Nebula Submission

Implementation of Scalable GPU Interconnects

with AXI/CHI Protocols & Network-on-Chip Architecture

Technical Abstract

Proposed Solution for Scalable AI Computing Infrastructure

Project Category:	Hardware/Software Co-Design
Target Scale:	4-64 GPUs (2x2 to 8x8 grid configurations)
Key Technologies:	ARM AMBA AXI4/CHI, Network-on-Chip, SystemVerilog
Implementation:	RTL, SystemC TLM-2.0, Python Analysis Framework

August 2025

1 Problem Statement and Motivation

Current GPU interconnect solutions face fundamental architectural limitations when scaling beyond 8-16 GPUs, creating critical bottlenecks for large-scale AI training and inference systems. Traditional PCIe-based connections exhibit **O(1) bandwidth scaling** characteristics, where aggregate bandwidth remains constant regardless of the number of GPUs, fundamentally limiting system performance. Point-to-point topologies fail to provide efficient routing for many-to-many communication patterns prevalent in modern AI workloads, while memory coherency protocols between GPUs require extensive software management, introducing significant latency overhead.

The growing demand for larger AI models necessitates distributed training across 32+ GPUs, yet existing interconnect solutions cannot efficiently support such scales. Software-managed coherency protocols add 100-1000+ cycle overhead for inter-GPU memory operations, while PCIe fabric contention creates unpredictable latency characteristics that severely impact training convergence times.

Our objective is to develop a standardized, hardware-based interconnect solution that achieves **O(N) bandwidth scaling** with the number of GPUs while maintaining full protocol compatibility with existing GPU architectures and providing deterministic, low-latency communication guarantees.

2 Technical Approach: Nebula Submission

2.1 System Architecture Overview

We propose a hierarchical interconnect system implementing ARM AMBA AXI4 and CHI protocols over a 2D mesh Network-on-Chip topology. Our solution consists of four integrated components providing seamless GPU-to-GPU communication with hardware-managed cache coherency.

GPU 0,0	**GPU 1,0**	**GPU 2,0**	**GPU 3,0**
GPU 0,1	**GPU 1,1**	**GPU 2,1**	**GPU 3,1**
GPU 0,2	**GPU 1,2**	**GPU 2,2**	**GPU 3,2**
GPU 0,3	**GPU 1,3**	**GPU 2,3**	**GPU 3,3**

Figure 1: Example 4x4 Nebula Submission Grid Topology (16 GPUs)

2.2 Component 1: AXI4/CHI Protocol Implementation

2.2.1 AXI4 Interface State Machine Design

We implement sophisticated finite state machines managing five independent channels (AW, W, B, AR, R) with comprehensive transaction tracking:

- **Outstanding Transaction Management**: Hardware tables tracking up to 16 concurrent transactions per interface with transaction ID mapping and completion ordering
- **Burst Transaction Optimization**: Support for 256-beat transactions with WRAP, INCR, and FIXED burst types, including boundary protection logic
- **Wide Data Path Implementation**: 512-bit data paths structured as 8×64-bit lanes with sophisticated byte-enable generation and data steering
- **Advanced Flow Control**: Credit-based back-pressure management with configurable thresholds and adaptive rate control

2.2.2 CHI Coherency Engine Architecture

Our CHI implementation provides hardware-managed cache coherency through three distinct processing engines:

- **Request Node (RN) Engine**: Manages cache line requests, implements snoop filtering, and handles coherency state transitions with atomic updates
- **Home Node (HN) Engine**: Acts as memory controller interface with distributed directory maintenance and transaction serialization
- **Snoop Filter Logic**: Hardware-based directory tracking cache line ownership across all grid nodes, enabling efficient invalidation broadcasts
- **Five-State Protocol**: Complete implementation of I, UC, UD, SC, SD coherency states with race condition prevention through message ordering

2.3 Component 2: Network-on-Chip Router Microarchitecture

2.3.1 Five-Stage Router Pipeline

Each router implements a sophisticated pipeline architecture optimized for low latency and high throughput:

1. **Buffer Write (BW)**: Flit validation, error detection, and virtual channel allocation with conflict resolution
2. **Route Computation (RC)**: XY routing algorithm implementation with adaptive routing extensions and lookahead computation
3. **Virtual Channel Allocation (VA)**: Downstream VC allocation with deadlock prevention and priority management
4. **Switch Allocation (SA)**: Crossbar arbitration using round-robin and priority schemes with starvation prevention
5. **Switch Traversal (ST)**: Physical transmission with credit generation and comprehensive error monitoring

2.3.2 Advanced Buffer Architecture

Input buffering implements sophisticated virtual channel management:

- **Multi-Level Buffering**: 16-flit deep buffers per virtual channel (4 VCs per port) with shared buffer pool optimization
- **State Machine Management**: Four-state VC management (Idle, Routing, Active, Waiting) with transition optimization
- **Credit-Based Flow Control**: Separate credit counters per VC with adaptive back-pressure mechanisms
- **Congestion Monitoring**: Real-time buffer occupancy tracking for adaptive routing feedback and load balancing

2.3.3 Routing Algorithm Implementation

Our routing implementation provides deadlock-free operation with performance optimization:

- **XY Routing Core**: Coordinate comparison logic with mathematical deadlock freedom proof through dimensional ordering
- **Adaptive Extensions**: Congestion monitoring with alternative path computation maintaining deadlock freedom
- **Load Balancing**: Weighted path selection algorithms with hotspot detection and mitigation
- **Virtual Channel Ordering**: Dependency tracking preventing cyclic wait conditions across message classes

2.4 Component 3: Network Interface Protocol Translation

2.4.1 AXI-to-NoC Translation Engine

Seamless protocol conversion between burst-oriented AXI and packet-switched NoC:

- **Transaction Decomposition**: Intelligent segmentation of 256-beat AXI bursts into optimal NoC packet sizes
- **State Tracking**: Hardware tables managing up to 64 outstanding operations with aging mechanisms and timeout detection
- **Address Mapping**: Configurable translation from AXI addresses to NoC destination coordinates with load balancing
- **Packet Assembly**: Header generation, payload optimization, and end-to-end error detection code insertion

2.4.2 CHI-to-NoC Protocol Mapping

Coherency protocol preservation across packet-switched infrastructure:

- **Message Classification**: CHI request/response/data messages mapped to appropriate NoC virtual channels with priority assignment
- **Coherency State Management**: Distributed state tracking across packet-switched network with snoop traffic optimization
- **Cache Line Optimization**: 64-byte cache line segmentation with integrity checksums and reconstruction logic
- **Ordering Preservation**: CHI completion semantics maintained through NoC packet sequencing and dependency tracking

2.5 Component 4: System Integration Architecture

2.5.1 Hierarchical Addressing Scheme

Scalable address space management supporting up to 8×8 grid configurations:

- **Global Address Space**: Partitioned addressing with dedicated ranges per GPU and automatic address translation

- **Route Computation**: Hardware logic extracting routing information directly from memory addresses
- **Collective Operations**: Broadcast/multicast support with special addressing modes for AI training primitives
- **Memory Consistency**: Hardware-enforced ordering guarantees across distributed memory hierarchy

2.5.2 Performance Optimization Mechanisms

Comprehensive optimization framework for AI workload characteristics:

- **Adaptive Routing**: Real-time congestion monitoring with dynamic path selection and load balancing
- **Quality of Service**: Traffic classification and priority queuing with bandwidth allocation guarantees
- **Transaction Optimization**: Request combining, response batching, and predictive prefetching
- **Virtual Channel Management**: Traffic class separation preventing head-of-line blocking

3 Implementation Methodology

3.1 Hardware Implementation Strategy

3.1.1 SystemVerilog RTL Design

Complete synthesizable implementation following industry standards:

- **IEEE 1800-2017 Compliance**: Synthesizable RTL using proper coding guidelines and non-blocking assignments
- **Clock Domain Strategy**: Single global clock domain with synchronous reset and planned multi-clock support
- **Protocol Verification**: Comprehensive SystemVerilog assertions for AXI4 and CHI compliance checking
- **Parameterization**: Configurable grid dimensions, buffer sizes, and data widths for scalability

3.1.2 Module Hierarchy

Modular design enabling independent development and verification:

```
gpu_grid_top
|-- axi4_master (per GPU connection)
|-- axi4_slave (per GPU connection)
|-- chi_request_node (per GPU)
|-- chi_home_node (per memory controller)
|-- noc_router (per grid position)
|-- network_interface (AXI/CHI to NoC)
‘-- grid_interconnect (top-level mesh)
```

3.2 System-Level Modeling

3.2.1 SystemC TLM-2.0 Implementation

High-level system exploration and performance analysis:

- **Transaction-Level Models**: Blocking transport interfaces with accurate timing annotation using `sc_time`
- **Abstraction Levels**: Support for functional, cycle-approximate, and cycle-accurate modeling
- **Performance Modeling**: Router delays, serialization effects, and protocol overhead characterization
- **Parameterization**: Template-based grid dimensions and runtime configuration support

3.3 Comprehensive Verification Framework

3.3.1 Multi-Level Verification Strategy

Ensuring functional correctness and protocol compliance:

- **Protocol Compliance**: AXI4 and CHI protocol checkers with formal verification for critical properties
- **UVM Testbenches**: Constrained random verification with functional coverage tracking
- **Deadlock Detection**: Runtime monitoring for circular wait conditions and livelock prevention
- **FPGA Emulation**: Extended verification on hardware platforms for realistic scenarios

3.3.2 Performance Validation

Comprehensive performance characterization and optimization:

- **Synthetic Workloads**: Traffic generators supporting uniform random, hotspot, and nearest-neighbor patterns
- **AI Training Traces**: Realistic GPU communication patterns from neural network training workloads
- **Scalability Analysis**: Automated testing across all grid configurations with bottleneck identification
- **Comparative Evaluation**: Performance comparison with PCIe-based baseline implementations

4 Performance Analysis Infrastructure

4.1 Simulation and Modeling Framework

4.1.1 Build System Integration

Coordinated development and testing infrastructure:

- **Makefile Automation**: Integrated SystemVerilog and SystemC compilation with dependency management

- **Simulator Support**: Verilator for fast simulation, ModelSim/VCS for full-featured debugging
- **Continuous Integration**: Automated regression testing with performance monitoring
- **Configuration Management**: Parameter sweeps and design space exploration automation

4.1.2 Advanced Metrics Collection

Comprehensive performance monitoring and analysis:

- **Latency Measurement**: Cycle-accurate timing with timestamp injection and extraction
- **Bandwidth Analysis**: Per-link, per-router, and per-virtual-channel utilization tracking
- **Congestion Monitoring**: Buffer occupancy histograms and hotspot identification
- **QoS Metrics**: Traffic class performance tracking and service level validation

4.2 Analysis and Visualization Tools

4.2.1 Python Analysis Framework

Automated performance analysis and report generation:

- **Data Processing**: Pandas-based statistical analysis with automated metric calculation
- **Visualization**: Matplotlib/Seaborn integration for publication-quality figures
- **Network Analysis**: NetworkX-based topology visualization and communication pattern analysis
- **Scalability Curves**: Automated generation of performance scaling characteristics

5 Expected Deliverables and Impact

5.1 Technical Deliverables

5.1.1 Hardware Implementation

Complete RTL implementation targeting FPGA prototyping:

- **SystemVerilog RTL**: Synthesizable designs for all AXI4/CHI interfaces and NoC components
- **Parameterizable Modules**: Support for 2×2 to 8×8 grid configurations with runtime optimization
- **Protocol Compliance**: Comprehensive testbenches with verified AXI4 and CHI specification compliance
- **FPGA Targets**: Synthesized implementations for Xilinx and Intel FPGA platforms

5.1.2 System Models and Analysis

High-level modeling and performance analysis framework:

- **SystemC Models**: TLM-2.0 implementations for rapid system-level exploration
- **Python Framework**: Comprehensive simulation and analysis infrastructure
- **Configuration Support**: Topology and traffic pattern configuration management
- **Build Infrastructure**: Automated compilation, testing, and analysis workflows

5.2 Performance Validation Results

5.2.1 Functional Verification

Demonstrated protocol compliance and system correctness:

- **Protocol Verification**: AXI4 and CHI specification compliance with formal verification
- **Deadlock Freedom**: Mathematical proof and runtime verification of deadlock-free operation
- **Coherency Validation**: Hardware-managed cache coherency correctness across all grid configurations
- **Error Handling**: Fault tolerance and graceful degradation under failure scenarios

5.2.2 Performance Characterization

Comprehensive performance analysis and optimization results:

- **Scaling Analysis**: Performance characterization across 2×2 to 8×8 grid configurations
- **Latency Optimization**: Cycle-accurate latency analysis with optimization recommendations
- **Bandwidth Efficiency**: Throughput analysis demonstrating O(N) bandwidth scaling
- **Comparative Evaluation**: Performance advantages over traditional PCIe-based interconnects

6 Innovation and Industry Impact

6.1 Technical Contributions

- **First Comprehensive Implementation**: Complete AXI4/CHI protocol implementation over mesh NoC for GPU interconnects
- **Novel Network Interface Design**: Seamless protocol translation with minimal latency overhead and maximum throughput
- **Scalable Router Architecture**: Advanced microarchitecture supporting 64+ GPUs with deterministic performance
- **Hardware-Managed Coherency**: Elimination of software coherency overhead in GPU-to-GPU communication

6.2 Performance Improvements

- **Bandwidth Scalability**: O(N) bandwidth scaling vs. O(1) for PCIe-based solutions
- **Latency Reduction**: Direct mesh communication eliminating CPU bottlenecks and software overhead
- **Deterministic Performance**: Predictable communication characteristics enabling optimized AI training
- **Fault Tolerance**: Redundant path support with graceful degradation capabilities

6.3 Industry Relevance and Adoption

- **Standards Compliance**: ARM AMBA protocol compatibility ensuring ecosystem integration
- **Vendor Agnostic**: Modular design supporting different GPU vendors and memory hierarchies
- **FPGA Prototyping**: Hardware validation pathway enabling real-world demonstration
- **Open Source Delivery**: Community-driven development and research collaboration

7 Conclusion

The Nebula Submission represents a comprehensive solution to the fundamental scalability limitations of current GPU interconnect architectures. Through the innovative combination of ARM AMBA AXI4/CHI protocols with advanced Network-on-Chip technology, we deliver a hardware-based interconnect solution capable of efficiently scaling to 32+ GPUs while maintaining full protocol compatibility.

Our approach addresses critical bottlenecks in current AI training infrastructure by providing O(N) bandwidth scaling, hardware-managed cache coherency, and deterministic low-latency communication. The comprehensive implementation spans RTL-level hardware design through system-level modeling and analysis, ensuring both functional correctness and performance optimization.

The expected impact includes enabling larger-scale AI training systems, reducing training times through improved communication efficiency, and providing a standards-compliant foundation for next-generation GPU computing architectures. The modular, parameterizable design ensures adaptability to different system configurations and vendor ecosystems, promoting widespread adoption and further research collaboration.

This project establishes a new paradigm for scalable GPU interconnects, moving beyond the limitations of traditional PCIe-based solutions toward a future of efficient, hardware-managed, mesh-based computing architectures specifically optimized for AI workload characteristics.