POLITECNICO DI TORINO
Repository ISTITUZIONALE

HTA: A Scalable High-Throughput Accelerator for Irregular HPC Workloads

Original
HTA: A Scalable High-Throughput Accelerator for Irregular HPC Workloads / Fotouhi, P.; Fariborz, M.; Proietti, R.; Lowe-Power, J.; Akella, V.; Yoo, S. J. B.. - STAMPA. - 12728:(2021), pp. 176-194. (Intervento presentato al convegno 36th International Conference on High Performance Computing, ISC High Performance 2021 tenutosi a Virtual nel 24 June - 2 July 2021) [10.1007/978-3-030-78713-4_10].

Availability:
This version is available at: 11583/2973500 since: 2022-11-30T14:05:27Z

Publisher:
Springer Science and Business Media Deutschland GmbH

Published
DOI:10.1007/978-3-030-78713-4_10

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Springer postprint/Author's Accepted Manuscript (book chapters)

This is a post-peer-review, pre-copyedit version of a book chapter published in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). The final authenticated version is available online at: http://dx.doi.org/10.1007/978-3-030-78713-4_10

(Article begins on next page)

19 April 2024

HTA: A Scalable High-Throughput Accelerator for Irregular HPC Workloads*

Pouya Fotouhi[1 [0000−0002−5891−4003]], Marjan Fariborz[1 [0000−0002−1896−1489]], Roberto Proietti[1 [0000−0001−6378−7005]], Jason Lowe-Power[1 [0000−0002−8880−8703]], Venkatesh Akella[1 [0000−0003−3014−5326]], and S. J. Ben Yoo[1 [0000−0002−7420−1871]]

University of California Davis, Davis, CA 95616, USA
{pfotouhi, mfariborz, rproietti, jlowepower, akella, sbyoo}@ucdavis.edu

Abstract. We propose a new architecture called HTA for high throughput irregular HPC applications with little data reuse. HTA reduces the contention within the memory system with the help of a partitioned memory controller that is amenable for 2.5D implementation using Silicon Photonics. In terms of scalability, HTA supports 4× higher number of compute units compared to the state-of-the-art GPU systems. Our simulation-based evaluation on a representative set of HPC benchmarks shows that the proposed design reduces the queuing latency by 10% to 30%, and improves the variability in memory access latency by 10% to 60%. Our results show that the HTA improves the L1 miss penalty by 2.3× to 5× over GPUs. When compared to a multi-GPU system with the same number of compute units, our simulation results show that the HTA can provide up to 2× speedup.

1 Introduction

The advent of exponentially-growing data-intensive applications across several domains has created a category of throughput-oriented workloads. This class of *irregular* applications impose new challenges for computer architects as their data sets are increasingly sparse and they exhibit poor locality in memory accesses. Unlike traditional compute-intensive applications, computing solutions designed for irregular applications should focus on reducing the latency and energy overheads of inevitable data movements.

The computing community has been utilizing GPUs as data-parallel accelerators given their massive throughput offerings. Though GPUs have proved to be effective as high throughput accelerators for many regular applications, we explore *specializing* data-parallel accelerators for efficient execution of *irregular* data-parallel workloads. These applications exhibit random memory access patterns, essentially making any shared component an architectural bottleneck limiting the obtainable throughput. Our main insight in designing HTA is to reduce the *contention* within the memory system and reduce the energy and performance cost of data movement.

* This work was supported in part by ARO award W911NF1910470.

On the scalability front, as we reach the end of transistor scaling, we cannot simply rely on increasing the number of compute units on a single die to scale. An alternative approach is to design processors utilizing multiple "chiplets" [13]. Chiplets assembled using advanced packaging technologies, such as multi-chip-modules (MCMs), can offer a scalable design compared to one large monolithic chip. However, the inter-chiplet communication and its energy efficiency are known as the dominant factors towards performance and scalability due to significant power penalties brought by MCM designs [4]. We propose to address this challenge by taking advantage of recent advances in 2.5D/3D packaging with Silicon Photonics, which offers advantages of significantly lower energy per bit and scalability to much larger interposers than what today's reticle size limits allow. For example, recently TSMC and Broadcom announced 1700 mm^2 interposer [36] which is twice the size of the maximum reticle size by proposing to stitch together multiple interposers together.

In this paper, we present the design, evaluation, and 2.5D/3D packaging solution of the high-throughput scalable accelerator architecture called **HTA**. HTA's memory architecture exploits a partitioned memory controller (PMC) and all-to-all SiPh interconnects replacing conventional cross-bar based systems to support nearly-contention-free, high-throughput, and scalable data movement between the compute cores and the main memory. The partitioned memory controller reduces the queuing latency by 10% to 30% which translate to 5% to 26% reduction on overall memory access latency. In addition, addressing the contention in the memory controller reduces the variations in access latency by 10% to 60% in terms of 95^{th} percentile latency. Furthermore, HTA improves the performance of the memory system and reduces L1 misses penalty by 2.3× to 5×. Evaluating our design at scale shows 1.5× speedup on average for HTA compared to a multi-GPU system for the same number of compute units.

The rest of the paper is organized as follows. Section2 presents challenges towards scaling the memory system in the state-of-the-art data-parallel accelerators. Section 2.1 describes the architecture of partitioned memory controller, utilizing an interconnect fabric described in Section 2.2. Section 2.4 presents HTA architecture which builds on top of the proposed memory system. Through simulations with the methodology described in Section 3, the performance of partitioned controller and the proposed HTA architecture are evaluated in Section 4. Section 5 presents he related work, followed by the conclusions in Section 6.

2 HTA - Background, Rationale, and Design

GPUs are the de facto choice for high throughput accelerators in the HPC domain. The left side of Figure 1 shows an overview of state-of-the-art GPUs. We identify four key challenges to the architecture shown in Figure 1 when it comes to scaling irregular applications.

1) Crossbar Radix: Increasing the number of core clusters requires increasing the radix of the electrical crossbar between the cores and the L2 caches as current systems implement a mostly uniform L2 architecture. In addition to the

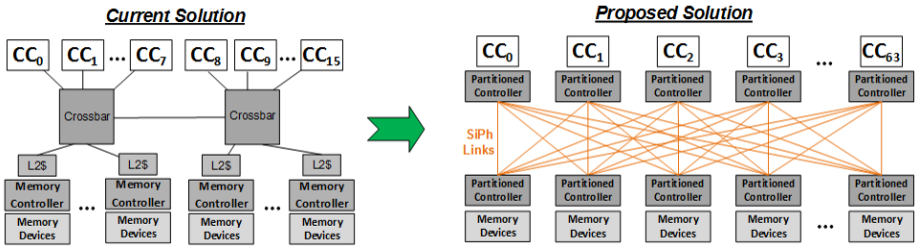


Fig. 1: (left) Overview of baseline memory system where different core clusters (CCs) share a crossbar, a single read/write queue per channel, and a last level cache. (right) Proposed memory system addresses the contention by providing dedicated queues for each core cluster to send memory request to every channel through an all-to-all interconnect.

power and area overheads of the crossbar, it imposes a trade-off between latency and bandwidth: to increase the bisection bandwidth there must be more layers in the crossbar increasing both latency and area.

2) Overheads of Data Movement: Moving the data through multiple levels of memory hierarchy adds to memory access latency and results in increased energy consumption. This challenge becomes more important as physical distance between different levels increases in multi-chip module systems. In fact, the performance and energy overheads of data movements are known to be the main limiting factor towards scalability of multichip modules systems [4].

3) Bandwidth to Memory: Scaling the number of compute units in the system increases the demand for bandwidth to memory. Already limited by the latency-bandwidth trade-off due to the crossbar design, the number of available pins (between the compute dies and memory) add another constraint on bandwidth, especially in chiplet-based designs.

4) Variability in Memory Latency: Memory requests from different processing units share many deep queues including the crossbar, an L2 bank, the memory controller queues, DRAM bus, and DRAM banks. The contention from different compute units at these components increases the queuing delay which leads to variations in access latency and adds to the complexity of the scheduling for the memory controller and GPU cores.

Recent design trends from NVIDIA and AMD have taken steps to address these challenges. These solutions are inspired by similar techniques used in CPUs, and as a result, they do not address the underlying problem (i.e., contention) especially as we go towards scaling these systems. For instance, on a single GPU, NVIDIA's Ampere architecture [26] increases the number of compute units by 50% (from 84 in Volta to 128 in Ampere). To maintain a reasonable radix for the crossbar, the crossbar in is partitioned into two pieces. However, this approach introduces non-uniform latency and bandwidth to the memory, increasing the programming complexity on these systems. AMD's RDNA architecture [1] reduces the radix of the crossbar by adding a L1 cache which filters requests from

all Compute Units (CUs) within a core cluster. While this approach simplifies the crossbar design, and reduces the pressure on the globally shared L2 cache, it adds to variability in memory access latency and only helps workloads which have regular memory access patterns or temporal reuse. AMD's CDNA architecture [2] eliminates the L1 cache along with the fixed-functions logic dedicated for graphics application to free up area and power for adding more CUs. However, the crossbar (and subsequently the L2 cache) is divided into two slices to achieve a reasonable radix for the state-of-the-art electrical interconnect technologies. Similar to NVIDIA's design, this approach increases the programming complexity by introducing non-uniformity in both latency and bandwidth, and further increases the variability in memory access latency.

The *main idea* underlying our proposal for HTA is to eliminate the contention in the memory subsystem as much as possible. We focus on three sources of contention: the on-chip crossbar, the globally shared L2 cache, and the memory controller queues. Our proposal makes the following **contributions** towards addressing the sources of contention in data-parallel accelerators.

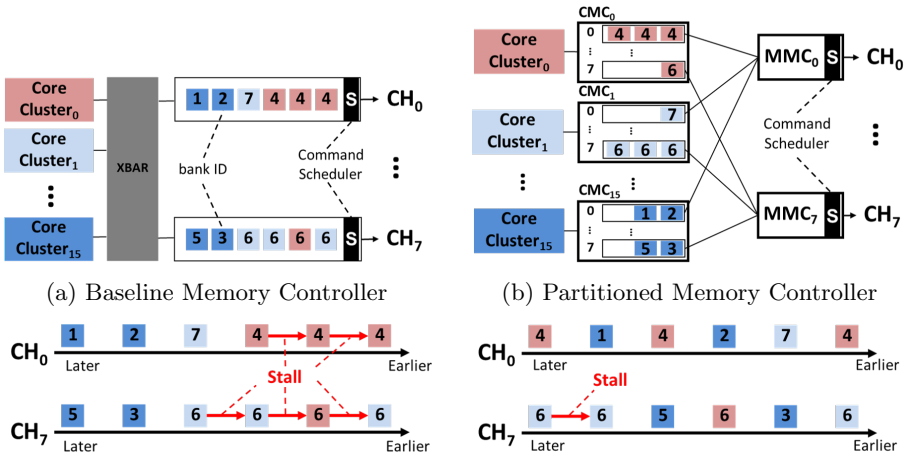
(a) To reduce the contention at the request queues, we partition the memory controller into two parts: core-side controller with dedicated queues per core cluster (CC), and memory-side controller in charge of scheduling and issuing DRAM commands. This reduces the contention on read/write queues by offering dedicated queues for each core cluster and reduces queuing latency by avoiding the head-of-line blocking in scheduling. We will discuss the architecture and scheduling of proposed memory controller in Section 2.1

(b) The contention at the crossconnect is reduced by providing direct point-to-point links. However, implementing such a topology using electrical links would be extremely challenging due to bandwidth, energy, and routing limitations. To that end, and to reduce the overhead of data movements, we leverage an efficient all-to-all passive optical fabric (called Arrayed Waveguide Grating Router or AWGR) enabled by silicon photonics by taking advantage of 2.5D packaging. Describing the key enabling technology for our architecture, the details of proposed interconnect and packaging solutions are presented in Section 2.2 and Section 2.3 respectively.

(c) We utilize the partitioned memory controller design, and propose HTA in Section 2.4, which benefits from a scalable unified memory architecture and avoid NUMA challenges.

2.1 Partitioned Memory Controller

In this section, we present the details of our proposed Partitioned Memory Controller (PMC) which consists of two parts: the compute-side memory controller (CMC) and the memory-side memory controller (MMC). For the discussions and evaluations presented in this paper, we target HBM as the DRAM device, but the core idea of our proposal is agnostic to DRAM micro-architecture and can be applied to other DRAM technologies (e.g., GDDR, DDR, etc.) in a similar fashion as we focus only on the memory controller design and require no changes to DRAM core architecture (see Section 2.3 for details).



(a) Baseline Memory Controller

(b) Partitioned Memory Controller

(c) Scheduling timeline for the baseline

(d) Scheduling timeline for PMC

Fig. 2: Working example of PMC, showing how the head-of-line blocking is addressed compared to the baseline. The stalls are avoided by scheduling requests from different core clusters, and is limited only to the conflicting requests within a core cluster.

Figure 2b presents an overall view of the components within PMC. The key idea is to eliminate the contention on request queues and improve bank utilization by avoiding stalls due to bank conflicts between requests from different core clusters. With dedicated set of queues per channel for each core cluster, the variation in the memory access latency will be limited to unavoidable conflicting patterns from a single core cluster.

While dedicated queues eliminate the contention, the memory controller still needs to have a single scheduler per bank as point of reference for DRAM timings. Thus, we *partition* the memory controller into two parts. We keep the *front-end* (containing dedicated read/write queues) on the accelerator side, and move the *back-end* (including scheduling logic, and command queues) to the memory side.

Our design requires an all-to-all interconnect between the front-end and the back-end. Section 2.2 describes how a multi-wavelength routing device called AWGR can be used to replace the long-latency electrical crossbar while offering high-throughput contention-free communication.

Compute-side Memory Controller (CMC) As Figure 2b illustrates, we keep read and write queues on the processor side, with dedicated read and write queues for each channel. The idea is to limit the contention only to requests from the CUs within a single core cluster, and not all core clusters within the system. These queues are the result of breaking down the single shared read/write queue in the baseline memory controller shown in Figure 2a into per core cluster queues.

Requests from L1 caches in each core cluster are routed to proper queues according to the address mapping scheme, similar to how corresponding L2 banks are selected for each request in the baseline architecture. Each Compute-side Memory Controller (CMC) has dedicated links to communicate with the Memory-side Memory Controller (MMC) for a given channel.

Memory-side Memory Controller (MMC) Figure 2b shows two channels of our proposed memory controller, and connectivity between MMC and read/write queues from different CMCs. The scheduler looks at requests from all core clusters regardless of their queue occupancy. Therefore, the scheduler can continue servicing memory requests even when one requester has several conflicting requests issued within a short period of time—a common case in high-throughput accelerators illustrated in Figure 2.

At each cycle, all CMCs send a copy of the request at the head of their queues. Then, an MMC selects a request to serve, it broadcasts back the requester ID (i.e., the winner) and the bank number to all CMCs. Thus, other requesters with requests for the same bank at the head of their queues can wait until the response is provided. Requests from different requesters (i.e., core clusters) are serviced in a round-robin fashion with an FR-FCFS scheduling policy similar to the baseline.

Figure 2 illustrates how the partitioned memory controller can address the head-of-line blocking problem. One core cluster (CC_0) is sending several conflicting requests (going to the same bank) to the first channel (CH_0). This results in several stalls during the scheduling. However, these stalls can be avoided by addressing non-conflicting requests from other CCs between the conflicting requests. PMC achieves this by allowing the MMC to select from dedicated queues for each channel within each CMC. Current systems use deep associative queues to avoid these stalls by finding requests to different banks within the queue. However, one CC can fill the queue with conflicting requests in a short period of time. This leaves the scheduler with no other options to choose from, even using the most sophisticated logic-intensive associative queues, and results in unnecessary back pressure applied to the whole system.

We should note that the processor's total queue size remains unchanged for each core cluster. We are essentially breaking down a large shared queue into n (*i.e.*, number of channels) smaller dedicated queues. The overhead of this is approach is limited to a small fraction to replicate the logic needed for maintaining those queues. On the memory side, there will be a small overhead for the added queues and we envision this logic to be implemented on the logic layer in 3D staked memories.

2.2 Interconnect

To address the contention at the crossconnect, our design utilizes a point-to-point connectivity between the core clusters and memory controllers. Besides addressing the contention, our proposed architecture requires an all-to-all connectivity

between CMCs and MMCs. This connectivity allows for our scheduling policy to make local decisions at the MMC and updating CMCs through broadcasting.

As discuss earlier, designing a scalable high-throughput accelerator requires addressing the cost of data movements. Disaggregating the monolithic chip into multiple smaller chiplets allows for more input/output interfaces for each core cluster. However, chiplet electrical interconnection suffers from high distance-dependent signal loss and limited I/O bandwidth [5]. Therefore, interconnecting many non-adjacent chiplets require multi-hop networks with repeaters, incurring large latency and energy overheads. These challenges can be overcome by silicon photonic technology: reducing latency with almost distance-independent communication energy and providing high pin bandwidth density through wavelength-division multiplexing (WDM) [25]. In the following sections, we present a summary on the principle of operation for optical links used in our design, and discuss the details about our proposed interconnect fabric and packaging solution.

Silicon Photonics Integrated optical interconnects, enabled by silicon photonics, offer properties that can be exploited to address the performance and energy overheads of data movements in high-throughput accelerators.

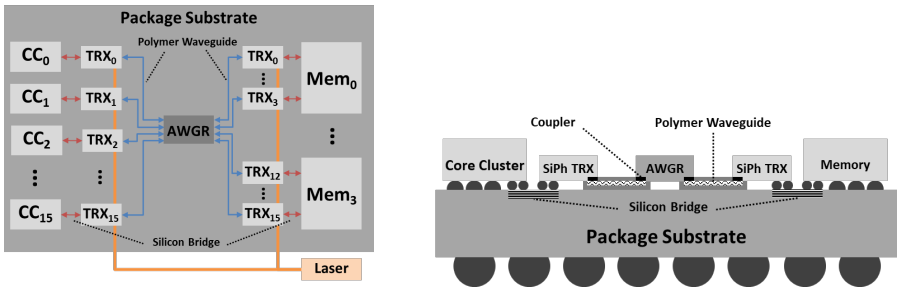
An external WDM laser (in form of an optical frequency comb source or individual lasers) generates the optical signal at the required wavelengths, which are then coupled from a fiber into on-chip waveguides. On-chip modulators encode bits onto wavelengths (one modulator for each wavelength). Then, the modulated wavelengths traverse the waveguides and are filtered out and converted back into the electrical domain by on-chip photodetectors. In terms of latency, electrical-to-optical (EO) and optical-to-electrical (OE) signal conversions are done at one cycle and incur no additional latency to the transmission line.

Arrayed Waveguide Grating Router One interesting property of WDM technology (aside from its bandwidth benefits), is that it allows connecting a single node to multiple receiver nodes by leveraging wavelength-selective routing devices. This method allows implementing an all-to-all network without a large number of point-to-point ports.

Among different SiPh wavelength routing devices that have been demonstrated [5], we utilize the Arrayed Waveguide Grating Router (AWGR) with a footprint of $\sim$1mm^2 [31] to provide contention-less point-to-point connectivity between all chiplets. AWGR is a *passive* SiPh fabric which provides all-to-all connectivity between any input and any output port. Several studies explored AWGRs as a uniquely compact solution for all-to-all interconnection with lower loss and crosstalk compared with other SiPh devices providing similar connectivity [14, 17, 40]. The reader can refer to the following articles for what concerns the physics, design principle, and scalability of AWGRs [41, 29, 15].

2.3 Packaging

Figure 3 presents an overview of the packaging approach we use in our design. We adopt a previously proposed technique for intra-package communication [38, 9,



(a) Top-view. (b) Side view.

Fig. 3: Example of proposed packaging solution, where Compute and Memory dies are optically-interconnected through an AWGR using SiPh transceivers with transceiver-chiplets and Si bridges on an organic substrate.

14] which can be applied to our memory controller design. This approach considers developing dedicated SiPh transceiver chiplets connected to their respective (compute or memory) dies.

The advantage of this design decision is that it can be leveraged to provide support for off-the-shelf memory devices (e.g., HBM, GDDR, etc.) by choosing the proper command scheduler in MMCs. By integrating the MMC and SiPh TRx (on the memory side) on the same die, no extra logic is required on memory dies, and MMCs can be designed to work with existing PHY interfaces - with minimal distance for data movements on electrical wires.

The dedicated SiPh transceiver chiplets connected to their respective dies on one side through Si bridges and to AWGR (the fabric providing all-to-all connectivity) through polymer waveguides (PWGs). These polymer waveguides are integrated on top of the organic package substrate and provide inter-chiplet optical connectivity. The reader can refer to the work of Dangel et al. [11, 12] for the details on the overall integration process for polymer waveguides.

Combining SiPh and Si bridges, our proposal utilizes each interconnection technology where it is the most efficient: SiPh for long-distance cross-package interconnect between chiplets and Si bridges for short-distance electrical interconnect between the TRXs and the memory controller.

SiPh manufacturing processes exploits well established CMOS processes, and photonic integrated circuit design kits (PDKs) have seen significant growth in the past ten years, resulting in cost-effective SiPh integration [39]. The reader can refer to [14, 20] for more detailed cost analyses and roadmap.

2.4 HTA Architecture

We discussed the challenges in scaling the memory architecture for today's high-throughput accelerator and how our proposed memory architecture addresses them. In this section we build on top of the proposed memory system, and introduce a high-throughput accelerator (HTA) architecture which takes the

advantage of low-latency all-to-all optical fabric and allows elimination of the shared last level cache.

Elimination of last-level caches provide significant advantages in terms of dedicating more area for compute, reducing access latency, and improving predictability in memory access time. The photonic interconnect used in our proposal provides us higher bandwidth at a lower energy per bit cost to make the underlying design tradeoffs such as eliminating the last level caches feasible, especially for irregular workloads with poor locality.

Implications on Core Architecture Memory accesses in GPUs takes hundreds of cycles to be serviced, and this latency can drastically change during the application execution as different compute units compete for receiving their data through shared memory channels. GPU architects have addressed this issue by increasing the number of contexts executed simultaneously on GPUs. However, this design choice comes with several challenges:

Context Scheduler: Allowing execution of multiple contexts at the same time requires dedicated logic to maintain, track, the state for each of them. Moreover, based on the state of contexts, additional logic is required to perform scheduling with proper arbitration and decoding units involved.

Physical Register Files: GPUs rely on large register files to store data required for computation. Providing support for tens of contexts to be executed simultaneously translates in larger register files, scaling almost linearly with the number of contexts supported.

Both area and power dedicated to the operations discussed above are obstacles towards achieving scalability for high-throughput accelerators. Our proposed memory architecture mitigates these overheads by lowering the access latency and improving the predictability in memory access. The evaluation of these opportunities for micro architectural improvements requires substantial work in terms of modeling, and we leave them for the future work.

Scalability of HTA One of the main benefits of SiPh interconnects is their distance-independent energy consumption and performance. Combining this with the benefits of packaging solution discussed in Section 2.3 allows HTA to scale.

Considering the area saving from eliminating L2 cache (occupying $\sim$50% of chip area), a single package instance of HTA can support 4$\times$ more compute units. Moreover, multiple packages can be utilized to scale further, and realize a scalable high-throughput accelerator with a unified address space without considerable energy and performance overheads.

The major component in HTA that needs to scale with the system is the AWGR. In this paper, we study HTA with 64 and 256 CUs which can be realized using 16×16 and 64×64 AWGR respectively. Scaling above 256 CUs requires AWGR with more than 64 ports. While 512×512 AWGR has been demonstrated [8], the main challenge for implementing AWGRs with high port counts (i.e., >64) is the optical crosstalk. However, the new Thin-CLOS architecture successfully demonstrated by Proietti et al. [29] can utilize multiples of

smaller AWGRs (lower port count) in parallel to provide the same functionality of a larger AWGR at lower crosstalk. While these solutions have larger footprints, the area overhead might be negligible in large accelerators with more than 256 CUs.

The bandwidth between any input-output pair in AWGR is limited to the information that can be carried out by a single modulated wavelength. If the bandwidth requirements exceed what a single wavelength offers, there are two alternative options. The first one is to leverage multiple free spectral ranges (FSRs) of an AWGR [16, 17], and virtually create a parallel channel of communication. The second one is to use spatial-division multiplexing (SDM), i.e., integrating and transmitting data through parallel AWGRs (either planar or 3D-stacked [32]). Multi-FSR strategy requires a broader laser spectrum and higher laser power to compensate for higher crosstalk inside the AWGR and to guarantee the required minimum optical power at the receiver. The SDM approach has similar laser power requirements but does not need a broader laser spectrum. However, it needs a larger die area or more SiPh layers, as well as more optical IO pins.

3 Methodology

3.1 System Comparisons

To evaluate our proposed HTA architecture, we compare it against a system similar to AMD's RDNA architecture with details of the memory hierarchy shown in Figure 1. CUs within a core cluster have private caches ("L0") and share the L1 cache, which centralizes all caching functions within each cluster [1]. L1 caches are connected to a globally shared L2 cache through a long-latency crossbar interconnect, resulting in ~100 cycles hit latency for L2 [21]. Therefore, for our simulations, we modelled the electrical crossbar with a latency of 50 cycles in each direction.

Within the memory controller of a given channel, all requests from different CUs share a read and a write queue. In each cycle, the scheduler performs an associative search and issues commands for requests in a First-Ready First-Come-First-Served (FR-FCFS [30]) fashion. For our evaluations, we refer to this design as the baseline memory controller. While we use AMD's RDNA memory hierarchy as our baseline, the challenges in scaling the memory hierarchy of GPUs are common in NVIDIA's systems and our proposal can be applied there similarly.

One example of HTA can host 64 CUs by utilizing a 16×16 AWGR to interconnect eight compute chiplets (each with four CUs) to four stacks of HBM2 memory. SiPh links use WDM with 16 wavelengths and perform modulation/demodulation at 32Gbps. On the compute side, each compute chiplet uses one SiPh WDM TRX with 64GB/s bandwidth in each direction, making a total of 16 SiPh TRXs for CMCs. On the memory side, four SiPh WDM TRXs can match the 256GB/s bandwidth of a single stack of HBM2 which results in a total of 16 SiPh TRXs for MMCs.

Table 1: Simulation Parameters

Compute Cluster			
Number of CUs	**64**	**CUs per CC**	**4**
Memory Hierarchy			
L0 V$	**16kB (per CU)**	**L0 I$**	**32kB (per CC)**
L0 K$	**16kB (per CC)**	**L1 $**	**64kB (per CC)**
L2 $	**2MB (8 banks)**	**DRAM**	**4GB HBM2 [22]**

3.2 Simulations

Performance To model our target systems we use MGPUSim [34] which models the Graphics Core Next 3 (GCN3) ISA. We extended the simulator to model a three level cache hierarchy. We integrated the timing model from DRAMSim3 [22] after extending it to model our proposed partitioned memory controller design discussed in Section 2. We utilize MGPUSim for collecting the traces on the memory system, and piped those traces on detailed timing model on DRAMSim.

For the performance of the interconnect technologies used in this paper, we used latency reporting in the previous work [21, 14]. The details of the modeled system in the simulator for different components are listed in Table 1. It should be noted that the trace-based evaluation approach limits our reporting to the performance of the memory system, and does not allow us to obtain execution times for the two systems under comparison. However, since a significant portion of the pipeline stalls are due to memory accesses, the performance of the memory system would be a reasonable candidate for our evaluation. To this end, we will look at the penalty of L1 misses when comparing the baseline with PMC in Section 4.

For evaluating our proposal we used benchmarks from AMD's Accelerated Parallel Processing (APP) Software Development Kit (SDK), Hetero-Mark suite [33], and Scalable Heterogeneous Computing (SHOC) suite [10].

Among those supported by MGPUSim, we chose different benchmarks with different memory behaviours to evaluate our proposal under different scenarios. Breadth-first Search *bfs* and Page Rank *pr* represent applications with irregular memory access patterns (i.e., poor locality). AES-256 Encryption (*aes*), Fast Fourier Transform (*fft*), and FIR Filter (*fir*) represent typical compute intense HPC applications with considerable amount of data reuse (i.e., medium locality). Simple Convolution (*conv*) implementation used for this work divides the image into sub-images to maximize data reuse (i.e., high locality).

4 Evaluation

In this section, we present the evaluation results on three aspects of our proposal.

First, we look at the performance of the proposed memory controller design compared to the baseline memory controller discussed in Section 2. This analysis is done under the same cache hierarchy. In these experiments, we look at the average DRAM access latency in both designs, as well as 95^{th} percentile latency as a measure of divergence in the access latency. Second, we evaluate HTA design against the baseline GPU architecture. In this set of analyses, we evaluate our memory controller design combined with a new cache hierarchy, and model a system like the one shown on right in Figure 1. We report the average miss penalty for L1 caches in the form of Average Memory Access Time (AMAT) for L1 misses. Third, we evaluate our proposal at scale by comparing the performance of HTA with 256 CUs against a multi-GPU system with 4× 64CU GPUs.

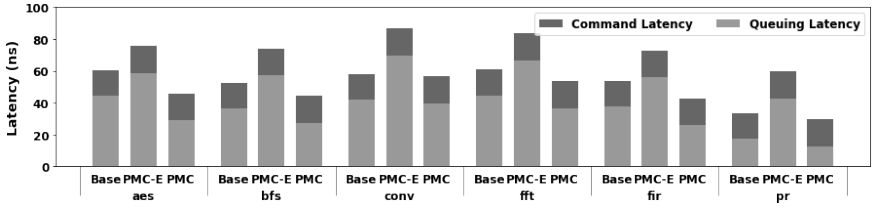
4.1 Evaluation of Partitioned Memory Controller

As our first step in evaluating our proposed architecture, we compare the performance of the partitioned memory controller against the baseline memory controller, both using the same cache organization. To emphasize on the importance of the enabling technology used in our design, an implementation of PMC using electrical links (PMC-E) is evaluated.

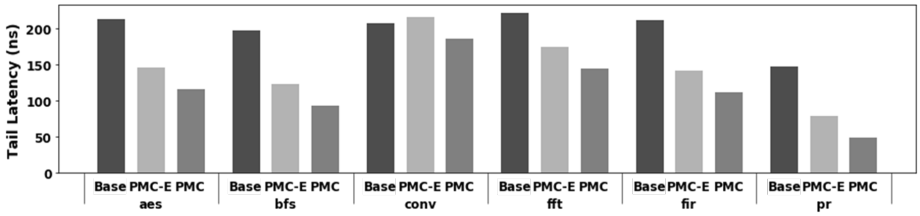
PMC design reduces access latency divergence by avoiding head-of-line blocking in scheduling. In the baseline design where all requesters share a single queue, if one requester sends a stream of requests over a short window (a common case in data-parallel accelerators), requests from other requesters are blocked until DRAM manages to return pending requests. PMC avoids this by having dedicated queues for each requester and directly applies the back-pressure to the original requester and not the whole system. Figure 4b shows the 95^{th} percentile in access latency, indicating a significant reduction in memory latency variation for PMC over the baseline memory controller. Depending on the access pattern in each workload, the 95^{th} percentile in access latency is improved by 10% to 60%. The benefits gained through scheduling are strong enough to result in improved tail latency even for the electrical implementation of the PMC which suffers from high-latency links.

Besides improving the predictability in access latency, PMC improves the access latency by increasing parallelism in bank accesses within the DRAM. Figure 4a depicts the average memory latency for the baseline memory controller and the proposed PMC. PMC achieves a lower average access latency by avoiding a portion of bank conflicts in the memory requests. If one requester sends several conflicting requests, those would limit bank activations in the baseline design, while in PMC, the scheduler can schedule requests from other requesters. Therefore, the queuing portion of memory access is reduced by 10% to 30% depending on the access pattern exhibited by each workload.

Both PMC-E and PMC take advantage of the scheduling scheme offered by PMC and avoid head-of-line blocking which translates to improvements in tail latency. This is purely due to the scheduling scheme in PMC, and it is independent of the technology used to implement the point-to-point fabric. However, as



(a) Average DRAM access latency



(b) 95^{th} percentile latency for DRAM access

Fig. 4: DRAM performance for the baseline memory controller (Base) compared to a system utilizing Partitioned Memory Controller with implemented with electrical and SiPh links (PMC-E and PMC, respectively). (a) In terms of access latency, PMC improves the queuing latency by 10% to 30% resulting in 5% to 26% reduction on overall access latency compared to the baseline memory controller. (b) The 95^{th} percentile latency for DRAM access is improved by 10% to 60% by reducing contention at read and write queues within the memory controller.

described in Section 2.1, the PMC design makes the crossbar latency part of the memory access. Therefore, the latency overhead imposed by the interconnect used in PMC is a critical part of this design. While the PMC design improves the average access latency by 10%-30% (*i.e.*, 5-20ns), these improvements can be masked when using a long-latency crossbar (*e.g.*, 50ns). As illustrated in Figure 4b, the implementation of PMC using electrical links (PMC-E) improves the tail latency. However, as shown in Figure 4a, the average access latency is significantly increased as the result of long-latency electrical links used in this design. This analysis shows the importance of interconnect technology used for our proposal, making SiPh and AWGR the key enablers for this design.

4.2 Evaluation of HTA

As the next step, we investigate the performance of proposed HTA system which allows for elimination of the last level cache against the baseline GPU described in Section 3, along with a GPU with 40MB of last level cache. In order to analyse different architectural differences between HTA and the baseline, we present evaluate two middle point between the two systems. First, we modeled a system similar to the baseline which utilizes the PMC under the same cache

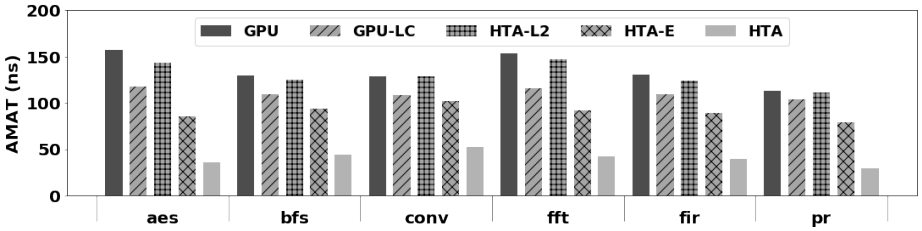


Fig. 5: Average Memory Access Time (AMAT) for L1 misses. The baseline (GPU) is compared to a GPU with 40MB of last level cache (GPU-LC), a similar system using PMC (HTA-L2), an implementation of HTA using electrocal links (HTA-E), and ultimately the proposed HTA. HTA improves the average L1 miss penalty by 2.3× to 5× compared to the baseline GPU architecture by avoiding data transfers over a high-latency crossbar.

organization (labeled HTA-L2). Moreover, we modeled HTA implemented using electrical interconnects to separate the architectural changes from the benefits gained purely from SiPh technology (labeled HTA-E).

As we discussed earlier in Section 3, our trace-based evaluation does not allow us to report runtime numbers. Thus, we choose to report the overall performance of the memory system. Figure 5 presents the L1 miss penalty, as a measure of performance of the memory system for both architectures under investigation. Average miss penalty for L1 caches is calculated in the form of AMAT for L1 misses.

As the third bar (HTA-L2) in Figure 5 shows, DRAM access latency improvements gained from PMC result in 10-15% reduction in L1 miss penalty. However, the latency-intensive (50 cycles) consult with the last level cache is hiding most of the benefits achieved. With L2 caches eliminated in HTA, all L1 misses are directly added to the CMCs, where requests are transferred over the all-to-all fabric to the MMCs.

Even the HTA system using electrical links (with 50 cycles of latency between CMCs and MMCs) significantly reduces the L1 miss penalty. Taking advantage of low-latency (3 cycles) interconnect fabric enabled by SiPh, HTA reduces the latency cost of L1 misses by 2.3x to 5x.

Reductions on the average miss penalty for L1 caches are mostly obtained through improvements on the 95^{th} percentile in access latency, emphasizing the importance of variations in memory access latency in the overall performance of the memory system for high-throughput accelerators.

The second bar (GPU-LC) represents a GPU with a large (*i.e.*, 40MB) last-level cache, similar to the architectural approach taken by NVIDIA [26], lowering the AMAT by reducing the traffic to DRAM. This approach benefits workloads with high locality. However, as can be seen in Figure 5, it will only achieve a small fraction of improvements offered by HTA for irregular HPC workloads with sparse data accesses.

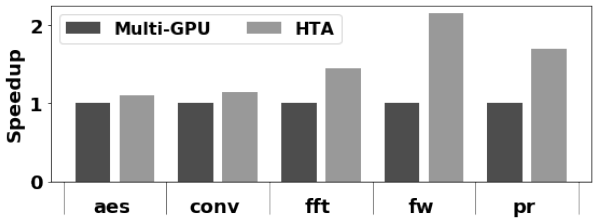


Fig. 6: The speedup of HTA with 256 CUs compared to a multi-GPU system with 4 GPUs each with 64 CUs. The overhead of data movements in multi-GPU setup result in a speedup of up to 2× for HTA.

4.3 Comparison with Multi-GPU systems

A key motivation for our HTA design is to achieve scalability. Utilizing a 64×64 AWGR, HTA can deliver an accelerator with 256 CUs. The state-of-the-art GPU systems can achieve this scale only by combining multiple GPUs.

For the last part in evaluating HTA, we compared its performance against a multi-GPU system with the same number of compute units (256 CUs). It should be noted that not all the benchmarks provided support for multi-GPU execution, and we only had a few options to run this experiment. Also, we should note that the speedups reported in Figure 6 are mainly a lower-bound for what the HTA can achieve. As of today, MGPUSim lacks a memory controller with timing details, and DRAM responses are satisfied at a flat latency. That is the main limiting factor for us to evaluate PMC in terms of execution time. However, to show the potential benefits of a scalable system enabled by HTA, we modeled a system with the average DRAM access latency measured in DRAMSim for the baseline controller and PMC. This approach does not take into account the benefits of lower variations in memory access achieved by PMC, and does not reveal the full performance potential of HTA.

According to the evaluation results shown in Figure 6, HTA can achieve 1.5× speedup on average compared to a multi-GPU system. This improvement is mainly achieved in HTA by avoiding the cross-GPU communication and scheduling overheads in a multi-GPU system.

One interesting observation here is the overhead of a multi-GPU system for different workloads. As can be seen in Figure 6, applications like *aes* or *conv* with smaller data sharing between their kernels experience less overhead ($\sim$10%) in the multi-GPU system. On the other hand, applications with more inter-kernel data dependencies such as Page Rank (*pr*), *fft*, and Floyd Warshal (*fw*) require more data movements between kernels (running on different GPUs), and result in larger slowdowns (up to 2×) in a multi-GPU setup. These variations depend on both architecture and workload, and impose several barriers in utilizing multi-GPU systems. HTA allows the programmers to migrate their applications to a scalable platform, and avoids considerable performance overheads especially for applications with significant data sharing across different compute units.

5 Related Work

Several studies have looked at the scalability of GPUs. Vijayaraghavan et al. illustrated the roadmap for exascale computing, and suggested aggressive use of chiplet technologies and die-stacking to meet a scalable system design [37]. MCM-GPU [3] argues that GPU scalability can be achieved by partitioning the GPU dies into GPU modules and reducing the cross GPU traffic. Pal et al. took a different approach, and looked at the design space of wafer-scale GPUs [28], where pre-manufactured GPU dies are directly bonded on to a silicon wafer which includes the interconnection fabric on it. Arunkumar et al. [4] created a framework for quantifying the scaling efficiency in terms of both performance and energy which, based on their analysis, lack of inter GPU module bandwidth increases the GPU idle time which increases the energy consumption in the system. As Arunkumar et al. pointed out, the performance and energy overheads of data movements are the main limiting factor towards scalability in GPUs. Many researchers looked at this problem from different viewpoints. Milic et al. [24] proposed a NUMA-aware multi-socket GPU architecture that reduces the traffic on the interconnects. They minimized the NUMA effects by dynamically optimizing the interconnect and the cache management policy in each phase of the application. In MGPUSIM [34] a new memory management policy is introduced in multi-GPU systems which can improve the data placement dynamically with the goal of reducing inter-node communications. We believe that the performance and energy overheads of data movements in GPUs should fundamentally be addressed. Considering how the memory system is designed for the state-of-the-art accelerators, this goal can only be achieved through co-designing the memory system and interconnect fabric. Another limiting factors in high-throughput accelerators is the memory access latency, both in terms of the absolute value and its variations. Lowering the memory access latency would decrease the idle time, improving performance and energy efficiency. Chatterjee et al. improved the performance of the memory systems in GPUs by proposing a new memory controller that can reduce the DRAM latency divergence within the warps [7]. Bojnordi et al. proposed a programmable memory controller along with added instructions to the ISA to improve request scheduling and bank utilization on DDR memories [6]. Hashemi et al. aimed to reduce the pressure on the memory system and proposed adding more logic to the memory controller to execute cache inefficient instructions near DRAM by dynamically identify such instructions at the processor [18]. Liu et al. improved the memory access mapping by using the window-based entropy mapping [23]. This technique reduces the virtual to physical address mapping overhead by quantifying the entropy of each address bit across all memory requests. Hussain et al. looked at the access pattern within irregular memory access, and reduced the DRAM latency by caching different patterns and scheduling memory accesses accordingly [19]. Oh et al. [27] improved the bandwidth utilization in HBM by load balancing across all channels, and decreased the stall time by effectively increasing the request queue. Tian et al. proposed an adaptive technique for bypassing caches [35] which can improve performance and energy efficiency in GPUs, especially for workloads

with poor cache utilization. We found all of the aforementioned related work on the memory controller design applicable to our design, providing several valuable pointers for the future directions.

6 Conclusion

In this paper, we proposed a novel partitioned memory controller (PMC) to reduce the contention in memory system of high-throughput accelerators. Utilizing the PMC design along with a scalable all-to-all optical fabric, we proposed a new high-throughput accelerator. Our simulation results show improvements for PMC on DRAM access latency and memory access divergence, and reduced miss penalty in L1 caches. Our chiplet-based design combines our novel PMC design and SiPh technology to support 4× more compute units.

Given the lack of publicly available area/power models of state-of-the-art GPUs, it is difficult to do a fair and accurate comparison of HTA with GPUs in terms of power and area. However, we can present a qualitative analysis. In terms of power consumption, SiPh links used in this work require 1.65-0.66 pJ/bit depending on the technology node used ranging from 65nm to 14nm. In terms of area overheads, PMC design does not add any logic for queuing as dedicated queues are result of breaking down the single shared queue in the baseline controller. Moreover, the SiPh components used in our design (the AWGR, and SiPh TRXs) have small footprints compared to size of the processor dies (less than 0.01% for typical compute dies [15]).

In this work we have assumed that the compute units in the HTA are similar to that of a GPU. However the proposed HTA architecture can apply to many different types of processors and accelerators. The combination of the significantly lower memory latency and more deterministic memory access time enables unexplored areas for micro-architecture design of advanced computing units and accelerators. This will form our future work.

References

1. AMD: Introducing rdna architecture. https://www.amd.com/system/files/documents/rdna-whitepaper.pdf (2019), [Online; accessed 12-10-2020]
2. AMD: Introducing amd cdna architecture. https://www.amd.com/system/files/documents/amd-cdna-whitepaper.pdf (2020), [Online; accessed 12-12-2020]
3. Arunkumar, A., et al.: Mcm-gpu: Multi-chip-module gpus for continued performance scalability. ACM SIGARCH Computer Architecture News **45**(2), 320–332 (2017)
4. Arunkumar, A., et al.: Understanding the future of energy efficiency in multi-module gpus. In: 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). pp. 519–532. IEEE (2019)
5. Bergman, K., et al.: Photonic network-on-chip design. Springer (2014)
6. Bojnordi, M.N., Ipek, E.: Pardis: A programmable memory controller for the ddrx interfacing standards. In: 2012 39th Annual International Symposium on Computer Architecture (ISCA). pp. 13–24 (2012)

7. Chatterjee, N., et al.: Managing dram latency divergence in irregular gpgpu applications. In: SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 128–139. IEEE (2014)
8. Cheung, S., et al.: Ultra-compact silicon photonic 512× 512 25 ghz arrayed waveguide grating router. IEEE Journal of Selected Topics in Quantum Electronics **20**(4), 310–316 (2013)
9. Cutress, I.: Intel launches stratix-10-tx leveraging emib with 58g transceivers. https://www.anandtech.com/show/12477/intel-launches-stratix-10-tx-leveraging-emib-with-58g-transceivers-, [Online; accessed 11-28-2020]
10. Danalis, A., et al.: The scalable heterogeneous computing (shoc) benchmark suite. In: Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units. pp. 63–74 (2010)
11. Dangel, R., et al.: Polymer waveguides for electro-optical integration in data centers and high-performance computers. Optics express **23**(4), 4736–4750 (2015)
12. Dangel, R., et al.: Polymer waveguides enabling scalable low-loss adiabatic optical coupling for silicon photonics. IEEE Journal of Selected Topics in Quantum Electronics **24**(4), 1–11 (2018)
13. Das, S.: It's time for disaggregated silicon! https://www.netronome.com/blog/its-time-disaggregated-silicon/ (2018), [Online; accessed 11-28-2020]
14. Fotouhi, P., et al.: Enabling scalable chiplet-based uniform memory architectures with silicon photonics. In: Proceedings of the International Symposium on Memory Systems. pp. 222–334 (2019)
15. Fotouhi, P., et al.: Enabling scalable disintegrated computing systems with awgr-based 2.5 d interconnection networks. IEEE/OSA Journal of Optical Communications and Networking **11**(7), 333–346 (2019)
16. Grani, P., et al.: Bit-parallel all-to-all and flexible awgr-based optical interconnects. In: Optical Fiber Communication Conference. pp. M3K–4. Optical Society of America (2017)
17. Grani, P., et al.: Design and evaluation of awgr-based photonic noc architectures for 2.5 d integrated high performance computing systems. In: 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA). pp. 289–300. IEEE (2017)
18. Hashemi, M., et al.: Accelerating dependent cache misses with an enhanced memory controller. In: 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA). pp. 444–455 (2016)
19. Hussain, T., et al.: Advanced pattern based memory controller for fpga based hpc applications. In: 2014 International Conference on High Performance Computing Simulation (HPCS). pp. 287–294 (2014)
20. Jeppix: Cost roadmap. https://www.jeppix.eu/wp-content/uploads/2020/04/JePPIXRoadmap2012.pdf, [Online; accessed 11-28-2020]
21. Jia, Z., et al.: Dissecting the nvidia volta gpu architecture via microbenchmarking. arXiv preprint arXiv:1804.06826 (2018)
22. Li, S., et al.: Dramsim3: a cycle-accurate, thermal-capable dram simulator. IEEE Computer Architecture Letters (2020)
23. Liu, Y., et al.: Get out of the valley: power-efficient address mapping for gpus. In: 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA). pp. 166–179. IEEE (2018)
24. Milic, U., et al.: Beyond the socket: Numa-aware gpus. In: Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture. pp. 123–135 (2017)

25. Miller, D.A.: Device requirements for optical interconnects to silicon chips. Proceedings of the IEEE **97**(7), 1166–1185 (2009)
26. NVIDIA: A100 tensor core gpu architecture. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf, [Online; accessed 11-31-2020]
27. Oh, B., et al.: A load balancing technique for memory channels. In: Proceedings of the International Symposium on Memory Systems. pp. 55–66 (2018)
28. Pal, S., et al.: Architecting waferscale processors - a gpu case study. In: 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). pp. 250–263 (2019)
29. Proietti, R., et al.: Experimental demonstration of a 64-port wavelength routing thin-clos system for data center switching architectures. Journal of Optical Communications and Networking **10**(7), B49–B57 (2018)
30. Rixner, S., et al.: Memory access scheduling. ACM SIGARCH Computer Architecture News **28**(2), 128–138 (2000)
31. Shang, K., et al.: Low-loss compact silicon nitride arrayed waveguide gratings for photonic integrated circuits. IEEE Photonics Journal **9**(5), 1–5 (2017)
32. Su, T., et al.: Interferometric imaging using si 3 n 4 photonic integrated circuits for a spider imager. Optics express **26**(10), 12801–12812 (2018)
33. Sun, Y., et al.: Hetero-mark, a benchmark suite for cpu-gpu collaborative computing. In: 2016 IEEE International Symposium on Workload Characterization (IISWC). pp. 1–10. IEEE (2016)
34. Sun, Y., et al.: Mgpusim: enabling multi-gpu performance modeling and optimization. In: Proceedings of the 46th International Symposium on Computer Architecture. pp. 197–209 (2019)
35. Tian, Y., et al.: Adaptive gpu cache bypassing. In: Proceedings of the 8th Workshop on General Purpose Processing using GPUS. pp. 25–35 (2015)
36. TSMC: Enhancing the cowos platform. https://pr.tsmc.com/english/news/2026 (2020), [Online; accessed 12-14-2020]
37. Vijayaraghavan, T., et al.: Design and analysis of an apu for exascale computing. In: 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA). pp. 85–96 (2017)
38. Wade, M., et al.: Teraphy: A chiplet technology for low-power, high-bandwidth in-package optical i/o. IEEE Micro **40**(2), 63–71 (2020)
39. Wang, J., Long, Y.: On-chip silicon photonic signaling and processing: a review. Science Bulletin (2018)
40. Werner, S., et al.: Towards energy-efficient high-throughput photonic nocs for 2.5 d integrated systems: A case for awgrs. In: 2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS). pp. 1–8. IEEE (2018)
41. Zhang, Y., et al.: Foundry-enabled scalable all-to-all optical interconnects using silicon nitride arrayed waveguide router interposers and silicon photonic transceivers. IEEE Journal of Selected Topics in Quantum Electronics **25**(5), 1–9 (2019)