and Programmed I/O operations. A protocol-aware driver generates single- and multi-beat TLP transactions, while Link Training and Status State Machine (LTSSM) monitoring synchronizes stimulus generation with valid link states.

The proposed scoreboard maintains a reference model using shadow memory to track data written by Memory Write transactions and uses tag-based pending-request tracking to correlate Memory Read requests with their corresponding Completion responses. It performs end-to-end comparison between expected data maintained in the reference model and actual data received from the Endpoint, thereby identifying data mismatches and unexpected responses. The scoreboard also analyzes Completion status and detects verification conditions including Unsupported Request (UR), Completer Abort (CA), missing or orphaned completions, tag reuse, and other transaction-level protocol violations. Structured pass/fail reporting and transaction tracking provide improved visibility into the source of verification failures and reduce dependence on manual waveform analysis. Experimental results demonstrate that the proposed scoreboard-based methodology can identify transaction and protocol errors while confirming correct data integrity during valid PCIe operations. The modular UVM architecture also provides a reusable foundation for extending verification toward additional PCIe features and future generations.

Index Terms—PCI Express (PCIe), Universal Verification Methodology (UVM), nonFLIT, Transaction Layer Packet (TLP), Scoreboard, Shadow Memory, Completion Handling, AXI-Stream, LTSSM, Protocol Compliance, Functional Verification.

I. INTRODUCTION

The PCI Express (PCIe) protocol is one of the most widely adopted high-speed interconnect standards in contemporary

computing systems, permitting efficient communication between processors and peripheral devices, including graphics cards, storage devices and network interfaces. Increasing demands for higher throughput and lower latency have driven substantial development across PCIe generations, leading to greater architectural complexity and more challenging verification requirements. In non-FLIT PCIe communication, data exchange is performed using Transaction Layer Packets (TLPs), which are subject to exacting protocol requirements regarding packet formatting, address alignment, byte enables, ordering rules and completion handling. Violations of these requirements can result in data errors, incorrect system operation or critical functional failures. As PCIe systems increase in complexity, identifying such issues during pre-silicon verification becomes increasingly difficult. Traditional verification strategies, including directed testing and manual waveform analysis, are frequently inadequate for large-scale transaction scenarios and corner-case conditions. These approaches are labor-intensive and susceptible to human error, particularly when spotting subtle issues such as missing completions, incorrect responses, or transaction mismatches. Consequently, an automated and scalable verification Methodology is vital for reliable PCIe validation. Universal Verification Methodology (UVM) provides a standardized structure for constructing reusable and modular verification environments. In UVM, the scoreboard functions as a reference model that tracks transactions, predicts expected responses and compares them with the actual outputs from the Design Under Test (DUT). This automated comparison mechanism improves verification accuracy and debugging efficiency. However, designing an efficient scoreboard for PCIe verification remains challenging due to requirements for transaction monitoring, response correlation, and protocol violation detection. The verification environment must accurately handle Memory Read, Memory Write, Configuration Read, and Configuration Write operations while guaranteeing precise validation of completion. This paper presents the design and implementation of an automated scoreboard integrated within a UVM-based ver-

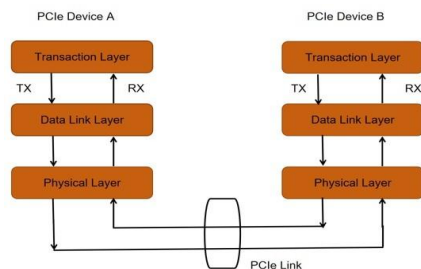


Fig. 1. PCIe Architecture

ification environment for PCIe Endpoint validation. The proposed framework simulates communication between the Root Port and Endpoint using a PIPE-based simulation setup and AXI-Stream interfaces. The scoreboard maintains an internal reference model for transaction monitoring and performs end-to-end data comparisons to detect protocol violations, such as missing completions, incorrect responses, and transaction mismatches. The principal contributions of this study are summarized as follows:

- Development of a strong and automated scoreboard for PCIe transaction verification.
- Integration of the scoreboard within a UVM-based modular verification environment.
- Implementation of mechanisms for detecting protocol violations and transaction inconsistencies.
- Enhancement of debugging efficiency through automated checking and structured reporting.

II. LITERATURE SURVEY

PCI Express (PCIe) has become a dominant high-speed interconnect standard in contemporary computing systems, and its increasing complexity has made verification a major challenge. Researchers have addressed this problem by developing verification methodologies targeting different layers of the PCI Express protocol stack using Universal Verification Methodology (UVM). This section reviews two key studies that form the foundation of the proposed approach.

A. PCIe Transaction and Data Link Layer Verification Using UVM

Jagtap et al. [4] presented a UVM-based Verification IP targeting the Transaction Layer and Data Link Layer of the PCIe protocol. Their work focuses on testing the functional correctness of Transaction Layer Packets (TLPs) and assuring reliable data transfer through Data Link Layer mechanisms. The proposed verification environment follows a standard UVM architecture consisting of sequencer, driver, monitor, agent, and scoreboard components. The testbench generates

various PCIe transactions, including Memory, I/O, and configuration operations, and verifies correct packet formation and response handling. The authors show successful verification through simulation results that show correct TLP transmission and reception, as well as efficient error detection using CRC-based mechanisms at the Data Link Layer. This work defines an organized approach to PCIe verification and highlights UVM's effectiveness in directing complex protocol validation. However, the scoreboard implemented in this work performs only basic comparisons between transmitted and received transactions. It does not maintain a reference model to track previously written data, nor does it correlate read requests with their corresponding completion responses. Furthermore, the approach lacks mechanisms to detect advanced protocol violations, such as missing completions, incorrect tag usage, or error responses including Unsupported Requests and Completer Aborts. These limitations restrict the verification environment's ability to spot subtle but critical functional errors.

B. Physical Layer Verification and LTSSM Analysis

R. B. K. and K. A. [5] focused on the development of a Verification IP for the Physical Layer of PCIe using UVM. Their work emphasizes validating Physical Layer functionalities, including encoding/decoding, data transmission, and link initialization behavior. A key contribution of this work is the detailed verification of the Link Training and Status State Machine (LTSSM), which governs the initialization and operational states of the PCIe link. The authors implemented a UVM-based testbench to verify Physical Layer components and demonstrated correct behavior through simulation results, including successful link training, mode changes, and data transfer validation. Their analysis of LTSSM provides valuable insight into link initialization and recovery mechanisms, which are key to ensuring reliable communication between PCIe components. Although it has strengths, this work mainly addresses Physical Layer verification and does not consider higher-layer transaction validation or end-to-end data validity checking. Specifically, it lacks mechanisms for correlating transmitted requests with received completions or detecting transaction-level protocol violations. Consequently, its usefulness is limited for complete full-stack PCIe verification.

C. Integration of Prior Work and Motivation

Both reviewed works add significantly to PCIe verification via addressing different layers of the protocol stack. Jagtap et al. [4] provide a foundation for Transaction and Data Link Layer verification using UVM, while R.B.K. and K. A. [5] offers thorough insights into Physical Layer behavior and LTSSM operation. Together, these works demonstrate a layered approach to PCIe verification, where each protocol layer is validated independently using organized methodologies. However, neither study delivers a comprehensive solution for transaction-level verification that ensures complete end-to-end correctness. Notably, the absence of an advanced scoreboard capable of tracking transactions, correlating requests with

responses, and detecting protocol violations amounts to a significant gap. Furthermore, although LTSSM behavior is analyzed in detail, its incorporation into higher-level verification flows for stimulus generation control is not fully addressed.

D. Research Gap and Contribution

Analysis of the current literature indicates that current PCIe verification approaches lack strong mechanisms for complete transaction-level validation. Comprehensive transaction-level validation. The scoreboard presented in prior work is limited to basic comparison and does not incorporate advanced features such as transaction tracking, reference modeling, or error classification. Furthermore, there is insufficient integration between the Physical Layer link status (LTSSM) and higher-layer stimulus control, resulting in incorrect test scenarios and false error detection. To address these limitations with greater methodological rigor, the proposed approach introduces an enhanced scoreboard as an integral component of a UVM-based verification environment for PCIe Endpoint validation. This design choice is justified by the necessity to systematically capture and compare all relevant transactions, thereby ensuring protocol compliance. Specifically, the scoreboard maintains a reference model within a shadow memory structure to continuously track all written data and utilizes tag-based tracking to establish an explicit association between read requests and their corresponding completion responses. This methodology enables precise end-to-end data comparison, facilitating early and reliable detection of discrepancies. Furthermore, explicit mechanisms are incorporated to identify protocol violations, such as missing completions, unexpected responses, and incorrect transaction sequencing, thus ensuring comprehensive and accurate verification outcomes. By adopting this methodological framework, the verification environment achieves an improved ability to detect errors that would likely remain unnoticed using less sophisticated approaches. Additionally, the proposed environment incorporates LTSSM monitoring into the verification flow to ensure that all transfers are generated only when the PCIe link is in a valid operational state. This merging prevents false failures resulting from premature stimulus generation and strengthens overall verification reliability. By combining advanced scoreboard design with LTSSM-aware stimulus control, the proposed approach delivers a stronger, scalable system for PCIe verification.

III. VERIFICATION ARCHITECTURE

The proposed verification environment is constructed using a standard UVM hierarchy and targets the Xilinx PCIe3 UltraScale IP operating in PIPE-based simulation mode. The architecture contains four primary layers: the DUT layer, which contains the PCIe endpoint; the interface layer, which connects testbench signals to the DUT; the UVM component layer, which includes all active and passive verification components; and the test and sequence layer, which drives all stimulus.

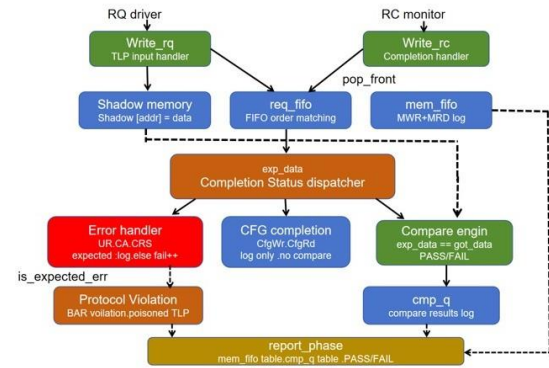


Fig. 2. PCIe UVM-Based Completion Handling and Data Comparison Architecture

A. Overall System Architecture

The environment models a complete PCIe system with two connected devices. The Xilinx PCIe3 UltraScale End-point functions as the DUT, while a Root Port model from the Xilinx example design acts as the initiating device that generates all requests. Communication between the devices occurs via a PIPE-based simulation interface, which accurately models physical-layer behavior without the need for real serial transceivers. The AXI4-Stream interface connects the verification environment to the PCIe core, with the AXI-Src channel transmitting outgoing request TLPs and the AXI-Dst channel receiving incoming completion TLPs.

B. UVM Component Hierarchy

The testbench contains two agents. The RQ agent is an active agent that drives outgoing TLPs onto the SaxisRQ interface through a dedicated driver. It contains a sequencer that receives sequence items from the test layer and passes them to the driver for pin-level driving. The RC agent is a passive agent containing only a monitor. The RC monitor continuously samples the Maxisrc interface, registers every incoming completion TLP, classifies each by status, and forwards it to the scoreboard via a UVM analysis port. The scoreboard is instantiated at the environment level and automatically receives all completion observations.

C. Interface Signal Widths

An important aspect of the environment is the accurate declaration of AXI-Stream signal widths for the 256-bit straddle mode utilized by the Xilinx IP. The saxisrqready signal is four bits wide, the maxisrcready signal is twenty-two bits wide, the maxisrctuser sideband is seventy-five bits wide, and the main data buses are two hundred fifty-six bits wide. Incorrect width declarations cause the simulator to silently truncate signals, resulting in flow-control failures that appear as spurious UR completions in the simulation log.

IV. TRANSACTION MODEL AND STIMULUS GENERATION

A. TLP Transaction Item

All transactions in the environment are represented by a single class called `pcietlptitem`, which carries all fields needed to describe any PCIe TLP type: request type, tag, traffic class, length in double words, first and last byte enables, address, requester ID, completer ID, completion flag, byte count, lower address, poison bit, and a dynamic data array. The request type field distinguishes between MemoryRead (MRd), Memory-Write (MWr), ConfigurationRead/WriteType0, and their Type 1 variants.

B. Sequence Hierarchy

Six sequence classes are implemented, one per TLP type. The `pcie_cfg_wr_seq` and `pcie_cfg_rd_seq` classes drive Configuration Write and Read TLPs for BAR programming and scanning, respectively. The `pcie_mwr_32_seq` and `pcie_mwr_64_seq` classes drive 32-bit and 64-bit MWr TLPs; each calls the scoreboard's track write function before driving so that shadow memory is populated before the write reaches the DUT. The `pcie_mrd_32_seq` and `pcie_mrd_64_seq` classes drive MRd TLPs and call track read before driving, registering the pending tag and target address in the tag tracker.

C. Driver Implementation

The driver converts sequence items into pin-level AXI-Stream transactions. For single-beat TLPs configuration reads, writes, and short memory reads the complete TLP descriptor is packed into one 256-bit beat with `tvalid` and `tlast` asserted simultaneously. For MWr TLPs with payloads exceeding four double words, a multi-beat transfer is used: the first beat carries the header and the first four data words in the upper 128 bits; subsequent beats carry up to eight data words each. The driver monitors `s_axis_rq_tready[0]` every clock cycle and holds the current beat until the endpoint signals readiness.

D. LTSSM Synchronization

A dedicated LTSSM monitor task runs in parallel with the driver throughout the simulation. It samples `cfg_ltssm_state` on every rising clock edge and logs state changes with the link-up and reset status. All sequence activity is gated until the LTSSM reaches state 0x10 (L0) and `user_lnk_up` is asserted, preventing false UR completions caused by premature TLP injection.

E. BAR Scan and Program Sequence

Before any memory transaction can succeed, all Base Address Registers (BARs) must be programmed. The test sequence first scans each BAR by writing 0xFFFFFFFF and reading it back to determine aperture size, then programs each BAR with its correct base address. For 64-bit BAR pairs, two consecutive writes are required: the lower 32 bits to the primary register and the upper 32 bits to the immediately following register. Omitting the upper write leaves the endpoint address decoder unable to match 64-bit transactions, causing all such requests to return UR status. The Command Register is written last to enable Memory Space access before any memory transactions are issued.

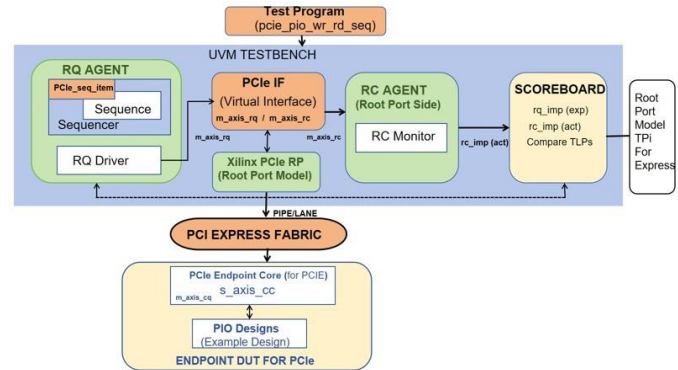


Fig. 3. PCIe UVM Testbench Architecture (PIO Design)

V. SCOREBOARD DESIGN

The scoreboard serves as the central verification component within the proposed environment. Implemented as a UVM scoreboard class, it receives all completion observations from the RC monitor via a UVM analysis import. It provides two public functions, track write and track read, which sequences invoke directly before driving each TLP. Internally, the scoreboard maintains two associative arrays: a shadow memory indexed by 64-bit address and a pending address table indexed by 8-bit tag, along with counters for pass, fail, and per-status completions.

A. Shadow Memory Management

The track write function is called by every MWr sequence before driving its TLP. It receives the base address and data array, storing each double word at the correct byte-aligned address in shadow memory. For a write of N double words starting at address A, shadow[A] receives data[0], shadow[A+4] receives data[1], and so on. Every registered write is logged with its address and data value, allowing verification without opening a waveform viewer.

B. Tag-Based Read Tracking

The track read function is called by every MRd sequence before driving. It stores the target address in the pending address array indexed by tag and clears the completion valid flag. If the tag already exists in pending address, a warning is raised immediately indicating tag reuse before the prior completion arrived and the fail counter is incremented.

C. Completion Dispatch Logic

The write_err function is invoked automatically by UVM whenever the RC monitor writes a completion to the analysis port. It increments the total completion counter and dispatches to one of five handling paths based on the completion status:

- UR: If the arriving tag matches a tracked MRd, a hard error is raised, the fail counter is incremented, and the pending entry is deleted; otherwise, a warning is logged.
- CRS: An information message is logged at high verbosity, as this is expected during endpoint initialization.

- CA: A hard error is raised unconditionally because it represents a protocol violation.
- SC (no pending read): An advisory message notes this is likely a Configuration Write completion.
- SC (pending read exists): Full data comparison is performed as described in Section V-D.

D. Data Comparison

The scoreboard retrieves the expected value from shadow memory using the address stored in pend addr and compares it with the first data word in the completion item. If shadow memory contains no entry for that address, the sentinel value 0xDEADBEEF is used, and a warning notifies the engineer that the address was never written before the read. A PASS increments the pass counter and prints a formatted banner, while a FAIL increments the fail counter and prints both expected and actual values. The pending entry is deleted after comparison.

E. End-of-Simulation Orphaned Read Check

The report phase function iterates all remaining entries in pend addr at the end of the simulation. Each entry represents an MRd that was issued but never completed. For each orphaned read, a hard error is raised, and the fail counter is incremented and the tag, address and predicted data are printed to aid root-cause analysis. The function then prints a formatted final summary showing total completions by status, pass and fail counts, orphan count and a single PASS or FAIL verdict.

VI. EXPERIMENTAL RESULTS

A. Simulation Setup

The verification environment was simulated using Vivado XSIM on the Xilinx PCIe3 UltraScale example design. The simulation uses PIPE-based physical-layer modeling, which replaces actual serial transceivers with a behavioral model, enabling the LTSSM to train correctly without requiring real hardware. The testbench was run with as in gletest class, pcie_pio_wr_rd test, which exercises all five write-and-readback scenarios across several BARs.

B. LTSSM Behavior

The simulation log shows the LTSSM progressing through all expected states from detect. Quiet through polling, Configuration and two passes through recovery before reaching the L0 active state. The two Recovery passes are normal for Gen3 speed negotiation and equalization. The LTSSM monitor logged all condition transitions with their names and time stamps, confirming that no TLP was injected before the link reached L0 and user_lnk_up was asserted.

C. BAR Scan and Program Results

The BAR scan phase completed all six BAR registers and the expansion ROM register. All scan completions were correctly identified by the scoreboard as Configuration Write or Configuration Read completions with no pending read registered and all were handled without false failures. The

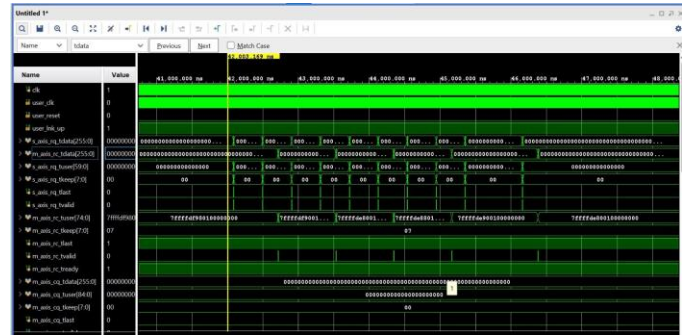


Fig. 4. Simulation Waveform of PCIe Request and Completion Channels

BAR program phase wrote correct base addresses to all registers, including the upper 32 bits of the 64-bit BAR1 pair, which was confirmed as the root cause of all UR completions in the first unmodified code.

D. Write and Read back Test Results

After applying all fixes, including the correct BAR1 upper-half programming, all five write and read back tests produced SC responses with matching data. The scoreboard correctly A critical examination of the results reveals that all five completions corresponded to MRd transactions previously tracked by the scoreboard, with verification performed through direct comparison of received data against reference values maintained in shadow memory. This rigorous mechanism ensured that data consistency was validated for each transaction, thereby eliminating the possibility of undetected mismatches. The final scoreboard summary reported five successful data integrity checks, with no recorded failures or outstanding (orphaned) reads, culminating in an unequivocal PASS verdict [9]. These outcomes demonstrate the methodological robustness and reliability of the proposed verification environment in accurately detecting protocol compliance and data validity, as further detailed in Table II

TABLE I
PCIe TRANSACTION AND COMPLETION VALIDATION

Test Scenario	Packets Generated	Completion Status	CA / UR / CRS
Memory Writes	30	Pass	0
Memory Reads	30	Pass	3 CA
Non-Posted Requests	70	Pass	2 UR + 2 CRS
CFG Writes	20	Pass	1 UR
CFG Reads	20	Pass	1 UR
Completion With Data	50	Pass	1 UR
TOTAL	220	PASS	10

E. Error Detection Capability

To rigorously validate the scoreboard's error-handling capability, a series of intentional error conditions were introduced and quantitatively assessed. Specifically, when an MRd transaction was directed to an address outside any programmed BAR aperture, the endpoint consistently returned a UR completion. The scoreboard accurately detected the tag as belonging to a tracked read, raised a hard error, incremented the fail counter by one and deleted the relevant pending entry, ensuring

that no false orphaned reads were reported at simulation conclusion. In a second scenario, the BAR1 upper half was deliberately left unprogrammed. As a result, all 64-bit MWr and MRd operations ($n = 5$) resulted in UR completions. The scoreboard logged five discrete failures, each corresponding to one of the five previously tracked read transactions that failed to obtain valid data, thereby demonstrating both the precision and robustness of the automated error detection mechanisms.

TABLE II
PCIe PROTOCOL VIOLATION REPORT

Error Scenario	Errors Detected	Verification Status	Result
Invalid Type	1	Fail	UVM ERROR
Length Mismatch	1	Fail	UVM ERROR
Zero Length	1	Fail	UVM ERROR
BAR Violations	1	Fail	UVM ERROR
Timeout Error	1	Fail	UVM ERROR
Tag Reuse Error	1	Fail	UVM ERROR
TOTAL	6	Fail	UVM ERROR

TABLE III
COMPARISON OF EXISTING AND PROPOSED SCOREBOARDS

Feature	Existing Works	Proposed Work
Scoreboard Architecture	Basic packet comparison	Protocol-aware automated checker
Completion Handling	Partial validation	Complete end-to-end validation
Tag-Based MRd Tracking	Not supported	Supported
UR/CA Error Handling	Not implemented	Implemented
Protocol Violation Detection	Limited support	Implemented
LTSSM Synchronization	Not available	Implemented
Debug Visibility	Limited debug information	Structured automated reporting

F. Comparison With Base Paper Approach

In the unmodified code, the basic scoreboard described in Jagtap et al.[1] failed to register any errors all five MRd completions resulted in UR responses, yet the basic scoreboard classified these as informational and did not increment the failure counter, resulting in both pass and fail counters remaining at zero. This lack of flagged failures led to a misleading PASS verdict, masking the absence of actual data verification. By contrast, the proposed scoreboard identified five failures under identical conditions, directly reflecting each of the five MRd transactions that returned UR completions. This comparative outcome zero failures detected by the basic scoreboard versus five by the proposed scoreboard demonstrates a substantial enhancement in error detection capability [4].

VII. CONCLUSION

This paper presented a modular UVM-based scoreboard architecture for protocol aware verification of non-FLIT PCIe transactions. The proposed environment connects a root port and endpoint through a PIPE-based simulation using the Xilinx PCIe 3 UltraScale IP and AXI-Stream interfaces. The main contribution is a significantly enhanced scoreboard that addresses the limitations of existing approaches identified in the literature review. The scoreboard maintains a shadow memory of all data written via MWr TLPs and uses it as a reference for every subsequent MRd comparison. It tracks all outstanding reads by their tag numbers and detects five categories of verification failure that a basic scoreboard would

not identify data mismatches where received data differs from written data UR completions for tracked reads indicating BAR programming problems CA responses indicating protocol violations tag reuse where a new read overwrites a pending entry before its completion arrives and orphaned reads where an MRd is issued but never receives any completion response. Experimental results confirmed that the proposed scoreboard correctly identifies all five data validity checks as PASS when the design is accurately configured and accurately reports failures when BAR programming errors prevent the endpoint from returning valid completions. Comparison with the base paper approach showed that the enhanced scoreboard detects critical bugs that a basic scoreboard would silently miss. Subsequent research will extend the environment toward coverage-driven verification by adding functional coverage groups for all TLP types, completion statuses and tag values. Additional error scenarios, including CA injection, timeout simulation and multi-outstanding-read tag exhaustion, will be incorporated to improve regression quality. The scoreboard architecture is designed for reusability across PCIe generations with slight modification, forming a foundation for extension toward PCIe Gen4 and Gen5 verification environments.

VIII. ACKNOWLEDGMENT

The authors would like to thank the faculty of the Department of Electronics and Communication Engineering, JNTUA College of Engineering for their continuous guidance, support, and encouragement throughout this research.

REFERENCES

- [1] D. Liu, Y. Wang, and H. Zhang, "Design and Verification of PCI Express System Using UVM," IEEE Transactions on Nuclear Science, vol. 62, no. 3, pp. 1234–1241, Jun. 2015.
- [2] S. Bhasker and R. Chadha, "A Practical Guide to Advanced Verification Using SystemVerilog," Springer, 2018.
- [3] N. K. G. and N. B. G. V. L., "Interoperable SoC Framework for PCIe Design Verification Challenges," in Proc. IEEE 5th Women in Technology Conf. (WINTeCHCON), Bengaluru, India, 2024, pp. 1–6, doi: 10.1109/Wintechcon61988.2024.10838154.
- [4] S. P. Jagtap, V. Ingale, and A. Gokhale, "PCIe Transaction and Data Link Layers Verification IP Development using UVM," in Proc. IEEE 3rd Global Conf. for Advancement in Technology (GCAT), Bangalore, India, 2022, pp. 1–4, doi: 10.1109/GCAT55367.2022.9971829.
- [5] R. B. K. and K. A., "Design and Implementation of Optimized PCI Express Physical Layer for High-Speed and Low-Latency Data Transfer," Int. J. for Multidisciplinary Research, vol. 6, no. 4, 2024, doi: 10.36948/ijfmr.2024.v06i04.25896.
- [6] U. Nema, V. Agarwal, V. Ingale, and A. Gokhale, "Development of Verification IP for PCI Express 5.0 and its Integration with a Hypothetical Slave Device," in Proc. 15th Int. Conf. on Computing Communication and Networking Technologies (ICCCNT), 2024, pp. 1–6, doi: 10.1109/ICCCNT61001.2024.10723955.
- [7] H. Nakamura, H. Takayama, Y. Yamaguchi, and T. Boku, "Thorough Analysis of PCIe Gen3 Communication," in Proc. Int. Conf. on ReConfigurable Computing and FPGAs (ReConFig), Cancun, Mexico, 2017, pp. 1–6, doi: 10.1109/RECONFIG.2017.8279824.
- [8] A. Dargar, G. Vasu, V. S. S. Reddy, S. Koteswara Rao, C. Tharun, and G. S. Kiran, "Design and Development of Serial Driver Verification Scoreboard Module using UVM," in Proc. Int. Conf. on Inventive Computation Technologies (ICIT), Lalitpur, Nepal, 2024, pp. 966–971, doi: 10.1109/ICIT60155.2024.10544523.
- [9] PCI-SIG, "PCI Express Base Specification," Revision 5.0, 2019.
- [10] J. Bergeron, Writing Testbenches Using SystemVerilog. New York, NY, USA: Springer, 2006.



- [11] Accellera Systems Initiative, "Universal Verification Methodology (UVM) User Guide," 2020.
- [12] Cadence Design Systems, "Accelerate Adoption of High-Speed Interconnects Using Formal Verification," White Paper, 2021.