BEAM: A Multi-Channel Optical Interconnect for Multi-GPU Systems

Chongyi Yang[1], Bohan Hu[1], Peiyu Chen[1], Yinyi Liu[2], Wei Zhang[2], and Jiang Xu[1]
[1]*Microelectronics Thrust, The Hong Kong University of Science and Technology (Guangzhou)*
[2]*Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology*

***Abstract*—High-performance computing and AI applications necessitate high-bandwidth communication between GPUs. Traditional electrical interconnects for GPU-to-GPU communication face challenges over longer distances, including high power consumption, crosstalk noise, and signal loss. In contrast, optical interconnects excel in this domain, offering high bandwidth and consistent power dissipation over long distance.**

This paper proposes BEAM, a Bandwidth-Enhanced optical interconnect Architecture for Multi-GPU systems. BEAM extends electrical-optical interfaces into the GPU package, positioning them close to GPU compute logic and memory. Unlike existing single-channel approaches, each BEAM optical interface incorporates multiple parallel optical channels, further enhancing bandwidth. An arbitration scheme manages channel usage among data transfers. Evaluation on Rodinia benchmarks and LLM training kernels demonstrates that BEAM achieves a speedup of $1.14-1.9\times$ and reduces energy consumption by $29-44\%$ compared to the electrical-interconnected system and state-of-the-art schemes, while maintaining comparable chip area consumption.

I. Introduction

GPUs have become a cornerstone in high-performance computing (HPC) and AI applications, driving the need for efficient multi-GPU systems [1]. In these systems, communication between GPUs plays a crucial role in overall performance and scalability [2]. As the computational power of GPUs continues to increase, the communication subsystem faces significant challenges in keeping pace, particularly in terms of bandwidth and energy efficiency [3] [4] [5].

Current multi-GPU systems rely on copper-based electrical links for communication. However, GPUs are extremely bandwidth-intensive [6] [7]. To meet this demand, existing solutions employ multiple electrical wires to transmit data in parallel [8] [9]. However, this approach consumes substantial pin count, chip area, and power, limiting scalability for systems with higher number of GPUs and even greater bandwidth requirements [10] [5].

Optical interconnects offer a promising solution for long-distance, high-bandwidth communication in multi-GPU HPC systems. These links provide nearly distance-independent latency and power consumption, enabling energy-efficient communication over extended distances [5], [10]. Wavelength division multiplexing (WDM) technology further enhances bandwidth by allowing simultaneous transmission of multiple data streams over a single medium [10]. Already implemented in rack-scale data center communications (e.g., optical circuit switches in TPU v4 [11]), the industry is now moving towards integrating optical interfaces within packages, as evidenced by Ayar Labs' TeraPHY [12] and Intel's Optical Compute Interconnect [13].

Corresponding author: jiang.xu@hkust-gz.edu.cn

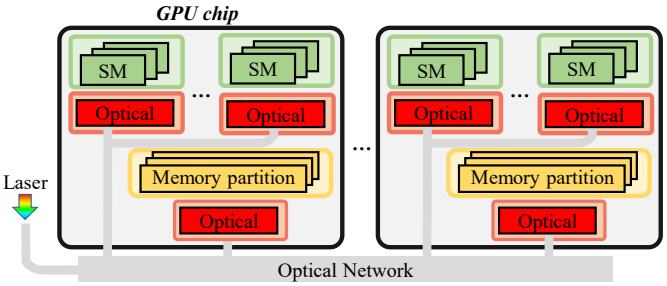


Fig. 1: The conceptual view of multi-GPU system with optical interconnects

While extensive research has focused on CPU network-on-chip and inter-CPU communication, these approaches often rely on complex network-specific schemes such as routing and arbitration, designed for diverse CPU traffic patterns, aiming at reducing response latency [14] [15] [16]. However, GPU traffic exhibits distinct characteristics - typically uniform and bursty, with GPU performance being particularly bandwidth-sensitive. Some studies have explored optical interconnects within single GPU chips, but leave the communication between GPUs underexplored [17] [18].

This paper proposes BEAM, a **B**andwidth-**E**nhanced optical interconnect **A**rchitecture for **M**ulti-GPU systems (Fig. 1). BEAM adopts a chiplet-based packaging approach, separating each GPU into multiple Streaming Multiprocessor (SM) dies for compute (green) and a memory die (yellow). Optical interfaces, implemented on separate CMOS-compatible dies (red), are co-packaged with the GPU electrical dies and extended into the GPU package, each dedicated to its associated die.

BEAM utilizes multiple waveguides within each optical interface, each serving as a discrete physical optical channel and is driven by its own optical transceiver. This approach complements existing designs that employ a single waveguide with WDM support [19]. While WDM offers bandwidth scaling proportional to the number of channels, it faces increased complexity in debugging and testing as the number of channels grows [6] [12]. BEAM addresses this challenge through waveguide duplication while considering additional costs including chip area and power consumption.

To manage the shared optical channels, BEAM implements

978-3-9826741-0-0/DATE25/© 2025 EDAA

a dynamic arbitration mechanism that allocates bandwidth resources based on parameters such as transfer size and channel availability. The scheme aims to optimize overall bandwidth utilization while considering system-wide power efficiency, adjusting channel allocation dynamically to balance bandwidth and power consumption for multiple data transfers.

Our contributions are as follows:

1) We propose an optical interconnect architecture and inter-GPU interface floorplan for multi-GPU communication. A 3D-stacked co-packaging approach for the optical die is also presented.
2) We propose a bandwidth-enhanced multi-channel optical interface as well as an arbitration scheme for multi-channel management.
3) We quantitatively evaluate system performance, energy consumption, and memory access latency compared to SEECHIP [17] and GROOT [18]. Implementation cost and chip area are also analyzed.

The rest of the paper is organized as follows. Section II outlines the overall multi-GPU architecture. Section III details the optical interconnect architecture, including channel design and the arbitration scheme. Evaluation is presented in Section IV. Finally, Section V concludes the paper.

II. The Multi-GPU System

A. Architecture Overview

The multi-GPU system is depicted in Fig. 2. It comprises four nodes, each containing four GPUs, totalling 16 GPUs. GPUs in the same column belong to the same node. This system scale reflects on an HPC node [20]. Each GPU comprises 128 SMs, each equipped with numerous CUDA cores to execute massive threads in parallel. The SMs are disintegrated into four homogeneous dies (32 SMs each), consistent with the state-of-the-art CDNA3 architecture [21].

In the light of memory hierarchy, each SM is equipped with private shared memory, texture memory, and L1 cache. The memory partitions, which function as L2 cache banks [1], are separated from the SMs and shared across the node. On-chip High Bandwidth Memory (HBM, not shown in the figure) enhances memory bandwidth and utilizes interposer technology [22]. All GPUs within a node share a unified memory address space, whereas each node maintains a unique address space.

B. Inter/Intra-GPU Network

As illustrated in Fig. 2, our proposed architecture leverages optical communication for networks between different GPUs (denoted as inter-GPU network) and between dies within each GPU (denoted as intra-GPU network).

In inter-GPU network, each optical link connects four GPUs from the same row or column, forming a 2D torus topology. While depicted as a ring in the figure, in reality, these optical fibers are freely positioned without sharp bends, ensuring minimal signal loss. Each link consists of off-chip optical fibers

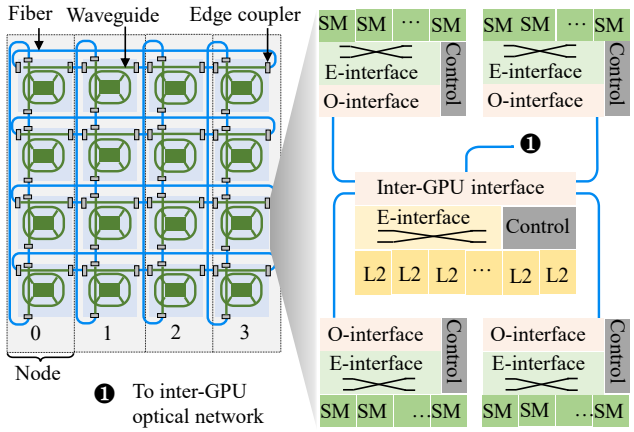


Fig. 2: The multi-GPU system architecture. The optical interconnect between GPUs is shown on the left, with the SMs, L2 caches, electrical and optical interfaces shown on the right.

and on-chip waveguides, coupled via edge couplers at the inter-GPU interface. This design of shorter optical links connecting subsets of GPUs, rather than a single long link for all 16 GPUs, reduces coupling and switching losses, thereby lowering overall laser power consumption.

In intra-GPU network, the optical interface is extended to and replicated for each SM die, allowing SMs to access local memory partitions through optical links. Each SM die has an independent optical link to the L2-side inter-GPU interface, thus preventing SMs from different chiplets contending bandwidth resources when accessing local memory partitions concurrently.

The inter-GPU interface bridges both networks, enabling SMs to access remote memory partitions. This fully optical configuration allows one-hop end-to-end packet transmission. This contrasts to electrical networks which require multiple hops for inter-GPU communication.

III. Optical Interconnect Architecture

A. Packaging

The chiplet-based GPU is co-packaged with an optical die using 3D stacking technology, as shown in Fig. 3. The optical die, 3D-stacked with the electrical SM dies or memory partition die via Through-Silicon Vias (TSVs), includes waveguides, switches, modulators, and photodetectors. Separating optical components from electrical dies mitigates temperature variation issues and leverages the short distance of TSVs. The optical die can be independently manufactured and tested for reliability. Adjacent optical dies are connected via optical fiber. This packaging approach has been revealed in recent industry prototypes, demonstrating its manufacturing feasibility [12] [23].

To transmit data, electrical signals are sent from the electrical die through TSVs to the optical die, where light is modulated into optical signals. These optical signals travel through waveguides and optical switches, and photodetectors at the destination convert them back into electrical signals for the respective electrical die.

[1]Memory partition and L2 will be used interchangeably in the rest of the paper.

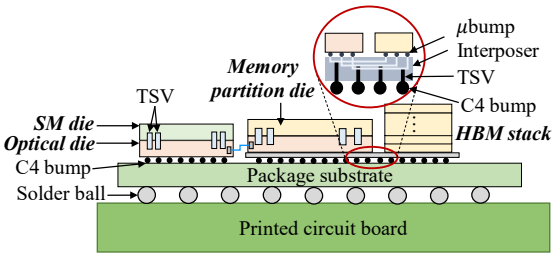


Fig. 3: Cross-section view of a chiplet-based GPU co-packaged with optical dies

B. Optical Die

The floorplan of the SM-side and L2-side optical dies is shown in Fig. 4a. The SM-side optical die is essentially a pair of electro-optical (EO) and opto-electrical (OE) interfaces that converts electrical data packets into optical signals to transmit to the L2-side optical die, and vice versa. The L2-side optical die includes two pairs of EO/OE interfaces and an inter-GPU interface that bridges both inter- and intra-GPU optical networks via MR-based optical switches. This interface manages various network traffic types, including requests, replies, memory copies, and control.

Given that GPU traffic is relatively uniform and bandwidth-intensive, we design multiple optical paths, each dedicated to a distinct network traffic type so as to avoid bandwidth contention caused by different network traffic types, as described below.

1) Request: For request packets, data originates from the *From SM* port and is routed to the L2 electrical interface. Depending on whether the memory partition being accessed is local or remote (located in its own GPU or another GPU), the routing can be as follows: local requests reach the local L2 ❶ , local SMs access remote L2 ❷ , or requests from remote SMs reach the local GPU (local GPU is the destination) ❸ .

2) Reply: Conversely, reply packets, which contain the data requested by the SM, are generated from memory partitions and directed towards the *To SM* port, either locally ❹ or remotely ❺ .

3) Direct memory copy: The memory copy network facilitates communication between memory partitions across different GPUs. Packets from local memory partitions switch to the *To L2* port to be sent to remote destinations ❻ , while remote packets arrive via the *From L2* port and switch to the local memory partition ❼ .

4) Control: Control messages originate from path reservation, which is conducted by the electrical control subsystem before optical data transfer. Path reservation involves a series of operations to reserve the optical path from the source to the destination of the data transfer, requiring collaboration among inter-GPU interfaces. This collaboration involves communication categorized as control traffic, transmitted optically between GPUs, similar to data traffic. The control subsystem is distributed across all GPUs and integrated into each electrical interface, as detailed in Section III-C.

C. Bandwidth-Enhanced EO/OE Interface

To enhance optical transmission bandwidth beyond traditional WDM technology, we utilize multiple optical data channels, physically implemented as multiple waveguides, for parallel data transmission. This approach employs a three-step process: arbitration, path reservation, and data transmission. Initially, an arbiter determines the number of waveguides allocated for each data transfer, considering both transfer size and global channel status. Next, the control subsystem reserves the specified number of waveguides. Finally, data is transmitted in parallel over these reserved optical channels.

The EO and OE interface architectures are similar, with OE operation flow reversed and lacking arbitration or control fabric for path reservation. Here we focus on the EO interface, as illustrated in Fig. 4(b). Each electrical input port, connected to an SM, feeds data transfer requests into a system of three buffers: (1) input buffer, which stores the requests awaiting arbitration and path reservation; (2) outstanding buffer, holding the requests undergoing these processes; and (3) output buffer, containing the requests with successful path reservations ready for optical transmission.

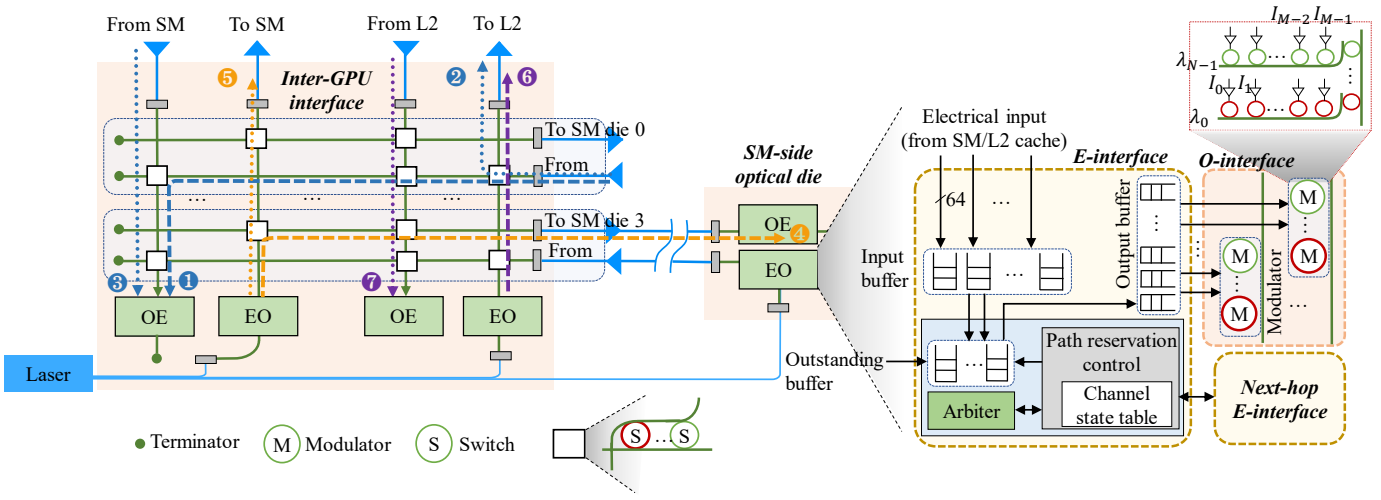
The operational flow begins when a request pops out from the input buffer and moves to the outstanding buffer. The arbiter determines the appropriate number of optical channels to assign. Next, the control subsystem initiates the path reservation process. Upon successful reservation, the data transfers will be popped out from the outstanding buffer and relocated to the output buffer. From here, data is transmitted via multiple optical channels, each driven by a dedicated modulator.

D. Control Subsystem

The control subsystem is responsible for path reservation based on the arbiter's channel allocation decisions. As shown in Fig. 4b, to facilitate efficient arbitration, each path reservation control fabric maintains a local two-dimensional channel state table (CST). In this table, rows represent per-hop multi-channel status, columns represent per-channel status, and each table cell is a single bit indicating channel availability. This design allows arbiters to access channel status information without the need for remote queries. To maintain channel status coherence across the system, channel status updates are broadcast to all tables via optical control paths.

In our system configuration of 16 GPUs arranged in a 2D torus topology with N bidirectional waveguides per hop, the storage overhead for the channel state table is as follows: With 4 hops and N waveguides per direction, each GPU requires $4N$ bits to represent the availability of all waveguides. Thus, the system-wide storage requirement amounts to $64N$ bits per EO interface.

We choose 8 channels for both reply and memory copy networks since large messages require more bandwidth. For request and control networks, where messages are short in size, a single channel suffices. This design is chip area-aware while meeting bandwidth needs. The edge couplers are placed at the package edge, not affecting the electrical pin area faced at the bottom. This allows us to increase some optical channels without affecting overall chip packaging.



(a) (b)

Fig. 4: The optically interconnected multi-GPU architecture. (a) Optical dies and their floorplan: dashed and dotted arrow lines represent the optical paths for local and remote traffic, respectively, while lines in different colors indicate traffic from different network semantics. (b) The EO interface at the sender side.

E. Arbitration Scheme

The arbitration scheme is implemented in the arbiter hardware (see Fig. 4b). The complete algorithm is presented in Algorithm 1. It operates in three main steps:

1) Adaptive Scaling: To prevent large transfers from monopolizing resources, BEAM introduces an adaptive scaling factor that considers the current system load by comparing the total available channels to the number of awaiting transfers (Line 4).
2) Initial Channel Estimation: For each data transfer, the arbiter calculates an initial channel request to determine a baseline number of channels that could adequately serve the transfer. This estimation considers the transfer size and the per-channel bandwidth (Line 7).
3) Final Channel Allocation: The arbiter combines the initial channel request and the adaptive scaling factor to determine the final number of channels granted to each transfer. It also ensures each transfer receives at least one channel while adapting to system load (Line 8).

The algorithm has a time complexity of $O(N+M\cdot N)$, where M is the number of transfers in the input buffer. The initial step requires $O(N)$ operations to count available channels across 64 rows and N columns in CST. For each transfer, the algorithm performs $O(N)$ operations in the worst case. Each arbitration involves $64N$ reads, 1 write, 1 read, and 2 arithmetic operations for initialization. Each transfer requires $2N+5$ reads, $N+1$ writes, and 4 arithmetic operations in the worst case, where all N channels in a row are allocated.

IV. Evaluation

We evaluate our inter/intra-GPU optical interconnect architecture against three other configurations (see Table I). Our analysis encompasses full-system performance and power metrics, and network-scale impact on overall system behavior.

Algorithm 1 Adaptive multi-channel arbitration scheme

Require: Input buffer IB, Channel status table $CST[64][N]$
Ensure: Channel allocations A

1: $A \leftarrow \emptyset$
2: $N_a \leftarrow \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} CST[i][j]$ /*Count total available channels*/
3: $N_t \leftarrow$ Size(IB) /*Get the number of awaiting transfers*/
4: $F \leftarrow \min(N_a/N_t, 1)$ /*Calculate adaptive scaling factor*/
5: **while** IB is not empty **do**
6: $T \leftarrow$ IB.pop() /*Get next transfer from queue*/
7: $C_b \leftarrow T.size \gg \log_2 BW$ /*Initial channel estimation*/
8: $C_r \leftarrow \max(1, C_b \times F)$ /*Final channel allocation*/
9: $R \leftarrow T.waveguide$ /*Get waveguide row from transfer*/
10: $C_a \leftarrow \sum_{j=0}^{N-1} CST[R][j]$ /*Count available channels in row R*/
11: $C_s \leftarrow \min(C_r, C_a)$ /*Determine allocated channels*/
12: **if** $C_s > 0$ **then**
13: $A[T] \leftarrow C_s$ /*Record allocation for transfer T*/
14: $j \leftarrow 0$
15: **while** $C_s > 0$ and $j < N$ **do**
16: **if** $CST[R][j] == 1$ **then**
17: $CST[R][j] \leftarrow 0$ /*Update CST*/
18: $C_s \leftarrow C_s - 1$
19: **end if**
20: $j \leftarrow j + 1$
21: **end while**
22: **else**
23: IB.push(T) /*Push back to queue if no channels available*/
24: **end if**
25: **end while**
26: **return** A

A. Experimental Setup

Simulator. We modifed Accel-Sim [24] and AccelWattch [25] to simulate the performance and power of the chiplet-based GPU architecture. The latency and energy consumption of the electrical and optical interconnect are analyzed by OEIL [26]. Table II details the GPU parameters.

Benchmark. We use HPC benchmarks from Rodinia 3.0 [27] and a distributed multi-GPU adapted transformer model train-

TABLE I: Comparison of interconnect configurations

Scheme	Description
Electrical	GPU system connected via traditional electrical links is set as baseline.
SEECHIP [17]	A reconfigurable unicast and multicast optical bus design between GPU SM dies and memory partition dies. We extend the same optical channel design approach between GPUs.
GROOT [18]	A group-based optical channel design that connects chiplets within one GPU only. We extend the same optical channel design approach between GPUs.
BEAM	Our proposed multi-channel optical interconnect architecture employed in multi-GPU system

ing [28]. The LLM model size adapts to GPU memory capacity. We identified three resource-intensive kernels via the Nsight Compute CLI profiler [29]: a compute- and memory-intensive feed-forward kernel, and two vectorized element-wise operation kernels for backward propagation and weight update, with the latter involving significant inter-node communication.

TABLE II: GPU System Parameters

Parameter	Value
Compute Subsystem	
SM	4 SM dies, 32 SMs per SM die, 32 warps per SM, round-robin warp scheduling, @1 GHz
Memory hierarchy	
L1 cache	32 KB per SM, 128 B lines, 4 ways
L2 cache	4 MB per chiplet (8 slices, 16-way, 256-set)
Global memory	4 GB per stack×4 stacks, 180 GB/s per stack, 100 ns, FR-FCFS scheduler
Electrical network	
Link	Inter-die: 16 B flit; 42 GB/s, 2 ns, 2 pJ/b; Inter-GPU: 4 lanes × 50 Gb/s per lane, 4.21 pJ/b [26]
Network	Packet size (request/reply/memory copy/control): 1/8/8/1 flits; inter-GPU: XY routing, 2D torus
Optical Network	
Device loss	MR passing: 0.06 dB; MR insertion: 1 dB; waveguide propagation: 0.5 dB/cm; edge coupling: 2 dB
Device power	Laser power conversion efficiency: 30%; receiver sensitivity: -20 dBm; MR tuning: 0.65 mW
Link power	Inter-die: 3.08 pJ/b reply, 2.08 pJ/b request, Inter-GPU: 3.23 pJ/b reply, 2.22 pJ/b request [26]
Link bandwidth	Reply and memory copy network: 8 links, 50 Gbps×8 WDM channels per link, $OI(32,8)$; request and control network: 1 link, 10 Gbps×1 WDM channel per link, $OI(2,1)$
Arbiter	96-entry outstanding buffer, 64×8 CST size for reply and memory copy network

B. System Performance

We evaluate the system performance across four interconnect configurations: electrical (baseline), SEECHIP [17], GROOT [18], and our proposed BEAM architecture. Fig. 5 depicts the system performance relative to the baseline system connected by electrical links. Overall, the system performance benefits significantly from the integration of optical interconnects, with BEAM achieving a 1.9× speedup compared to the baseline. SEECHIP and GROOT achieve speedups of 1.14× and 1.47×, respectively. Notably, the benchmarks *nn*, *streamcluster*, and all LLM-induced kernels exhibit over 2× performance improvement with BEAM.

To gain insights into how optical interconnects impact system performance, we analyze the average memory access latency.

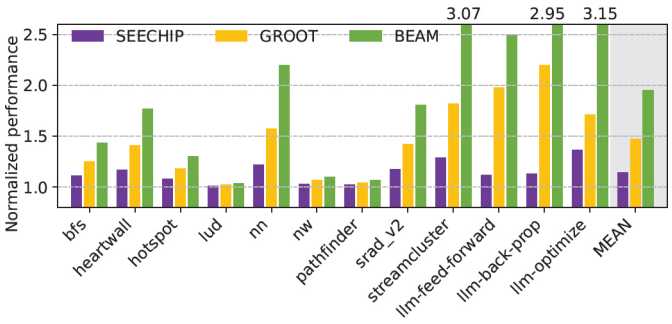


Fig. 5: Normalized system performance comparison for different interconnect configurations

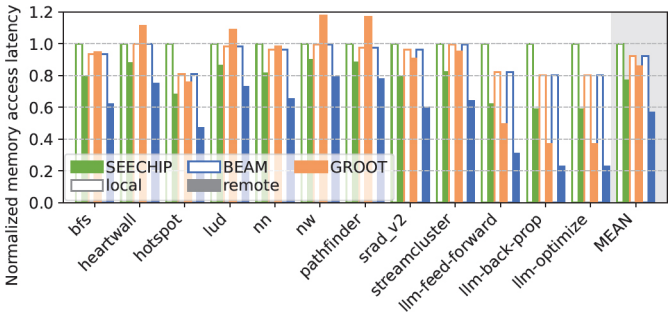


Fig. 6: Memory access latency breakdown

Memory access latency refers to the time from SM issuing memory requests, travel through inter/intra-GPU interconnect (depending on local or remote memory partition that the data locates), access memory and receive the data reply. Fig. 6 illustrates the memory access latency categorized by whether the requested data just resides in its own GPU (local) or resides in other GPUs within the node (remote). Overall, the memory access latency is inversely proportional to system performance. BEAM reduces memory access latency by 44%, with *hotspot* and all three LLM kernels reducing by more than 50%. In contrast, SEECHIP and GROOT show less improvement for benchmarks like *heartwall*, *lud*, *nw*, and *pathfinder*, due to bandwidth limitations of their single-waveguide designs. BEAM's multi-channel approach allows for higher bandwidth utilization, which benefits remote memory accesses in communication-intensive applications.

C. Energy Consumption and Energy Delay Product

1) System Energy Breakdown: Fig. 7 shows the system energy breakdown normalized to the electrical baseline configuration. The system power is composed of constant, static, and dynamic power, with dynamic power primarily originating from compute, memory, and network subsystems. BEAM reduces system power by 44%, 38%, and 29% compared to Electrical, SEECHIP, and GROOT, respectively. For BEAM, dynamic power accounts for 71% of system power, respectively. For all four configurations, *llm-optimize* consumes $75-82\%$ of dynamic power, whereas *lud, nw, pathfinder* take up only $53-55\%$ of dynamic power, with relative constant energy

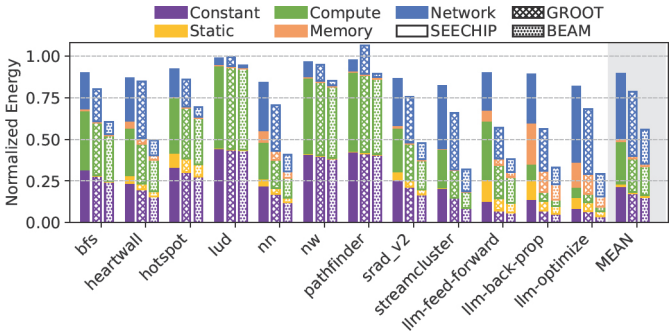


Fig. 7: System energy breakdown

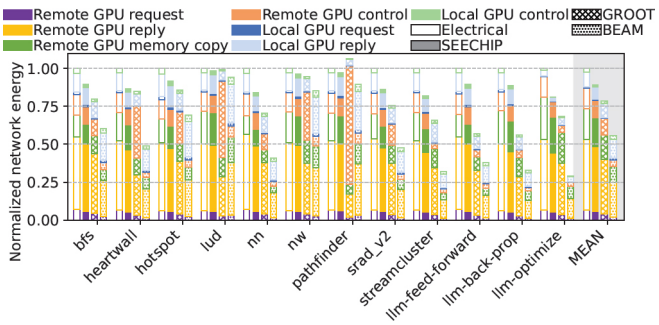


Fig. 8: Network energy breakdown

reaching over 40%. Such difference is influenced by the amount of network traffic and compute load, with lower network traffic resulting in lower power consumption.

2) Network Energy Breakdown: Dynamic power in BEAM is primarily split between compute (44%) and network (52%) subsystems. Despite higher optical static laser power and additional power for arbitration and CST updates, BEAM's faster execution offsets these costs. BEAM reduces network energy consumption by 62-69% compared to other configurations, with compute and memory energy also decreasing by 35-65% due to shorter execution time.

For optical interconnects, the network energy incorporates both optical transmission and optical path setup by the electrical control agent and MR tuning. Figure 8 further breaks down the network energy based on the traffic type. The total network energy of BEAM is reduced by 34%, 23%, and 44% compared to SEECHIP, GROOT, and baseline, respectively. For baseline, SEECHIP, GROOT, and BEAM, remote access consumes $6.2\times, 7.3\times, 5.4\times$ and $2.4\times$ more network energy than local access, primarily due to higher optical path setup overhead and lower energy efficiency for long-haul electrical links.

3) Energy-Delay Product: Fig. 9 demonstrates the energy-delay product (EDP) normalized to the baseline configuration. While BEAM incurs additional power costs from its multi-channel design, the reduced program runtime still results in lower EDP compared to SEECHIP and GROOT.

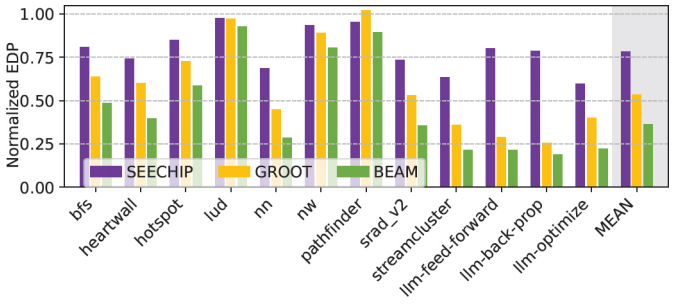


Fig. 9: Normalized EDP for different interconnect configurations

D. Optical Die Implementation Cost

Table III shows the optical device cost with respect to the number of fibers, waveguides, and MRs within each GPU. Our design incorperates a comparable number of MRs of EO [18].

In terms of area consumption, given a MR diameter of 10 μm and a waveguide pitch of 2 μm, the SM-side and L2-side optical dies consume approximately 0.43 mm^2 and 1.21 mm^2, respectively. These areas primarily result from the MRs, as edge couplers are one-dimensional and are placed at the package edge, making the area impact trivial. Compared to a standard NVLink interface ($\approx 1.8mm^2$ [9]), each SM-side die occupies about 24% of the area, while the L2-side die uses about 67%. The total optical die area is 2.93 mm^2 per GPU, which is about $1.63\times$ of a NVLink interface. This demonstrates that our design's area overhead is relatively modest.

TABLE III: Optical device cost per GPU

Component	SM-side	L2-side	Total
Fiber count	10	42	52
Waveguides	36	216	252
Transceiver MR	5504	15456	21472
Optical switching MR	0	512	

V. Conclusion

In this paper, we have proposed BEAM, a multi-channel optical interconnect architecture for multi-GPU systems that addresses power and bandwidth challenges of traditional electrical interconnects. Key features include an inter-GPU interface bridging inter- and intra-GPU networks, multi-channel EO/OE interfaces for enhanced bandwidth, and efficient arbitration for channel allocation. Evaluations demonstrate a $1.14 - 1.9\times$ speedup and $29 - 44\%$ energy reduction compared to existing state-of-the-art approaches. Future work envisions co-designing optical interconnects with memory controllers to support page migration and addressing the need for disaggregated computing powered by optics.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (No.62474152), Guangzhou-HKUST(GZ) Joint Funding Program (No. 2023A03J0013), Nansha District Key Area S&T Scheme (No. 2024ZD007), and InnoHK ACCESS.

REFERENCES

[1] D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro *et al.*, "Efficient large-scale language model training on GPU clusters using megatron-LM," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–15.

[2] Z. Jiang, H. Lin, Y. Zhong, Q. Huang, Y. Chen, Z. Zhang, Y. Peng, X. Li, C. Xie, S. Nong *et al.*, "{MegaScale}: Scaling large language model training to more than 10,000 {GPUs}," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024, pp. 745–760.

[3] L. Dai, H. Qi, W. Chen, and X. Lu, "High-speed data communication with advanced networks in large language model training," *IEEE Micro*, 2024.

[4] H. Qi, L. Dai, W. Chen, Z. Jia, and X. Lu, "Performance characterization of large language models on high-speed interconnects," in *2023 IEEE Symposium on High-Performance Interconnects (HOTI)*. IEEE, 2023, pp. 53–60.

[5] R. Mahajan, X. Li, J. Fryman, Z. Zhang, S. Nekkanty, P. Tadayon, J. Jaussi, S. Shumarayev, A. Agrawal, S. Jadhav *et al.*, "Co-packaged photonics for high performance computing: Status, challenges and opportunities," *Journal of Lightwave Technology*, vol. 40, no. 2, pp. 379–392, 2021.

[6] D. Patterson, I. De Sousa, and L.-M. Achard, "The future of packaging with silicon photonics," *Chip Scale Rev*, vol. 21, no. 1, pp. 1–10, 2017.

[7] S. B. Yoo, "Prospects and challenges of photonic switching in data centers and computing systems," *Journal of Lightwave Technology*, vol. 40, no. 8, pp. 2214–2243, 2021.

[8] J. Choquette, "Nvidia Hopper H100 GPU: Scaling performance," *IEEE Micro*, 2023.

[9] J. Choquette, W. Gandhi, O. Giroux, N. Stam, and R. Krashinsky, "Nvidia A100 tensor core GPU: Performance and innovation," *IEEE Micro*, vol. 41, no. 2, pp. 29–35, 2021.

[10] C. J. Nitta, M. Farrens, and V. Akella, *On-Chip hotonic interconnects: A computer architect's perspective*. Springer Nature, 2022, pp. 2–4.

[11] N. Jouppi, G. Kurian, S. Li, P. Ma, R. Nagarajan, L. Nai, N. Patil, S. Subramanian, A. Swing, B. Towles *et al.*, "TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–14.

[12] Ayar Labs, "TeraPHY Optical I/O Chiplet," 2024. [Online]. Available: https://ayarlabs.com/teraphy/

[13] Intel Corporation, "Intel® Silicon Photonics," 2024. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/network-io/silicon-photonics.html

[14] Y. Pan, J. Kim, and G. Memik, "Flexishare: Channel sharing for an energy-efficient nanophotonic crossbar," in *The International Symposium on High-Performance Computer Architecture (HPCA)*, 2010, pp. 1–12.

[15] D. Vantrease, N. Binkert, R. Schreiber, and M. H. Lipasti, "Light speed arbitration and flow control for nanophotonic interconnects," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2009, pp. 304–315.

[16] P. Yang, Z. Pang, Z. Wang, Z. Wang, M. Xie, X. Chen, L. H. K. Duong, and J. Xu, "RSON: An inter/intra-chip silicon photonic network for rack-scale computing systems," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018, pp. 1369–1374.

[17] H. Zhang, Y. Chen, Z. Huang, H. Zhang, and F. Dai, "SEECHIP: A scalable and energy-efficient chiplet-based GPU architecture using photonic links," in *Proceedings of the 52nd International Conference on Parallel Processing*, 2023, pp. 566–575.

[18] C. Li, F. Jiang, S. Chen, X. Li, J. Liu, W. Zhang, and J. Xu, "Towards scalable GPU system with silicon photonic chiplet," in *Design Automation and Test in Europe (DATE)*, 2024.

[19] M. R. Jokar, L. Zhang, J. M. Dallesasse, F. T. Chong, and Y. Li, "Direct-modulated optical networks for interposer systems," in *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, 2019, pp. 1–8.

[20] Oak Ridge Leadership Computing Facility, "Frontier spec sheet," 2019. [Online]. Available: https://www.olcf.ornl.gov/wp-content/uploads/2019/05/frontier_specsheet.pdf

[21] AMD, "AMD CDNA3 architecture: The all-new AMD GPU architecture for the modern era of HPC and AI," 2023. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/white-papers/amd-cdna-3-white-paper.pdf

[22] A. Kannan, N. E. Jerger, and G. H. Loh, "Enabling interposer-based disintegration of multi-core processors," in *Proceedings of the 48th International Symposium on Microarchitecture*, 2015, pp. 546–558.

[23] Ushering in a new era of optical connectivity. [Online]. Available: https://www.celestial.ai/about

[24] M. Khairy, Z. Shen, T. M. Aamodt, and T. G. Rogers, "Accel-Sim: An extensible simulation framework for validated GPU modeling," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020, pp. 473–486.

[25] V. Kandiah, S. Peverelle, M. Khairy, J. Pan, A. Manjunath, T. G. Rogers, T. M. Aamodt, and N. Hardavellas, "AccelWattch: A power modeling framework for modern GPUs," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 738–753.

[26] Z. Wang, J. Xu, P. Yang, X. Wang, Z. Wang, L. H. K. Duong, Z. Wang, R. K. V. Maeda, and H. Li, "Improve chip pin performance using optical interconnects," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 4, pp. 1574–1587, 2015.

[27] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *2009 IEEE International Symposium on Workload Characterization (IISWC)*, 2009, pp. 44–54.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[29] N. Corporation, "Nsight Compute CLI — NsightCompute 12.5 documentation," 2024. [Online]. Available: https://docs.nvidia.com/nsight-compute/NsightComputeCli/index.html