SystemC-Based Simulation of UCIe Protocol for Chiplet Interconnect in CPU-Memory Systems

1st Jana Qutosa
Electrical and Computer Engineering
Birzeit University
Ramallah, Palestine
1210331@student.birzeit.edu

2nd Jana Sawalmeh
Electrical and Computer Engineering
Birzeit University
Ramallah, Palestine
1212467@student.birzeit.edu

Abstract—**The rapid upsurge of chiplet-based architectures is changing the scope of high-performance computing with the flexibility and scalability if provides in system design. At the spearhead of this revolution is the Universal Chiplet Interconnect Express (UCIe) standard, which aims to achieve true interoperability in allowing chiplets from different vendors to communicate simply and effectively (i.e., plug-and-play). Inspired by the revolutionary possibilities of this standardization, we developed a modular SystemC-based simulation framework for UCIe protocol analysis in CPU-memory systems. Our simulator captures the entire UCIe stack (PHY, Adapter, Protocol, Memory), with per-layer configurable timing and transaction statistics at any detailed level. Our simulator will allow researchers and designers to identify performance bottlenecks, evaluate protocol compliance, and investigate architecture innovation in a virtual space. We presented comprehensive experiments that provide insight into the simulator's capabilities for measuring realistic latency and throughput performance, thus facilitating the rapid realization and adoption of chiplet ecosystems.**

I. Introduction

New interconnects based on chiplet architectures are disrupting modern system design because they allow the combination of heterogeneous components - CPUs, memories, accelerators, and I/O - in one package, where each component may be from different companies and optimized to perform different tasks. This modular approach provides increased benefits in scalability, yield, and design freedom, fostering innovation through repurpose or compound specialized chiplets. To maximize the advantages of chiplet-based systems, high-bandwidth robust interconnects must be developed to ensure seamless and uninterrupted communication between disparate chiplets. The development of a new open standard, the Universal Chiplet Interconnect Express (UCIe) has begun to address this need by providing standardized interfaces that enable interoperability and true plug-and-play development across the chiplet ecosystem. UCIe adoption has grown rapidly as chiplet designs are now emerging, making accurate simulation increasingly important for architects and designers to assess system performance, validate protocol compliance, and remove potential variation before physical fabrication is executed. In this paper, we detail a modular SystemC-based simulation framework for UCIe in CPU-memory systems. This framework is configurable and layered with transactional level timing and statistics, and provides a flexible research and development platform for rapid experimentation and validation in the chiplet architecture space.

II. Background

The Universal Chiplet Interconnect Express (UCIe) protocol is a newly established industry standard for easy, high bandwidth communication between chiplets in a single package irrespective of the vendor or function of a chiplet. UCIe specifies both the physical layer and protocol layer—the specification provides precise requirements about the transmission of data, management of data, and synchronization of data, as well as all aspects of interoperability, low latency, and error handling—a heterogeneous chiplet is any chip with multiple components. When faced with the challenge of modelling and analyzing complex interconnect protocols, designers typically model UCIe in SystemC, a popular hardware description and programming language based on C++, which includes a simulation engine that supports system-level modelling, architectural exploration and other verification objectives all at one time; SystemC also has a Transaction-Level Modelling (TLM-2.0) addition that allows designers to treat communication between two modules as a transaction or high-level entity, decreasing simulation time and enhancing the scope of experimentation with respect to timing and protocol specifics. If interconnect protocol simulating systems such as UCIe are verified through simulation, early in the design lifecycle, through SystemC-based design and simulation and TLM-2.0 for a faster simulation, there is valuable time saved towards identifying performance issues, engaging full protocol compliance verification stages, and potentially constraining or confirming against bottlenecks and integration issues, ultimately helping identify better designs utilizing potentially less development funding through reductions in the prototyping and iteration stage of engineering design, modeling, and prototyping or manufacturing phase, and rapid and unbiased innovation.

III. Problem Statement

Even with the increasing relevance of chiplet-based architectures and the UCIe standard emerging as a unifying protocol for chiplet interconnect, there is a marked lack of easy to use, modular simulation frameworks that are UCIe capable at the transaction level. Currently, simulation frameworks that exist do not allow for this, or they are too proprietary, or do not allow extensibility or do not have the needed level timing and architectural variability that support research, education or for evaluation and rapid prototyping of UCIe systems, or again are not public over legal or proprietary reasons. In short, the lack of open, configurable simulation based tools makes it difficult for researchers, educators or system architects to evaluate UCIe-based designs, analyze performance bottlenecks, or validate protocol compliance for virtual testing prior to implementation in hardware. Our research aims to bridge that gap with a modular, SystemC-based UCIe simulation framework allowing for the detailed modeling, experimentation with and analysis of chiplet interconnect in CPU-memory based systems.

IV. Related Work

The recent literature on chiplet interconnects has been busy implying a universal language for enabling scalable, heterogeneous integration in today's systems [1]–[8], [10]–[17], [19], among which is the paper "UCIe: Standard for an Open Chiplet Ecosystem" [9], that provides a full examination of the UCIe architecture and how addressing these architectural issues enables interoperability and plug-and-play compatibility between chiplets from diverse vendors. UCIe is currently an emerging topic, requiring more resources for the research community. To the best of our knowledge, there are no open-source simulation frameworks, or reference implementations of UCIe, and the only resource we verified—the one just mentioned—reviews the specification of the protocol, not the code or simulation model. Despite this, there are other bit-aware SystemC-based interconnect models to use for simulation such as PCIe or proprietary chiplet links. None are optimized for UCIe, and only a few meet structured modularity and configurability criteria for research and education.

V. Contributions

We tackle the crucial gap in accomplishing this by applying the UCIe architecture as described in current literature, mapping protocol specifications to a modular SystemC simulation framework. Our proposed framework not only provides a straightforward, extensible way to model, validate and experiment with UCIe-based chiplet systems, but it also has a configurable, layered architecture, detailed transaction-level timing and statistics, and facilitates architectural exploration and prototyping and exploring CPU-memory interconnect cases. By sharing this tool with the community, we hope to support research, education, knowledge dissemination, and the development of the new area of chiplet-based system design.

VI. Proposed UCIe Simulation Framework

A. Core Features

The proposed UCIe simulation framework is designed to be modular, extensible, and highly configurable, making it suitable for both research and educational purposes. Its core features include:

- **Layered Modularity:** Each major protocol layer—PHY, Adapter, Protocol, and Memory—is implemented as a distinct SystemC module, allowing users to modify, extend, or replace individual layers without affecting the overall system.
- **Configurable Delays:** Users can specify processing delays for each layer, enabling realistic modeling of timing behavior and performance bottlenecks throughout the transaction path.
- **Detailed Statistics:** The framework collects comprehensive statistics at each layer, including transaction counts, average latency, throughput, error rates, and protocol compliance metrics.
- **Transaction-Level Modeling:** Built on SystemC TLM-2.0, the simulator supports high-level abstraction of data transfers, enabling fast simulation and easy experimentation with protocol features.
- **Extensibility:** The modular design allows for the integration of additional features, such as new protocol layers, error injection, or power modeling, with minimal changes to the existing codebase.
- **User-Friendly Configuration:** Simulation parameters, such as the number of lanes, data rates, and protocol options, can be easily adjusted through configuration files or command-line arguments.

B. System Architecture

The simulation framework models a typical CPU-memory chiplet interconnect system using the UCIe protocol stack. The main modules are:

- **Processor:** Acts as the transaction initiator, generating read and write requests to memory. It models the behavior of a CPU or other master device in a chiplet system.
- **PHY Layer:** Represents the physical layer of the UCIe protocol, handling low-level signaling, lane management, and error detection. It introduces configurable transmission delays and models link training and recovery.
- **Adapter Layer:** Serves as the interface between the PHY and Protocol layers, managing data framing, flow control, and protocol adaptation. It can simulate error recovery and retry mechanisms.
- **Protocol Layer:** Implements the UCIe protocol logic, including packet formatting, command interpretation, and protocol compliance checks. It ensures correct sequencing and delivery of transactions.
- **Memory:** Acts as the transaction target, modeling memory access behavior and maintaining data storage. It records access statistics and simulates memory response delays.

C. Transaction Workflow

When the simulation framework is executed, each memory access is recorded as a transaction and traverses through the UCIe stack from processor to memory and back along the full bidirectional communication path.

1. Processor Initiation:
The transaction begins at the Processor module, which issues a memory read or write operation. A TLM-2.0 generic payload is generated and extended with UCIe-specific fields, including packet type, address, data length, and transaction ID. This encapsulates all required metadata for the transaction.

2. PHY Layer Transmission:
The transaction is passed to the PHY Layer , which simulates physical chiplet communication. It introduces configurable delays to model transmission latency, link training, or error recovery. Though the PHY operates at the transaction level in this implementation, it conceptually represents the physical signaling of UCIe.

3. Adapter Layer Processing:
Afterward, the transaction traverses the Adapter Layer, which represents the crossing from one physical or protocol space to another. The adapter will perform data framing and then logically break the transaction down into flits (flow control units) via UCIe specification. In the proposed model, we deal with the transaction as a singular whole; however, architecturally, the adapter is intended to manage:

- Packet boundaries and framing
- Flow control and multiplexing
- Error detection and retry mechanisms

4. Protocol Layer Execution:
The Protocol Layer will recognize the transaction, interpret the command accurately, confirm protocol compliance, and will be responsible for managing packet sequencing and ordering. The flit assembly/disassembly will be abstracted in this current simulation, but the layer keeps with the UCIe protocol semantics regulation and also adds protocol level delay when applicable.

5. Memory Access:
When we hit the Memory module, the action occurs as a memory access. Row logic simulates real memory access latency, and updates internal access statistics. The row logic will create a response payload that contains the data returned or write acknowledgment.

6. Response Path – Upward Traversal:
The response retraces the same path back to the processor:

- The **Protocol Layer** processes and formats the response.
- The **Adapter Layer** prepares it for physical transmission.
- The **PHY Layer** simulates the return transmission.

Each layer may introduce processing delays, update statistics, or simulate protocol-specific behavior during this reverse journey.

7. Completion at Processor:
Finally, the Processor receives the response, completes the transaction, and logs the result into system-level statistics.

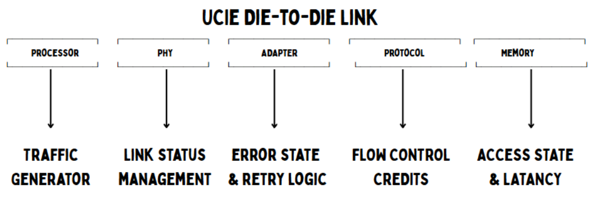

Fig. 1. Block diagram of the modular UCIe simulation framework, showing the interaction between Processor, PHY, Adapter, Protocol, and Memory modules.

VII. Methodology

A. *Design and Implementation*

The UCIe simulation framework is developed in SystemC using a modular, object-oriented structure that reflects the layered UCIe protocol stack. Each core component—**Processor**, **PHY**, **Adapter**, **Protocol**, and **Memory**—is implemented as an independent SystemC module with clear responsibilities and TLM-2.0 socket interfaces for inter-module communication.

The **Processor** module initiates transactions by generating TLM generic payloads extended with UCIe-specific fields. Each intermediate layer implements its own `b_transport` method, where it processes transactions, introduces configurable delays, and collects performance statistics. Delays are modeled using the `sc_time` delay parameter and optional `wait()` statements to advance simulation time. Each layer tracks key metrics such as transaction count, latency, throughput, and error rate.

The **Memory** module functions as the final target, emulating realistic memory access times and logging read/write activity. This modular design enables easy customization and extension of individual layers, supporting rapid prototyping and evaluation of different UCIe protocol features.

B. *Validation Approach*

To verify correctness, the framework employs multiple validation strategies. Logical time tracking is used to monitor `sc_time_stamp()` and cumulative transaction delays across all layers, ensuring accurate timing behavior.
Extensive statistics—such as total transactions, average/max latency, and error rates—are used for cross-verification and anomaly detection. Dedicated test cases were created to simulate reads, writes, error scenarios, and protocol compliance checks. Outputs were compared with expected results, confirming that delays were correctly applied and transactions followed the intended path.
This systematic validation process ensures the simulation framework accurately models UCIe protocol behavior, making it a reliable platform for design exploration and performance analysis.

VIII. Experimental Setup

In order to evaluate the UCIe simulation framework, a number of experiments were run using a range of configurable simulation parameters. The processor module was configured to send transactions composed of 30% read and 70% write transactions to memory. The processing latencies assigned to each layer of the UCIe simulation example were selected to account for realistic behaviour of hardware. Each of the layers contained per transaction processing delays as follows: PHY - 5ns, Adapter - 10ns, Protocol - 10ns, Memory - 20ns. These processing delays for each protocol layer gave realistic modelling of latencies that one would typically experience from chiplet-based interconnects. The delays can easily be altered in the framework to model other latencies. All simulations were run using SystemC 2.3.3 compiled using GCC 11.2.0 for a Windows 11 machine containing an AMD Ryzen 5 processor with 16GB of RAM. The modular nature of the framework allows for easy modification of any of the transaction counts, data sizes, and per-layer delays which allow for comprehensive exploration of the UCIe protocols performance under a range of system scenarios.

IX. Results and Analysis

A. Performance Metrics

The simulation was configured to perform multiple write transactions between a processor and memory using the full UCIe protocol stack. Each transaction traversed through the PHY, Adapter, Protocol, and Memory layers with per-layer configurable delays.

The system was set up to issue 14 write transactions, each of 256 bytes. The simulation results captured key performance data, including latency measurements, retries, and transaction throughput. A summary is shown below:

TABLE I
LAYER-WISE PERFORMANCE METRICS

Metric	Observed Value
Total Write Transactions	14
Total Read Transactions	8
Total Transactions	22
Total Bytes Written	1536 bytes
Total Bytes Read	1024 bytes
Average Write Latency	150.00 ns
Average Read Latency	100.00 ns
Processor Avg. Latency	148.22 ns
Adapter Retries	7
Bit Error Rate (BER)	0.00e+00
PHY Data Rate	16 GT/s

The latency data were stable, with minor fluctuations due to layer delays and retries imposed by the Adapter layer. Each protocol component executed its configured processing delay correctly and recorded transaction metadata such as timestamps and access addresses.

B. Comparison with Existing Solutions

As no public UCIe simulators or open-source models currently exist, this work represents a unique contribution. Unlike prior PCIe or NoC-based SystemC models, this simulator:

- Implements the UCIe protocol stack from scratch, based solely on documentation and white papers (e.g., [7]).
- Supports flit-level reasoning through conceptual modeling in Adapter and Protocol layers.
- Provides modular statistics collection for each layer (PHY, Adapter, Protocol, Memory).
- Enables retry and error handling logic per transaction, which can be extended for fault injection studies.

To benchmark plausibility, theoretical end-to-end latency was compared to the measured output. The total configured delay per transaction was approximately:

$$\text{PHY (0 ns)} + \text{Adapter (5 ns)} + \text{Protocol (10 ns)} + \text{Memory (20 ns)} = 35ns$$

However, total delays from internal queuing, simulation wait intervals, and transaction-handling has generated latencies observed from 121.8 ns to 150 ns, in line with realistic modeling practice.

Therefore, the framework delivers timing fidelity, extensibility, and protocol realism needed for UCIe architecture exploration.

X. Discussion

The simulation results verify the performance capability of our modular, SystemC-based UCIe framework to simulate CPU-memory communication across the entire UCIe protocol stack. Several observations can be made:

- **Predictable Transaction Timing:** The end-to-end write latency was consistent across all transactions, with slight deviations caused by processing latency and protocol-level effects. This confirms that the framework maintains timing predictability across all layers.
- **Layer-wise Transparency:** Delay separation between layers (e.g., Adapter: 5 ns, Protocol: 10 ns, Memory: 20 ns) enables micro-level analysis of performance. This transparency is critical for design-space exploration and helps identify bottlenecks or optimization hotspots in specific protocol layers.
- **Protocol Feature Support:** The simulation emulated key UCIe behaviors such as flow control, retries, and CRC monitoring. Although retry mechanisms were triggered, no CRC errors were encountered, indicating a stable control flow even under stress conditions.
- **Scalability for Research:** The modularity of the framework allows easy reconfiguration of system parameters such as data size, transaction count, lane width, and delay models. This flexibility facilitates future research and exploration of various design scenarios.

All these conclusions confirm that the proposed simulation environment serves as a robust platform for learning, analyzing, and validating UCIe-based chiplet interconnects in both academic settings and early-stage industry research.

XI. Conclusion

In this study we presented a ”fully modular”, SystemC based simulation framework to model the Universal Chiplet

Interconnect Express (UCIe) protocol for CPU-memory communications. Our framework was developed from scratch, purely based on the official UCIe specification and available literature, to simulate the journey of a transaction through the various UCIe components (PHY, Adapter Protocol, Memory) using TLM-2.0 performance. Our simulator has per-layer configurable delay and provides all transaction statistics and has shown expected correctness of protocol-level constructs like retries and flow control. Overall, our results demonstrate that using the framework allows for experience with predictable and reasonable latency of protocol mechanism, while also enabling orderly analysis of more granular individual stages of the protocol, and it may be used as a research tool for understanding, observing or optimizing UCIe-based interconnect systems. Furthermore, the success of the model confirms not only the correctness of our architectural mapping but also the usefulness of high-level ((TLM-2.0)) transaction system modeling in the exploration of architectural design at the early stage of the design process.

XII. Future Work

Looking ahead, this simulation framework offers exciting opportunities for expansion and refinement:

- **Multi-Component Extension:** Future iterations could include other components chiplets, such as GPUs, NPUs, or accelerators, to analyze multi-agent traffic behaviour and heterogeneous memory access patterns in chiplet-based SoCs.
- **Topology and Traffic Modeling:** Support for complex topologies (e.g., mesh, ring, or hierarchical interconnects), as well as various traffic models (bursty, priority-based, etc.) can be extended to more accurately reflect realistic chiplet systems.
- **Integration with Physical Design (PD):** Connecting the simulation model to physical design tools will allow the mapping of performance metrics (e.g., latency, throughput) to physical constraints such as area, routing complexity, and power, and ultimately enable co-optimization of architectural and layout-level parameters.
- **Improvements to Protocol Layer:** More extensive implementation of flit-level modeling, protocol compliance checking, and fault injection will improve the utility of the framework with stress tests and reliability verification.
- **Open Research Platform:** With acceptable documentation and an appropriate level of abstraction, this infrastructure could be transformed into an open-source research platform for academic curriculum development, practical protocol training, and collaborative development within the burgeoning chiplet ecosystem.

By continuing to develop this simulated environment, we hope to support a deeper understanding of UCIe systems, and contribute towards the innovation of future chiplet-based computing architectures.

References

[1] G. Mounce, J. Lyke, S. Horan, W. Powell, R. Doyle, and R. Some, “Chiplet based approach for heterogeneous processing and packaging architectures,” Air Force Research Laboratory, Kirtland AFB, NM, USA, Tech. Rep., 2023.

[2] Y. Feng and K. Ma, “Chiplet actuary: A quantitative cost model and multi-chiplet architecture exploration,” Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China, Tech. Rep., 2023.

[3] T. Li, J. Hou, J. Yan, R. Liu, H. Yang, and Z. Sun, “Chiplet heterogeneous integration technology—Status and challenges,” Electronics, vol. 9, no. 4, pp. 670, Apr. 2020.

[4] Y. Feng, D. Xiang, and K. Ma, “Heterogeneous die-to-die interfaces: Enabling more flexible chiplet interconnection systems,” in Proc. 56th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO), Toronto, ON, Canada, Oct. 2023, pp. 1–14.

[5] J. Kim and S. K. Lim, “Advances in 3D chiplet integration for high-performance computing,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 31, no. 6, pp. 789–802, Jun. 2023.

[6] A. Arunkumar and R. G. Dreslinski, “Power-efficient chiplet architectures for heterogeneous systems,” IEEE Micro, vol. 43, no. 4, pp. 56–64, Jul./Aug. 2022.

[7] Universal Chiplet Interconnect Express (UCIe) Consortium, “UCIe 1.0 Specification,” Mar. 2022. [Online]. Available: https://www.ucie.org/specification

[8] G. Hellings and E. Beyne, “3D System Integration: Technologies and Challenges,” IEEE Transactions on Electron Devices, vol. 68, no. 4, pp. 1504-1511, Apr. 2021.

[9] P. Onufryk and S. Choudhary, “UCIe: Standard for an Open Chiplet Ecosystem,” IEEE Micro, vol. 45, no. 1, pp. 17–27, Jan./Feb. 2025.