UCIE PROTOCOL SIMULATION USING SYSTEMC FOR CHIPLET-BASED CPU-MEMORY COMMUNICATION

Presented By : Jana Qatosa, Jana Sawalmeh

WHY UCIE SIMULATION MATTERS

- **Industry Shift: The semiconductor landscape is rapidly evolving toward chiplet-based architectures to enhance scalability, reduce costs, and boost performance.**

- **The Challenge: Seamless high-speed interconnects are essential for chiplets to function cohesively—this makes UCIe (Universal Chiplet Interconnect Express) a foundational standard.**

WHY UCIE SIMULATION MATTERS

- **Our Mission: Build a SystemC-based UCIe simulation that models CPU-to-memory chiplet communication, enabling early-stage design validation, exploration, and tuning.**

- **The Payoff: This simulation framework helps accelerate chiplet development, minimize design risks, and contribute to the adoption and standardization of UCIe across the industry.**

CHALLENGES IN UCIE DESIGN VALIDATION

- **Problem: Validating UCIe interconnects for high-speed, low-latency CPU-memory communication is complex due to:**
    a. **Diverse protocol layers (Physical, Adapter, Protocol).**
    b. **Error handling (CRC, retries, link recovery).**
    c. **Performance metrics (latency, throughput, BER).**
- **Gap: Lack of accessible, modular simulation tools to model UCIe behavior under various conditions.**
- **Objective: Create a SystemC simulation to model UCIe transactions, analyze performance, and handle errors effectively.**

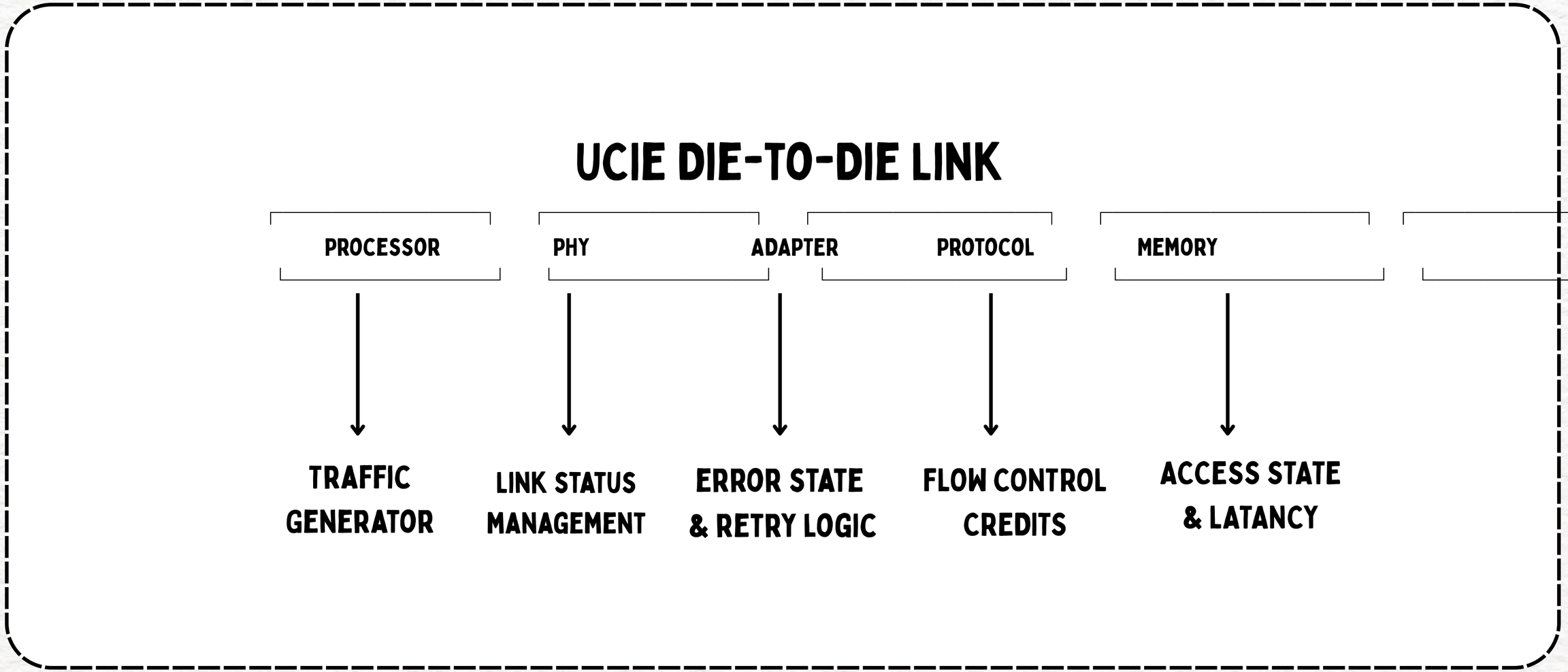
OUR KEY CONTRIBUTIONS

- **Developed a SystemC-based UCIe simulation framework for CPU-memory communication.**
- **Modeled key UCIe components: Processor, PHY, Die-to-Die Adapter, Protocol Layer, and Memory.**
- **Implemented error detection (CRC), retry mechanisms, and link state management per UCIe specifications.**
- **Provided detailed performance metrics (latency, throughput, BER, retries) for design analysis.**
- **Demonstrated transaction generation and processing with sequential data patterns and deterministic addressing.**

CORE FEATURES OF THE UCIE SIMULATION

- **Modular Design: Separated Processor, PHY, Adapter, Protocol, and Memory modules for flexibility.**
- **UCIe Compliance: Supports 256B flits, 16 GT/s data rate, and dynamic multiplexing.**
- **Error Handling: CRC-based error detection, replay mechanism, and link recovery states.**
- **Configurability: Adjustable transaction rates, read/write ratios, and memory sizes.**
- **Monitoring: Real-time statistics for latency, throughput, BER, and retry success rates.**

SYSTEM ARCHITECTURE



TRANSACTION WORKFLOW

- **Steps:**
  a. **Processor: Generates transactions (30% reads, 70% writes) with sequential data (0x00, 0x01, ...) .**
  b. **PHY: Forwards transactions, manages lanes (16 lanes).**
  c. **Adapter: Validates flit format, checks CRC, processes packets (data, retry, control).**
  d. **Protocol: Executes memory operations, handles flow control.**
  e. **Memory: Performs read/write with configured latencies, updates statistics.**
  f. **Error Handling: CRC errors trigger error recovery; invalid flits are rejected.**

DEVELOPMENT METHODOLOGY

Implementation

- **Developed using SystemC.**
- **Leveraged TLM 2.0 for efficient transaction-level modeling.**
- **Compliant with UCIe 1.0 specification:**
  - **Flit size, CRC computation, and link state handling.**
- **Modular C++ design:**
  - **Separate files for each module (processor.cpp, adapter.cpp, etc.).**
- **Included configurable simulation parameters:**
  - **burst_length = 256**
  - **memory_size = 1 GB**

VALIDATION METHODOLOGY

- Transaction flow verified via detailed logging outputs.
- Monitored key metrics:
  - Latency, Bit Error Rate (BER), Retry Count.
- Used GDB for debugging runtime issues (e.g., segmentation faults).
- Added enhanced logging to support traceability and error analysis.

SIMULATION SETUP

- Environment Setup
  - Platform: Linux (64-bit)
  - Simulator: SystemC 3.0.1 (lib-linux64)
  - Clock Frequency: 100 MHz (10 ns period)
  - Reset Pulse: 100 ns
  - Simulation Time: 3000 ns
- Metrics Collected
  - Total read/write transactions
  - Packets processed
  - CRC errors detected
  - Transaction latency
  - Bit Error Rate (BER)

RESULTS AND ANALYSIS

THE SIMULATION SHOWS ROBUST AND BALANCED CPU–MEMORY COMMUNICATION THROUGH THE UCIE STACK:

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

RESULTS AND ANALYSIS

- **Overall, the UCIe system achieved low latency, high throughput, and error-free operation, validating the design for mixed read/write traffic under simulated conditions.**

SUMMARY

- Developed a SystemC UCIe simulation for CPU-memory communication.
- Successfully modeled UCIe stack with error handling and performance monitoring.
- Addressed segmentation faults through logging and robust checks.

FUTURE WORK

- **Multi-Component Extension**

  Include GPU, NPU, and accelerator chiplets to analyze heterogeneous traffic.

- **Topology & Traffic Modeling**

  Support hierarchical topologies; add priority-based traffic.

- **Integration with Physical Design**

  Map performance (latency, power) to physical constraints for co-optimization.

- **Protocol Layer Enhancements**

  Add flit-level modeling, compliance checks, fault injection & stress testing.

- **Open Research Platform**

  Release as open-source for teaching, protocol exploration, and community use.

THANK YOU