

UVM-BASED SEQUENCE LIBRARY FOR NON-FLIT PCIE TRANSACTION LAYER PACKET GENERATION AND VERIFICATION

Matlavari Bindu Sree¹, C. Subhas²

¹ M.Tech Scholar, Department of Electronics and Communication Engineering, JNTUA College of Engineering, Anantapur, Andhra Pradesh, India
² Adjunct Professor, Department of Electronics and Communication Engineering, JNTUA College of Engineering, Anantapur, Andhra Pradesh, India
India\matlavarihindusree@gmail.com

Abstract—This paper presents a reusable Universal Verification Methodology (UVM) sequence library for generating and checking Non-FLIT Peripheral Component Interconnect Express (PCIe) Transaction Layer Packets (TLPs). The library organizes stimulus by transaction class, including Memory Read, Memory Write, Configuration Read, Configuration Write, Message, and Completion operations. Transaction items are passed through a UVM sequencer and driver to an AMD PCIe endpoint simulation environment. Monitors and scoreboard checks are used to observe packet fields, data transfers, tag handling, and request-to-completion relationships. The supplied project presentation records completed sequence development and simulation waveform captures for requester and completion activity. Reported directed scenarios include programmed I/O (PIO) memory write/read and configuration-space transactions; the source material describes successful data retrieval and no protocol violations in the executed scenarios, but does not provide numerical coverage or performance measurements. The framework is intended as a foundation for broader constrained-random verification and future PCIe-over-Universal Chiplet Interconnect Express (UCIe) integration.

Keywords—PCIe, Non-FLIT, Transaction Layer Packet, UVM, SystemVerilog, sequence library, scoreboard, UCIe, verification.

I. INTRODUCTION

High-speed computing platforms rely on reliable interconnects to move data between processors, accelerators, storage controllers, and peripheral devices. PCI Express (PCIe) provides a scalable packet-based interface. Its transaction layer represents memory accesses, configuration operations, messages, and responses as Transaction Layer Packets (TLPs). A verification environment must therefore check both packet construction and the relationship between requests and responses.

Directed tests are useful for known scenarios, but the combinations of addresses, lengths, tags, byte enables, payloads, and completion outcomes make repeated hand-written stimulus expensive. UVM supports reusable stimulus by separating sequence generation, driving, monitoring, and checking. This work develops a Non-FLIT PCIe TLP sequence library that can serve as a configurable traffic source for a PCIe endpoint and

as a future protocol-traffic source in a PCIe-over-UCIe environment.

The main contributions are: (1) transaction-specific sequence classes for the major TLP categories represented in the project; (2) constrained packet-field generation and handling of 4-KB address boundaries; (3) a monitor/scoreboard flow for request observation and tag-based completion correlation; and (4) simulation evidence drawn from the supplied project presentation.

II. BACKGROUND AND RELATED WORK

PCIe uses a layered architecture. The Transaction Layer forms and interprets TLPs, the Data Link Layer provides link-level reliability and error detection, and the Physical Layer handles signaling and serialization. The present implementation concentrates on Non-FLIT transaction-level traffic rather than FLIT-mode packet representation.

Transactions are grouped as Posted, Non-Posted, and Completion traffic. Memory Write is typically Posted and does not require a completion. Memory Read is Non-Posted and expects a completion carrying the requested data. Configuration accesses and messages have transaction-specific response behavior. A request commonly includes requester identity, tag, traffic class, length, address, and byte-enable fields; completion checking additionally considers completer identity, status, and request correlation.

Prior work has studied PCIe transaction-layer simulation and formal modeling, while UVM provides a reusable verification architecture. The project presentation notes that commercial verification IP is a common existing approach and motivates a sequence library to improve stimulus reuse and configurability. This paper focuses on the implementation described by the supplied paper draft and review presentation; it does not claim comparative performance against commercial VIP.

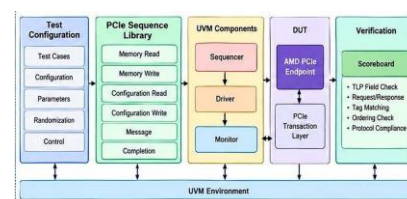


Fig. 1. UVM-based architecture for Non-FLIT PCIe TLP generation and checking.

III. OBJECTIVES AND PROPOSED FRAMEWORK

The project objectives are to create reusable

sequences for the principal TLP types, generate configurable traffic, check packet correctness and protocol-oriented relationships, and prepare for integration with a UCIE verification environment. The framework comprises a test, transaction-specific sequence, sequence item, sequencer, driver, endpoint (design under test, DUT), monitor, and scoreboard.

A test selects a scenario and starts the corresponding sequence. The sequence builds a transaction item and applies field constraints before sending it to the sequencer. The driver converts the item into interface activity for the endpoint. Monitors capture the observed traffic, and the scoreboard checks fields and request-response relationships. Keeping checking separate from stimulus enables the same observation logic to be reused across sequence classes.

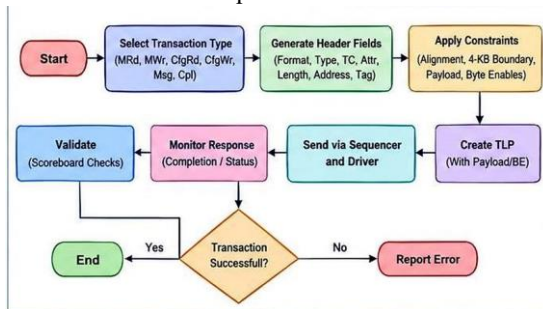


Fig. 2. Generation and validation flow for the sequence library.

IV. TLP GENERATION AND CONSTRAINTS

The sequence library covers Memory Read (MRd), Memory Write (MWr), Configuration Read (CfgRd), Configuration Write (CfgWr), Message, and Completion (Cpl/CplD) transactions. Fields are randomized according to transaction-specific rules. Data payloads are generated where the selected operation carries data; request identity and tag fields are retained for response correlation.

For a conceptual packet model, total packet size can be expressed as the header contribution plus any payload contribution:

$$TLP\ size = header\ size + payload\ size$$

For memory transactions, the project model relates the Length field to data quantity as Bytes transferred = Length $\times$ 4. The exact interpretation of Length and payload boundaries must follow the applicable PCIe specification and the endpoint configuration.

The model checks that a request does not cross a 4- KB boundary. The stated constraint is (Address mod 4096) + PayloadBytes $\leq$ 4096. If a requested transfer would cross this boundary, the sequence-generation procedure divides it into valid portions. The constraint is intended to prevent a generated request from violating the boundary rule.

For Non-Posted requests, completion responses are matched to outstanding requests using identifying

fields, including the tag. The project draft lists Successful Completion (SC), Unsupported Request (UR), Configuration Retry Status (CRS), and Completer Abort (CA) as completion-status cases considered by its checking model. These checks are intended to detect mismatched responses and unexpected completion outcomes.

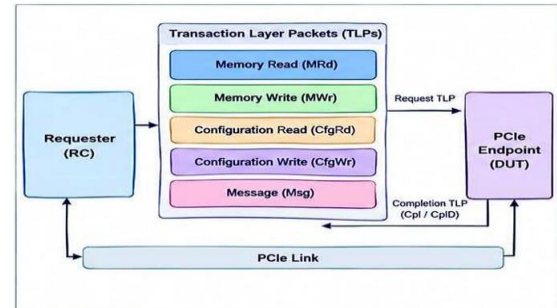


Fig. 3. Transaction categories and expected response behavior represented in the project.

V. VERIFICATION SETUP AND TEST STRATEGY

The supplied presentation describes a UVM-based Root Complex testbench and a simulated PCIe endpoint, with requester request (RQ) and requester completion (RC) activity observed through the verification environment. A programmed I/O (PIO) design is used for memory access. The paper draft further identifies an AMD PCIe endpoint environment as the target for generated requests.

The directed strategy includes: (1) PIO memory write followed by memory read; (2) configuration write and read; (3) observation of request descriptors and completion descriptors; and (4) checking relevant packet fields and request-completion correlation. Memory Write exercises posted transfer behavior. Memory Read exercises the non-posted request path and its completion. Configuration transactions exercise register-space access, including registers identified in the project material such as Vendor ID, Device ID, Command Register, and Base Address Registers.

The verification checks described in the source material include requester/completer identifiers, tags, traffic class, length, address, completion status, data values, and transaction ordering. Functional coverage, broad constrained-random regression, systematic error injection, and coverage closure are future extensions rather than measured achievements in the current results.

V-A. Transaction Fields and Request-Response Checking

The packet model treats header construction and payload generation as transaction-dependent operations. For memory requests, the sequence selects an address, length, byte enables, requester information and tag, then generates payload data when the operation carries data. Read requests instead describe the requested transfer and rely on a completion path to return the result. The checking layer should compare observed fields with the transaction item and retain

enough request context to match a completion to the correct outstanding operation. These checks are distinct from link-layer reliability checks and are scoped to the transaction-level behavior exercised by the supplied project.

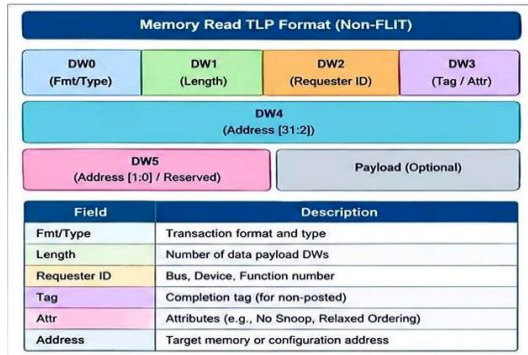


Fig. 7. Source-paper illustration of the Non-FLIT memory TLP format and fields.

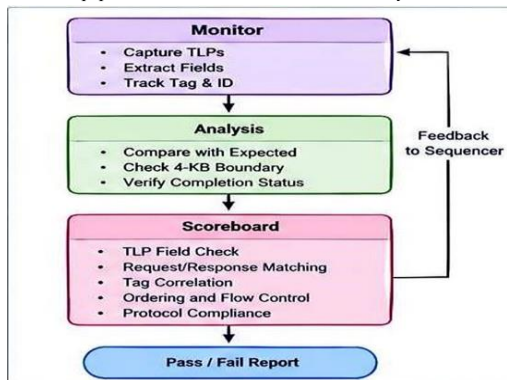


Fig. 8. Source-paper illustration of monitor, analysis and scoreboard checking.

VI. RESULTS AND WAVEFORM ANALYSIS

A. PIO memory write and read

The source paper reports execution of a directed `pcie_pio_wr_rd_test`. The sequence issues write requests with addresses and payload values, then reads the previously accessed locations and compares returned data against expected values. The reported outcome is successful data transfer and retrieval for the executed scenario. Monitor and scoreboard checks are described as examining address decoding, tag handling, ordering, and request-completion relationships.

B. Configuration transactions

Configuration Read and Configuration Write sequences were exercised for configuration-space access. The project paper reports correctly formatted requests and completion handling for the executed operations. The presentation's requester and completion descriptor slides illustrate the packet- descriptor fields used to construct and observe these transactions.

C. Simulation waveform evidence

Three supplied simulation captures are included below: the `pcie_pio_wr_rd_test` waveform, Configuration Write/Read activity, and Memory Write/Read activity. They provide qualitative evidence

of signal activity; waveform screenshots alone do not establish numeric functional coverage.

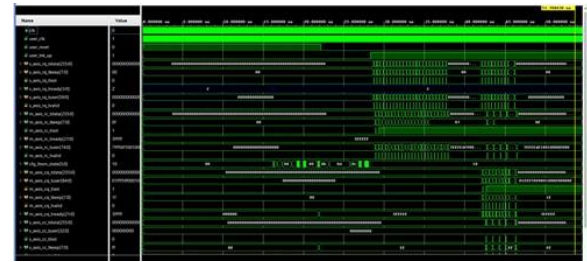


Fig. 4. Simulation waveform for `pcie_pio_wr_rd_test` (PIO memory write and read).

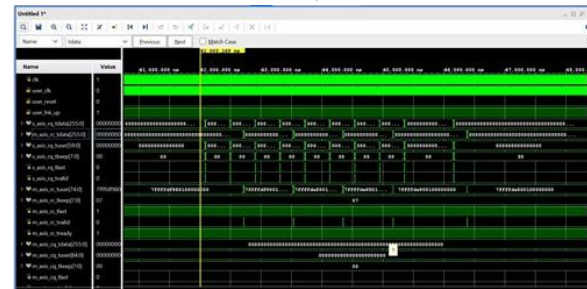


Fig. 5. Simulation waveform for Configuration Write and Configuration Read transactions.

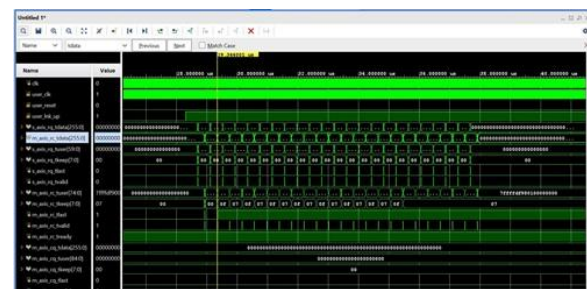


Fig. 6. Simulation waveform for Memory Write and Memory Read transactions.

D. Summary of reported results

Across the executed scenarios, the source paper reports that the generated traffic met the alignment and formatting constraints used by the verification model and that no protocol violations were observed in those scenarios. The results support the feasibility of the reusable sequence-library approach. They should not be interpreted as exhaustive protocol compliance or full functional coverage, because the project material does not provide total test counts, coverage percentages, or regression statistics.

VII. DISCUSSION

The main benefit of the framework is reuse: once a transaction class is represented by a sequence, tests can vary legal field values and transaction ordering without rebuilding the complete verification infrastructure. Constraints place important legality checks close to packet creation, while monitoring and scoreboard logic provide a separate checking layer.

The current evidence is directed and simulation- based. Stronger claims would require a documented regression plan, explicit pass/fail logs, coverage reports, and negative tests. The presentation identifies error injection, constrained-random testing, and integration into a UCle verification framework as future goals. Such extensions could test robustness and

enable PCIe traffic to be exercised across a die-to-die interconnect stack.

VIII. CONCLUSION AND FUTURE WORK

This paper has presented a UVM-based sequence library for Non-FLIT PCIe TLP generation and transaction-level checking. The library represents Memory Read, Memory Write, Configuration Read, Configuration Write, Message, and Completion traffic. The described environment drives generated requests to a PCIe endpoint and observes request and completion activity using monitors and scoreboard checks.

The supplied project results report successful execution of PIO memory write/read and configuration transaction scenarios, including data retrieval and tag-based completion correlation, with no protocol violations reported in the executed cases. Future work includes broader constrained-random regressions, functional coverage, negative testing, error injection, timeout checks, regression automation, and integration of PCIe traffic into a UCIe verification environment.

VI-A. SEQUENCE ITEM DESIGN AND REUSABILITY

The sequence item acts as the transaction-level contract between stimulus generation and the driver. In the project model, a transaction item carries the selected operation, format and type information, traffic class, attributes, transfer length, address, requester identity, tag, byte enables and any applicable payload. Keeping these fields together makes a generated request inspectable before it reaches the interface and allows the same driver path to support multiple transaction classes.

Transaction-specific sequences are preferable to duplicating complete tests for each operation. A Memory Write sequence supplies write data and byte-enable information, whereas a Memory Read sequence describes the requested transfer and preserves the identifying information needed to correlate the later completion. Configuration sequences use configuration-space addressing and their applicable data or status behavior. Message and completion sequences are modeled separately because their fields and expected protocol behavior differ from ordinary memory requests.

Randomization should be constrained by transaction type rather than treating every field as independent. Fields that are not meaningful for a selected operation should be constrained according to the packet model. Payload generation should be enabled only for operations that carry payload data. This reduces illegal combinations and makes failures easier to localize because the generated item can be inspected against the selected transaction class.

VI-B. ADDRESS, LENGTH, AND BYTE-ENABLE CHECKS

Address legality is important because a syntactically valid header can still describe an invalid transfer. The

model considers starting address, transfer length, payload size and byte enables together. The project uses the 4-KB boundary condition ($\text{Address mod } 4096$)

+ $\text{PayloadBytes} \leq 4096$ to prevent a generated request from crossing a 4-KB boundary. When a requested transfer exceeds the allowed region, the generation strategy can divide it into smaller legal requests, provided the scenario supports splitting.

For memory transactions, the model relates length in double words to payload size using $\text{PayloadBytes} = \text{Length} \times 4$. This calculation must be interpreted alongside applicable PCIe rules for zero-length encodings, first and last double-word byte enables, and endpoint-specific limits. The sequence item should not assume that header length alone fully describes meaningful bytes when byte enables mask part of the first or final double word.

Useful directed corner cases include aligned and unaligned addresses, transfers ending exactly at a 4-KB boundary, transfers that would cross the boundary, single-DW transfers, multiple-DW transfers and different byte-enable patterns. The supplied project material describes boundary checking and directed memory/configuration activity, but does not provide a complete coverage report for these cases. They are recommended regression targets rather than claimed completed coverage.

VI-C. MONITOR, ANALYSIS, AND SCOREBOARD OPERATION

The monitor observes the interface independently of the sequence that created the request. It reconstructs transaction-level information from observed activity and publishes it to the analysis path. This separation is valuable because the checker evaluates what the DUT and interface actually produced, rather than relying only on the values that the sequence intended to send. It also allows the monitor to be reused with different stimulus sequences.

The scoreboard maintains context required to compare observed traffic with expected behavior. For a non-posted request, the checker can retain requester identity, tag, address and expected transfer information until a corresponding completion is observed. The response is checked for correlation and status, and completion data is compared with expected data when the transaction returns data. Unexpected tags, missing responses, mismatched fields or inappropriate completion status should be reported with enough context to identify the originating request.

Ordering checks must account for transaction attributes and the ordering rules applicable to the scenario. Request/response matching, tag correlation, ordering and protocol compliance are distinct checks: a correct data value does not prove correct tag matching, and a matching tag does not establish that every packet field is legal. Keeping checks separate improves debug quality and makes failures more actionable.

VI-D. TEST SCENARIOS AND DEBUG METHODOLOGY

The directed test strategy begins with known operations and expected outcomes. In a PIO memory write/read test, the test writes a chosen value to an address and then issues a read to verify that the same value is returned. The write exercises posted request generation and payload handling; the read exercises the non-posted request path and completion-data path. Configuration read/write scenarios check configuration-space request formatting and corresponding completion behavior.

During debug, the first step is to compare the sequence item with the request observed by the monitor. If the fields differ, the issue may be in item construction, driver mapping or interface timing. If the request is correct but the response is unexpected, analysis should focus on endpoint behavior, completion status, tag association and returned data. If both packets appear correct but the test fails, expected-value bookkeeping or scoreboard correlation should be inspected. This staged method reduces time spent searching unrelated components.

The supplied project evidence reports successful data retrieval for the executed PIO scenario and correct handling of executed configuration transactions, with no protocol violations reported in those cases. Waveform captures provide qualitative evidence of request and response activity. They do not demonstrate exhaustive compliance or quantify functional coverage; those conclusions require test logs, coverage data and a documented regression.

VI-E. EXTENSION TO CONSTRAINED-RANDOM AND UCle VERIFICATION

A next stage is to combine directed sequences with constrained-random generation. Directed tests protect important known use cases, while constrained-random tests explore combinations of addresses, lengths, tags, attributes, traffic classes and payload values that may not be anticipated manually. Random seeds and configuration settings should be recorded so a failing transaction can be reproduced. Functional coverage should track transaction class, payload presence, length ranges, address alignment, byte-enable patterns, completion status and request/response matching outcomes.

Negative testing can strengthen the environment by introducing malformed or unsupported conditions in a controlled manner, such as unexpected completion tags, incorrect response status, timeout conditions or mismatched returned data. These tests should be separated from legal traffic tests, and expected error handling should be specified before they are added to regression. Automated summaries should record pass/fail counts, failure categories, seeds and coverage closure status.

For future PCIe-over-UCle integration, the sequence library can remain at transaction level while a transport

adaptation layer maps transactions into the selected interconnect environment. The integration should not assume PCIe TLP fields and UCle transport framing are identical. Interface mapping, encapsulation, timing, reset behavior and error propagation need dedicated validation. This is proposed future work; the present paper reports the Non-FLIT PCIe sequence-library work and the directed simulation scenarios documented in the project material.

IX. LIMITATIONS AND REPRODUCIBILITY

The supplied evidence consists of directed simulation scenarios and waveform captures. The paper does not provide a numerical functional-coverage percentage, regression pass count, timing measurements or a complete reproducibility package; such values are therefore not inferred here. The screenshots are useful for showing signal activity, but final protocol compliance should be established from the testbench checks, logs and applicable PCIe specification. Future evaluation should report coverage bins, test counts, seed management, failure classification and repeatable regression commands.

TABLE I. TRANSACTION COVERAGE SUMMARY

Operation / class	Request payload	Expected response / behavior
Memory Write (MW) — Posted	Write data present	No completion required
Memory Read (MRd) — Non-Posted	No request data payload	Completion with read data (CplD)
Configuration Write (CfGWr) — Non-Posted	Configuration write data, as applicable	Completion status
Configuration Read (CfGRd) — Non-Posted	No request data payload	Completion data or status
Message	Depends on message type	Message-specific behavior; do not assume a completion
Completion (Cpl/CplD)	Response packet, not a request class; CplD may carry data	Correlates with an outstanding Non-Posted request

REFERENCES

- [1] L. Wang, "Design and Simulation of PCI Express Transaction Layer," in Proc. International Conference on Computational Intelligence and Software Engineering (CISE), 2009, doi: 10.1109/CISE.2009.5363534.
- [2] V. Sudheer Raja, M. V. Raghavendra, and G. Subbarao, "Simulation of PCI Express Transaction Layer using Hardware Description Language," International Journal of Informatics and Communication Technology, vol. 4, no. 1, pp. 7–12, 2015.
- [3] P. Bohm, "Incremental Modelling and Verification of the PCI Express Transaction Layer," in Proc. 7th IEEE/ACM International Conference on Formal Methods and Models for Co-Design (MEMOCODE), pp. 36–45, 2009, doi: 10.1109/MEMCOD.2009.5185376.
- [4] PCI-SIG, PCI Express Base Specification, Revision 5.0, 2019.
- [5] UCle Consortium, Universal Chiplet Interconnect Express (UCle) Specification, Versions



1.0/1.1, 2022–2023.

[6] Accellera Systems Initiative, Universal Verification Methodology (UVM) 1.2 User Guide, 2015.