Design and Analysis of an APU for Exascale Computing

Thiruvengadam Vijayaraghavan†*, Yasuko Eckert, Gabriel H. Loh, Michael J. Schulte, Mike Ignatowski,
Bradford M. Beckmann, William C. Brantley, Joseph L. Greathouse, Wei Huang, Arun Karunanithi, Onur Kayiran,
Mitesh Meswani, Indrani Paul, Matthew Poremba, Steven Raasch, Steven K. Reinhardt, Greg Sadowski, Vilas Sridharan
AMD Research, Advanced Micro Devices, Inc.
†*Department of Electrical and Computer Engineering, University of Wisconsin-Madison*

***Abstract*—The challenges to push computing to exaflop levels are difficult given desired targets for memory capacity, memory bandwidth, power efficiency, reliability, and cost. This paper presents a vision for an architecture that can be used to construct exascale systems. We describe a conceptual Exascale Node Architecture (ENA), which is the computational building block for an exascale supercomputer. The ENA consists of an *Exascale Heterogeneous Processor* (EHP) coupled with an advanced memory system. The EHP provides a high-performance accelerated processing unit (CPU+GPU), in-package high-bandwidth 3D memory, and aggressive use of die-stacking and chiplet technologies to meet the requirements for exascale computing in a balanced manner. We present initial experimental analysis to demonstrate the promise of our approach, and we discuss remaining open research challenges for the community.**

I. Introduction

High-performance computing (HPC) utilizes networks of thousands of highly capable machines to tackle some of the world's most complex problems. These problems drive discoveries and innovations critical to the quality of human life, ranging from designing more efficient fuels and engines to modeling global climate phenomenon. With continuous innovations in semiconductor logic and memory technologies, architectures, interconnects, system software, programming models, and applications, petaflop performance (10^{15} floating-point operations per second) was first reached in 2008 [1].

However, the need for ever more computational power continues to grow as science tries to answer bigger and deeper questions. Specifically, the amount of commercial and scientific data has exploded (accompanied with rapid increases in the complexity of the corresponding analytics), and the problems impacting human life have not necessarily diminished. To address these needs, supercomputers with exaflop (10^{18} or one billion, billion double-precision floating-point operations per second) capabilities may soon become necessary. Due to limitations in technology scaling [2], simply scaling up the hardware and software of the current petaflop architectures is not likely to suffice. We need innovations across many areas of computing to achieve exascale computing by the anticipated 2022-2023 time frame [3].

* This work was done while the author was with AMD Research.

The challenges associated with exascale computing, however, extend far beyond a certain floating-point throughput. An exascale supercomputer is envisioned to comprise of on the order of 100,000 interconnected servers or nodes in a target power envelope of ∼20MW, with sufficient memory bandwidth to feed the massive compute throughput, sufficient memory capacity to execute meaningful problem sizes, and with user intervention due to hardware or system faults limited to the order of a week or more on average [3]. These system-level requirements imply that each node delivers greater than 10 teraflops with less than 200W. There is a 7× gap in flops per Watt between the current most energy-efficient supercomputer [4] and the exascale target.

Although there are potentially many paths to realize exascale computing [5]–[7], this paper presents one vision for how to construct an exascale machine using advances in key technologies including low-power and high-performance CPU cores, integrated energy-efficient GPU compute units (CUs), in-package high-bandwidth 3D memory, aggressive use of die-stacking and chiplet technologies, and advanced memory systems. Our coordinated research that spans from circuits to software enables a very efficient and tightly integrated processor architecture suitable for exascale systems.

We introduce an *Exascale Node Architecture* (ENA) as the primary building block for exascale machines. The ENA addresses the exascale-computing requirements through:

- A high-performance accelerated processing unit (APU) that integrates high-throughput GPUs with excellent energy efficiency required for exascale levels of computation, tightly coupled with high-performance multi-core CPUs for serial or irregular code sections and legacy applications
- Aggressive use of die-stacking capabilities that enable dense component integration to reduce data-movement overheads and enhance power efficiency
- A chiplet-based approach [8], [9] that decouples performance-critical processing components (e.g., CPUs and GPUs) from components that do not scale well with technology (e.g., analog components), allowing fabrication in disparate, individually optimized process technologies for cost reduction and design reuse in other market segments
- Multi-level memories that enhance memory bandwidth

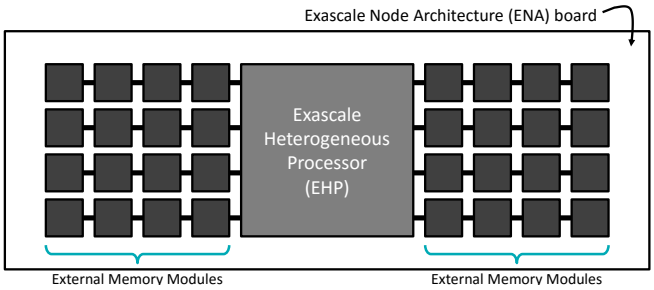

Figure 1. Exascale Node Architecture (ENA)

with in-package 3D memory, which is stacked directly above high-bandwidth-consuming GPUs, while provisioning high-capacity memory outside of the package

- Advanced circuit techniques and active power-management techniques which yield energy reductions with little performance impact
- Hardware and software mechanisms to achieve high resilience and reliability with minimal impact on performance and energy efficiency
- Concurrency frameworks that leverage the Heterogeneous System Architecture (HSA) [10] and Radeon Open Compute platform (ROCm) [11] software ecosystem to support new and existing applications with high-performance and high programmer productivity

While ongoing hardware and software research continues to further improve specific aspects of the ENA, this paper focuses on the overall physical node organization that integrates these aggressive technologies. We describe the proposed ENA concept and the rationale behind the different design decisions. We then present an initial quantitative evaluation of the ENA and discuss remaining challenges and opportunities for further research.

II. Exascale Node Architecture (ENA)

One possible exascale machine consists of 100,000 high-performance computing nodes. Supercomputers also require other components, such as the global interconnection network, I/O nodes for check-pointing and data storage, system and job management components, power delivery and cooling, and more. However, the focus of the present work is on the compute nodes that provide the primary computational horsepower for the overall exascale system.

The Exascale Node Architecture (ENA) is the building block for a single compute node (shown in Fig. 1). The ENA's computational capabilities are provided by a high-performance accelerated processing unit called the Exascale Heterogeneous Processor (EHP), which we will later describe in greater detail. The EHP makes use of high-bandwidth, in-package DRAM, but it is also coupled with a network of external-memory devices to provide large per-node memory capacity to handle very large application problem sizes. Given the performance goal of 1 exaflop and a power budget of 20MW for a 100,000-node exascale machine, we need to architect the ENA node to provide 10 teraflops of performance in a 200W power envelope. Below, we first describe the EHP and its constituent technologies, and then we discuss the memory system for the ENA.

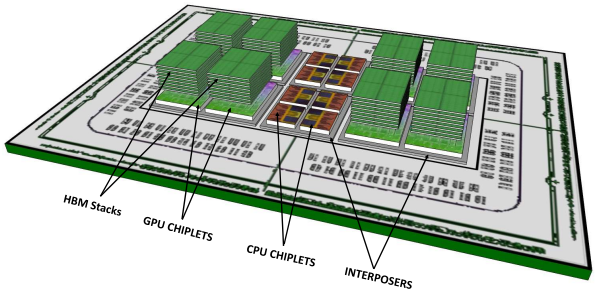

Figure 2. Exascale Heterogeneous Processor (EHP)

A. The EHP Architecture

The EHP is an accelerated processing unit (APU) consisting of a balanced mix of CPU and GPU compute resources integrated with in-package 3D DRAM. The overall structure makes use of a modular "chiplet" design, with the chiplets 3D-stacked on other "active interposer" chips. The aggressive use of advanced packaging technologies enables a large amount of computational and memory resources to be located in a single package.

Fig. 2 shows a conceptual rendering of the EHP. The EHP consists of two primary types of resources. In the center of the EHP are two CPU clusters, each consisting of four multi-core CPU chiplets stacked on an active interposer base die. On either side of the CPU clusters are a total of four GPU clusters, each consisting of two GPU chiplets on a respective active interposer. Upon each GPU chiplet is a 3D stack of DRAM (e.g., some future generation of JEDEC high-bandwidth memory (HBM) [12]). The DRAM is directly stacked on the GPU chiplets to maximize bandwidth (the GPUs are expected to provide the peak computational throughput) while minimizing memory-related data movement energy and total package footprint. CPU computations tend to be more latency sensitive, and so the central placement of the CPU cores reduces NUMA-like effects by keeping the CPU-to-DRAM distance relatively uniform. The interposers underneath the chiplets provide the interconnection network between the chiplets (i.e., network on chip (NOC)) along with other common system functions (e.g., external I/O interfaces, power distribution, system management). Interposers maintain high-bandwidth connectivity among themselves by utilizing wide, short-distance, point-to-point paths.

1) Integrated CPU+GPU (APU): Given the combination of parallel and serial regions in scientific applications, we believe that an APU architecture consisting of integrated

CPUs and GPUs is well-suited for the performance and power goals of the ENA.

The EHP uses eight GPU chiplets. Our initial configuration provisions 32 CUs per chiplet. Each chiplet is projected to provide two teraflops of double-precision computation, for a total of 16 teraflops. Based on the projected system size of 100,000 nodes, this would provide a total of 1.6 exaflops (we over-provision because real applications do not achieve 100% utilization).

The EHP also employs eight CPU chiplets (four cores each), for a total of 32 cores, with greater parallelism through optional simultaneous multi-threading. The number of CPU cores was carefully chosen to provision enough single-thread performance for irregular code sections and legacy applications without exceeding the package's area budget.

The CPU and GPU must be architected in a cohesive manner from an energy efficiency as well as programmability standpoint. To this end, Heterogeneous System Architecture (HSA) compatibility is one of the major design goals of the APU. HSA provides a system architecture where all computing elements (CPU, GPU, and possibly other accelerators) share a unified coherent virtual address space. This enables efficient programming and computation via several mechanisms like free exchange of pointers by both CPU and GPU code, eliminating expensive data copy operations, transparent management of CPU and GPU caches via cache coherence, task offloads by both CPU and GPU to each other or other CPU/GPU units [13], and efficient synchronization mechanisms. These features are supported by AMD's Radeon Open Compute platform (ROCm) to improve the programmability of such heterogeneous systems.

While APUs with HSA compatibility have been available for some time, the existing designs have been targeted more for consumer platforms rather than HPC systems. To this end, we are actively researching several mechanisms to provide heterogeneous computing for HPC. We have created novel mechanisms like the QuickRelease synchronization mechanism [14] and heterogeneous race free memory models (HRF) [15]–[17] to reduce synchronization overhead between GPU threads, and heterogeneous system coherence (HSC) [18] to transparently manage coherence between CPU and GPU caches. These make the APU easier to program and reason about for the application programmers, enabling more codes to be more easily ported to take advantage of the EHP's resources.

2) Modular Chiplet Design: The performance requirements of the exascale node require a large amount of compute and memory to be integrated into a single package. Rather than build a single, monolithic system on chip (SOC), we propose to leverage advanced die-stacking technologies to decompose the EHP into smaller components consisting of active interposers and chiplets. Each chiplet houses either multiple GPU compute units or CPU cores. The chiplet approach differs from conventional multi-chip module (MCM) designs in that each individual chiplet is *not* a complete SOC. For example, the CPU chiplet contains CPU cores and caches, but lacks memory interfaces and external I/O.[1] There are multiple benefits to this decompositional approach to SOC construction:

Die Yield: Building a single monolithic SOC that provides the equivalent capabilities of what we propose for the EHP would result in an impractically large chip with prohibitive costs. Smaller chiplets have higher yield rates due to their size, and when combined with known-good-die (KGD) testing techniques, can be assembled into larger systems at reasonable cost. This approach has already started garnering interest in both academia [9] and industry [8], [19].

Process Optimization: A monolithic SOC imposes a single process technology choice on all components in the system. With chiplets and interposers, each discrete piece of silicon can be optimized for its own functions. For example, the CPU chiplets can use performance-optimized devices and metal layers, while the GPU chiplets use density-optimized devices and metal. The interposer layers can use a more mature (i.e., less expensive) process technology node as the I/O components likely do not need transistors in the cutting-edge technology node nor as many metal routing layers as the compute chiplets.

Re-usability: A single, large HPC-optimized APU would be great for HPC markets, but may be less appropriate for others. The decomposition of the EHP into smaller pieces enables silicon-level reuse. For example, one or more of the CPU clusters could be packaged together to create a conventional CPU-only server processor.

3) Active Interposers: There are several options for the assembly of multiple discrete pieces of silicon within the same package. For example, MCMs have been used for years, and they have even been proposed in the context of chiplets as advocated by Marvell's MoChi (Modular Chips) concept [8]. *Passive* silicon interposers are also already in volume production, in particular for the integration of GPU and 3D memory as demonstrated by AMD's Radeon™ R9 Fury GPU [20]. However, due to the sheer amount of compute, memory, and other logic that we aim to integrate into the EHP package, we found true 3D stacking on top of active interposers to be desirable. Active interposers are fundamentally no different than other 3D stacking approaches considered by past research[2], and preliminary prototypes have even been successfully constructed and demonstrated [21].

[1] Contrast this to typical MCM parts that take identical CPU chips that are each fully functional SOCs, and then place them in the same package. With chiplets, there is no option to take a single chiplet by itself and convert it into a complete product without additional silicon.

[2] The primary point of differentiation, which is not a fundamental issue, is that most past 3D stacking research considered vertical stacks of identical/similar-sized chips, whereas active interposer configurations involve different-sized chips (i.e., the interposer vs. the chiplets) and some chips may also be stacked side by side (e.g., neighboring chiplets) rather than in a strictly vertical arrangement.

4) Power Optimizations: For a 100,000-node system, the ENA must provide (at least) 10 teraflops of compute with a relatively meager power budget of 200W. This includes not just power spent on computation, but also on the memory system, power delivery losses, cooling, and more. Our projections indicate that even an aggressive combination of state-of-art microarchitectures, SOC designs, memories, technology scaling, and traditional techniques like DVFS will not be enough to satisfy the ENA performance targets while staying within the power budget.

To this end, we have also investigated a wide range of power-optimization techniques. This includes circuit techniques such as near-threshold computing, asynchronous circuits, low-power interconnects [22], and data-compression circuitry. We combine these circuit-level mechanisms with high-level power-management techniques such as active DVFS [23] and power-gating [24] controls. We detail a selection of these techniques in Section V-E.

In addition, due to vertical integration of the DRAM and the GPU compute units, thermal feasibility is an important design consideration. In Section V-D, we study the thermal feasibility of our envisioned approach.

5) RAS Support: A small per-node probability of failure multiplied over a 100,000-node system substantially limits the overall system's mean-time-to-failure (MTTF). Shrinking process technology, increased transistor counts, and increased memory capacity all contribute to increases in transient faults. Given the aggressive technology node target and the scale of the exascale machine, resiliency, availability, and serviceability (RAS) are first-class design constraints. There are two main goals of a good RAS solution: minimize the rate of silent errors, and minimize the impact on performance, power, area, and cost.

The current state of the art for handling transient faults is error-correcting codes (ECC). ECC can be relatively easily deployed for regular structures like DRAM and SRAM arrays, but it still comes with area costs that are more challenging in our space-constrained EHP design. Also, some components may not normally have HPC-class RAS support by default. For example, current GPUs are still overwhelmingly used for graphics and visualization applications where a few errors result in unnoticed aberrations of a couple pixels, and so heavy-duty RAS support is unwarranted. To avoid burdening the GPU with excessive RAS features, which would reduce the re-usability of GPU chiplets in other markets, we have explored software-based solutions such as redundant multi-threading (RMT) [25]. RMT takes advantage of the fact that the GPU is not fully utilized for some workloads, and uses the otherwise idle GPU resources to redundantly perform computations and detect errors (note that this paper does not include quantitative evaluation of RMT).

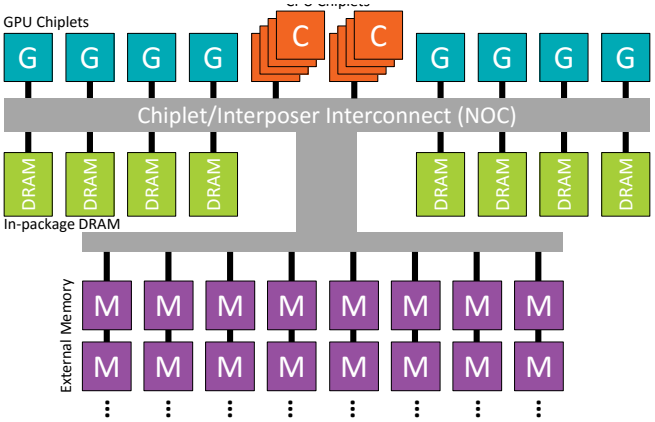

Figure 3. Block diagram of the ENA memory system

B. The ENA Memory System

The exascale memory requirements are very aggressive in terms of total memory capacity, the desired memory bandwidth, and the energy-related costs. Ideally, the entire system would make use of in-package 3D-integrated DRAM, as that provides the highest performance and capacity at the lowest energy cost. Unfortunately, our projections for the maximum amount of DRAM that we can integrate into a package fall short of the per-node memory targets. We instead make use of a heterogeneous memory architecture consisting of multiple levels of memory [26], [27]. Fig. 3 shows a conceptual view of the EHP's memory hierarchy.

1) Integrated 3D Memory: Many of the scientific workloads that we have analyzed demand more bandwidth than what can be provided by conventional DRAM outside of the processor package. The stated exascale targets [28] for memory bandwidth and energy efficiency are incredibly challenging for off-package memory solutions. Hence, we integrate 3D-stacked DRAM into the EHP package.

To estimate the memory capabilities of 3D DRAM in the exascale timeframe, we started with current JEDEC High-Bandwidth Memory (HBM) [12] and project forward. First-generation HBM provides 1GB capacity at 128 GB/s of bandwidth per stack, and second-generation HBM is projected to yield 8GB at 256 GB/s per stack. By the 2022-2023 time-frame, we estimate that HBM (or a similar equivalent) should have advanced at least two more generations. Assuming a doubling of capacity for each generation, we project that 3D DRAM in the exascale time-frame would provide 32GB per stack. As the interface speed for HBM2 is already 2 Gbps, we do not expect the same per-generation doubling of bandwidth; we instead assume only a single doubling (through wider interfaces or faster speeds). The EHP makes use of a total of eight 3D DRAM stacks (one per GPU chiplet), so the total 3D DRAM capacity is 256GB with a total aggregate bandwidth of 4 TB/s, meeting the exascale memory bandwidth target.

2) External-memory Network: The exascale target for per-node memory capacity is *at least* 1TB, which exceeds our in-package projections by a factor of four. As such, the ENA must augment the in-package memory with additional external memory. The ENA makes use of Memory Networks that consist of multiple memory modules interconnected with point-to-point links (as opposed to bus-based topologies for DDR) [29], [30]. A current example of this kind of memory approach is the Hybrid Memory Cube, which uses 3D-stacked DRAM inside each module, and the modules communicate with high-speed serial interfaces [31].

The envisioned EHP package provides eight external-memory interfaces, each of which connects to multiple memory packages. Fig. 3 shows each interface supporting a disjoint set of memory devices. A simple chain topology is shown, although support for other topologies is also expected. We assume that the memory interfaces are address-interleaved in some fashion such that there is never a need for one memory interface to send a request to a memory package connected to a different interface (similar to how multiple DDR memory channels are managed today). Optional links (not shown) could be used to cross-connect chains for redundancy purposes, which allow access to memory devices in the event of link failures.

Depending on the exact needs of the supercomputer customer, the external-memory network could consist of a mix of both DRAM and non-volatile memory (NVM) devices. NVM provides higher densities and therefore could be useful in scenarios where very large problem sizes are important, or they could be used to reduce the total ENA board-level component count by meeting a given capacity target with fewer higher-capacity packages. The trade-off is that the NVMs are typically slower, consume more dynamic power (especially for writes), and may suffer from write-endurance issues that could impact the system's MTTF.

3) Memory-system Management: The ENA's memory architecture provides (at least) two different levels of memory. There is the in-package 3D DRAM, and there is also the external-memory network (which could provide more than one additional level of memory if both DRAM and NVM modules are used). Such a heterogeneous memory system raises the question of how the different memory resources should be managed. While the ENA is designed to support a configurable memory system with multiple modes of operation, we envision the primary mode to be software-controlled. The ENA's physical memory address space is interleaved across the different memory resources (the granularity and overall mapping of which would be controlled by the system software). The operating system (OS) would then provide capabilities to monitor and migrate memory between memory resources in an attempt to maximize the fraction of memory requests that can be serviced by the in-package DRAM [26], [27], [32]. The system would also provide user-level APIs so that programmers could explicitly allocate data into specific types of memory [26], [33].

There has been a significant amount of work in recent years exploring how to make use of 3D DRAM as a large, hardware-managed cache [34], [35]. We envision the ENA supporting 3D-DRAM as a cache, but for HPC applications, this will often not be desirable. For an example ENA with 256GB of in-package DRAM and 1TB of external memory, using the in-package DRAM as a cache would sacrifice 20% of the system's total addressable memory capacity, which in turn reduces the problem sizes that can be solved. However, if a problem does not require the full memory capacity of the ENA, enabling the hardware-cache mode could provide a performance uplift without any application modifications.

III. Methodology

Traditional cycle-level simulators like gem5 [36] and GPGPU-Sim [37] may not be suitable for exploring the large and complex design space of the EHP (e.g., heterogeneous processors, 3D die stacking, multiple memory levels). Their execution times make it difficult to quickly gather first-order insights across a wide range of design candidates. Instead, we leverage an in-house high-level simulator for design-space exploration and use cycle-level simulation to account for the impacts of key microarchitecture choices.

Our high-level simulator [38] measures an application's execution on current hardware and uses a variety of scaling models to estimate the same application's performance and power on future hardware designs. For instance, we designed analytic performance [39] and power [40] scaling models for CPUs and we use data-movement performance counters combined with distance-based energy values for interconnect power calculations [41]. We use in-house technology-scaling models to estimate how power and our voltage-frequency curves will change.

We use machine-learning models to estimate how measured GPU power and performance will scale as the hardware configurations change [42]. These models are trained on numerous applications on a wide range of existing hardware configuration points [43].

These CPU and GPU models are effective for estimating the impact of high-level hardware-resource changes, but they assume that the underlying CPU and GPU microarchitectures remain the same as the hardware on which the initial measurements were taken. We use the AMD gem5 APU simulator [44] to account for differences from our EHP design (e.g., multi-chiplet organization) and adjust the high-level simulation results accordingly.

IV. Application Characterization

Table I lists six open-source scientific and security-related proxy applications [45] that we study, along with a highly compute-intensive application that is designed to measure the maximum achievable floating-point throughput [46]. We characterize the application kernels into three categories:

Table I
APPLICATION DESCRIPTIONS

Category	Application	Description
Compute Intensive	*MaxFlops*	Measures maximum FP throughput
Balanced	*CoMD* *CoMD-LJ*	Molecular-dynamics algorithms (Embedded Atom, Lennard-Jones)
	HPGMG	Ranks HPC systems
Memory Intensive	*LULESH*	Hydrodynamic simulation
	MiniAMR	3D stencil computation with adaptive mesh refinement
	XSBench	Monte Carlo particle transport simulation
	SNAP	Discrete ordinates neutral particle transport application

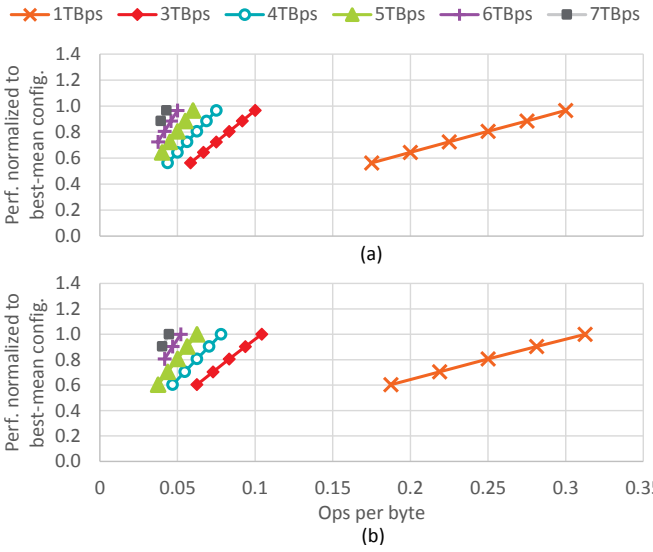

Figure 4. Performance of *MaxFlops* as we vary the bandwidth and (a) CU frequency or (b) CU count

compute intensive, balanced, and memory intensive[3]. We describe each category in detail below using simulated data.

In all figures in this section, the x-axis corresponds to the hardware computation capability with respect to the memory (ops-per-byte), which we compute as the product of compute unit (CU) count and GPU frequency, divided by memory bandwidth. We vary ops-per-byte by changing the bandwidth, CU count, and frequency.

A. Compute-intensive Kernels

Compute-intensive kernels have infrequent main-memory accesses, and the performance is bound by compute throughput. As such, these kernels benefit from higher CU counts and GPU frequencies, but they are relatively insensitive to memory bandwidth. In fact, in a power-constrained system like exascale supercomputers, provisioning higher bandwidth can be detrimental to the overall performance because that simply takes power away from the compute resources.

MaxFlops falls under this category, which is a highly compute-intensive kernel as shown in Fig. 4. While the performance increases linearly with more CUs and frequency (i.e., each bandwidth curve increases with higher ops-per-byte), bandwidth does not help (i.e., the corresponding CU-frequency points across different bandwidth curves have roughly the same performance level).

[3]The applications consist of multiple kernels, but we only report data for the most dominant kernel unless otherwise noted.

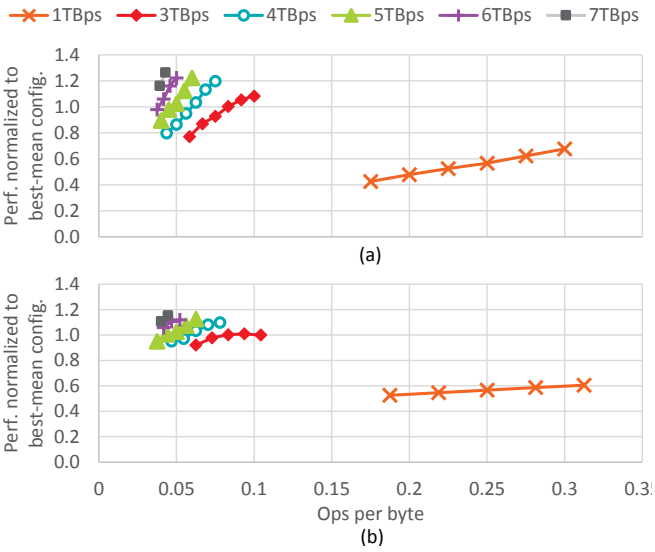

Figure 5. Performance of *CoMD* as we vary the bandwidth and (a) CU frequency or (b) CU count

B. Balanced Kernels

Balanced kernels, such as *CoMD* shown in Fig. 5, stress both the compute and memory resources. The best performance is observed when all resources are increased together. However, the rate of performance increase plateaus beyond a certain point. It is important to note that the plateau point is different across kernels.

C. Memory-intensive Kernels

Memory-intensive kernels, such as *LULESH* shown in Fig. 6, issue a high rate of memory accesses, hence are sensitive to the memory bandwidth. A notable characteristic of this class of kernels is that more CUs and higher GPU frequency are beneficial only up to a certain point. After that, the excessive number of concurrent memory requests starts to thrash the caches and increases contention in the memory and interconnect network, resulting in performance degradation.

V. ARCHITECTURE ANALYSIS

This section quantitatively evaluates the ENA's design choices described in Section II. We use a range of HPC applications that exercise various components of the architecture differently. Our analysis of over a thousand different hardware configurations found that utilizing a total of 320 CUs at 1 GHz with 3 TB/s of memory bandwidth achieves the best performance (when considering an average across all applications) under the ENA-node power budget

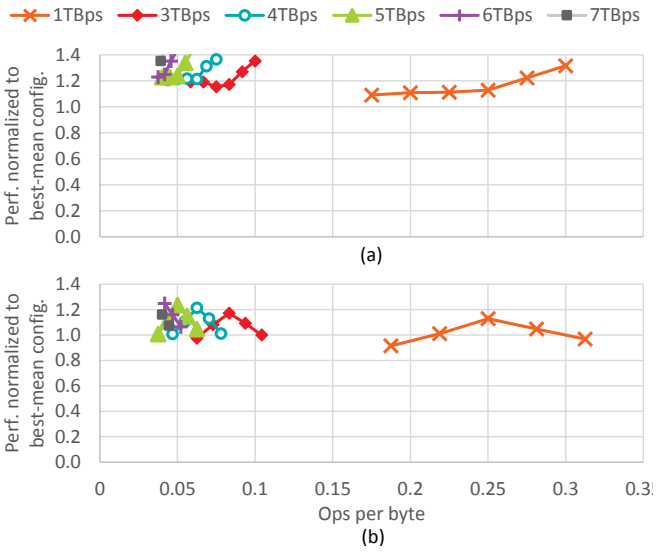

Figure 6. Performance of *LULESH* as we vary the bandwidth and (a) CU frequency or (b) CU count

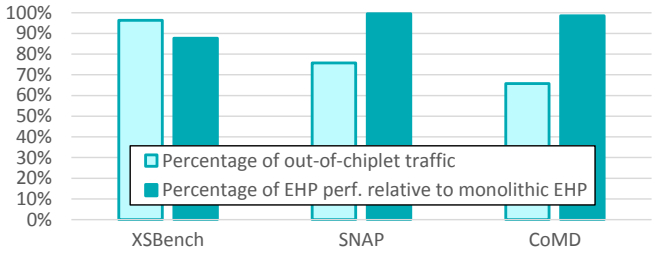

Figure 7. Out-of-chiplet traffic and impact on performance

of 160W[4] and area constraints (more details in Section VI). All the results presented below use this configuration unless otherwise noted.

A. Chiplet Organization

The benefits of a chiplet-based design were covered in Section II, but a multi-chip organization imposes additional latency for inter-chiplet communication (whether for coherence or main-memory access). Messages to a remote chiplet traverse from a source chiplet to the lower interposer layer through through-silicon vias (TSVs), go across the interposer, and go up to the destination chiplet through TSVs [9]. Thus, two extra vertical-communication hops are required compared to a hypothetical monolithic EHP. To assess the performance impact of these overheads, we compare a monolithic EHP design against our proposed multi-chiplet design. The findings below show only a small performance impact, suggesting that a chiplet-based design is feasible.

Finding 1: The out-of-chiplet traffic dominates the total traffic, ranging from 60-95% across kernels representing a wide range of behavior. This traffic includes traffic between

[4]We set the per-node power budget to 160W to leave enough power for cooling, inter-node network, etc. so that the total system-wide power would not exceed 20MW.

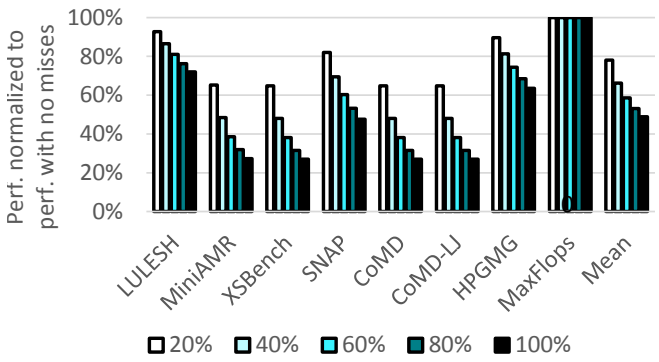

Figure 8. Performance impact of miss rates in the in-package DRAM

GPU CUs on different chiplets as well as between the CPU cores and the GPU CUs. Given that there are eight GPU chiplets on the EHP, the remote-traffic rate demonstrates a fairly even distribution of accesses across chiplets.

Finding 2: The performance impact of the large out-of-chiplet traffic is relatively small. The largest performance degradation compared to the monolithic EHP is 13%, while some application kernels, such as *SNAP*, have a negligible impact, as shown in Fig. 7.

Takeaway: The small performance impact despite the large out-of-chiplet traffic is due to a variety of reasons. First, the GPU's massive parallelism is effective at latency hiding. Kernels with abundant parallelism tend to be able to hide much of the latency to other chiplets. Second, some applications (especially those with cache-friendly access patterns) do not stress the memory system and are not sensitive to a few cycles of additional memory latency. Thus, our chiplet organization makes a compelling trade-off between longer network latency and improvement in die yield and cost.

B. In-package DRAM

Our design choice of integrating high-bandwidth 3D memory directly into the EHP package is driven by the exascale high-bandwidth requirement. Due to the limited capacity of the in-package 3D-DRAM, some data may reside in the external memory instead. Requests to addresses not in the first-level in-package memory result in costlier accesses to the external memory through a lower-bandwidth, off-package interface that can degrade performance. To highlight the importance of provisioning high-bandwidth memory closer to the compute and efficient data management for the multi-level memory, we artificially vary the fraction of requests serviced by the in-package DRAM.

Finding: Fig. 8 plots the performance reduction due to in-package 3D-DRAM misses[5] normalized to no misses (i.e., all requests serviced by in-package DRAM). As expected, the degree of reduction depends on the application characteristics. The compute-intensive application *MaxFlops* retains the same performance level regardless of the in-package

[5]Here, we use the term "miss" to describe a memory access to the external memory. We are *not* using the in-package DRAM as a hardware-controlled cache.

memory miss rate due to very infrequent memory accesses. In contrast, other applications observe degradations from 7% to as much as 75% as more misses in the in-package memory cause more external-memory traffic. *LULESH*, a memory-intensive kernel, has an interesting behavior as it shows lower sensitivity to bandwidth compared to a compute-intensive kernel like *CoMD*. This can be attributed to the *LULESH*'s irregular access patterns that make it more sensitive to memory latency than bandwidth. Separate analysis of the applications indicate that 46% to 89% of memory traffic may still need to access off-package memory, mainly due to large data footprints. These rates are derived from previous memory management techniques [27], and will likely be improved as research discovers better multi-level memory management algorithms.

Takeaway: The in-package memory is critical for many bandwidth-intensive kernels. Efficient data management via software and/or hardware techniques will continue to be important to ensure that as many requests as possible can be serviced from the in-package memory for both performance and energy concerns.

C. External-memory Configuration

Although high-density NVM is a potential alternative for configuring the external memory, one of the disadvantages is high read and write access energy. In this section, we compare the power of the baseline configuration with a DRAM-only external-memory system against a hybrid configuration that replaces half of the external DRAM with NVM while maintaining the same total capacity[6].

Fig. 9 shows a breakdown of the total ENA power for both external-memory configurations. Note that many of the simulated power components (e.g., interconnects and in-package memory) have been combined into a single category ('Other') for readability. '(S)' refers to static power and '(D)' refers to dynamic power. Serializer/Deserializer (SerDes) links are used to connect the external-memory devices.

Finding 1: The external-memory power (sum of 'External memory' and 'SerDes' for both static and dynamic power) ranges from 40W to 70W across all kernels and configurations. For the DRAM-only configuration, static/background power is the major contributor to the external power: 27W from DRAM static/refresh power and 10W from the SerDes background power.

Finding 2: The hybrid DRAM+NVM configuration cuts the static power of the external memory (sum of 'External memory (S)' and 'SerDes (S)') by about one half due to negligible static power of NVM and fewer SerDes links. This reduces the total ENA power of the less memory-intensive applications (*CoMD*, *CoMD-LJ*, and *MaxFlops*). However, NVM's high dynamic memory-access energy more than offsets the low static power for applications that frequently access the external memory (*LULESH*, *MiniAMR*, *XSBench*, *SNAP*, and *HPGMG*). In these scenarios, the total power of three of the applications increase by as much as 2×.

[6]The per-module capacity of NVM is assumed to be 4× that of DRAM.

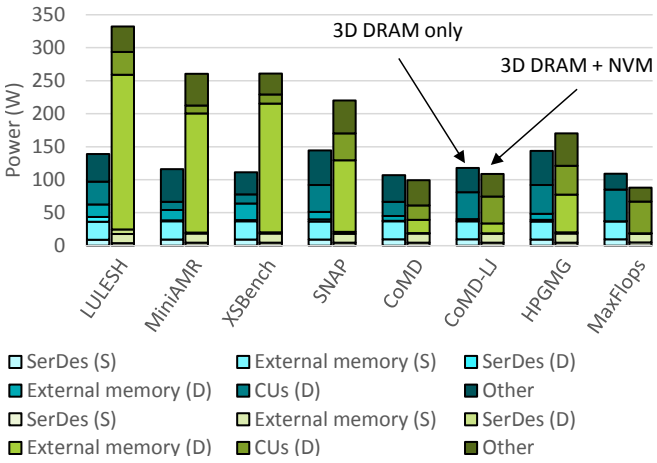

Figure 9. Impact of external-memory configurations on ENA power

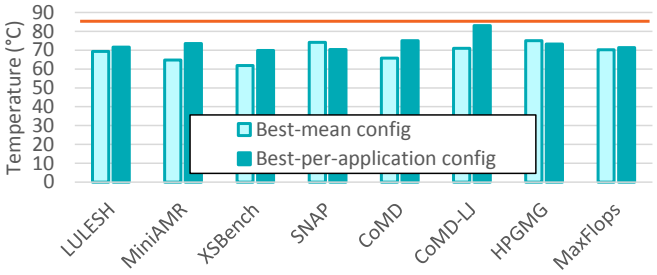

Figure 10. Peak in-package 3D-DRAM temperature

Takeaway: Application characteristics need to be considered carefully when configuring the external memory. Despite NVM's high densities, its high dynamic energy may limit the number of NVMs to provision in a power-constrained system. Note, however, that the numbers and types of external-memory modules is a design parameter that end customer can customize to their needs, and the overall ENA does not force any specific configuration.

D. Thermal Assessment

We simulate the EHP package alone as it has higher power density than the external memory. We used the HotSpot [47] model from the University of Virginia and calibrated it against an internal product model.

We use peak DRAM temperature as a metric in this analysis because DRAM is less tolerant to temperature than processors. DRAMs must stay below 85°C to avoid increasing the refresh rate [48]. We assume a high-end air-cooling solution [49] and 50°C ambient temperature in a 2U-server chassis [50].

Finding 1: As shown in Fig. 10, EHP's in-package DRAMs stay below the 85°C limit for all kernels with both the baseline EHP configuration and the best per-kernel config-

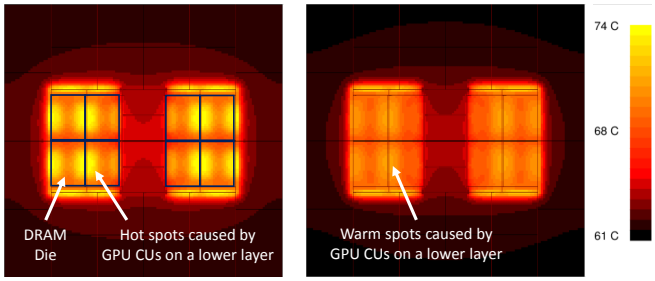

Figure 11. Heat map of the bottom-most in-package 3D-DRAM die for SNAP

uration[7]. Despite the high CU power, *MaxFlops* does not stress the memory temperature because of almost no memory accesses. On the other hand, *CoMD-LJ* approaches the thermal limit due to its modest memory accesses combined with high compute intensity that raises the CU dynamic power in the kernel-optimized EHP configuration. Note that these results do not include any power optimizations discussed later in Section V-E.

Finding 2: For most kernels, the best-per-application configuration leads to better performance, higher power, and higher temperature, compared to the best-mean configuration. However, for *SNAP* and *HPGMG*, this does not hold. Though their performance and power trends are the same as the other workloads, the temperature with the best-per-application configuration is lower than the mean configuration. This is because more power is shifted from the high-power-density CUs to the lower-power-density in-package DRAM. Fig. 11 shows the temperature difference in the bottom-most in-package DRAM die, with the best-mean configuration versus the best kernel-optimized configuration for *SNAP*.

Takeaway: Our use of aggressive die stacking should be thermally feasible even with air cooling. However, more advanced cooling solutions may become necessary as the hit rate of the in-package DRAM improves, more power from the external memory is shifted to the EHP, or if a design point uses a greater per-node power budget.

E. Power Optimizations

Our research suggests that meeting the exascale energy-efficiency goals would need more than what technology scaling and traditional techniques like DVFS can provide. To this end, we envision using additional aggressive power-saving techniques, each targeting one or more power components. Finding new avenues of power savings needs continuous research effort.

Below, we briefly describe the techniques we have explored and their estimated power benefits.

Near-Threshold Computing (NTC): NTC enables operating near the threshold voltage as long as the underlying

[7]See Section VI where for each kernel we find the combination of CU count, frequency, and memory bandwidth that maximizes performance under the 160W power limit.

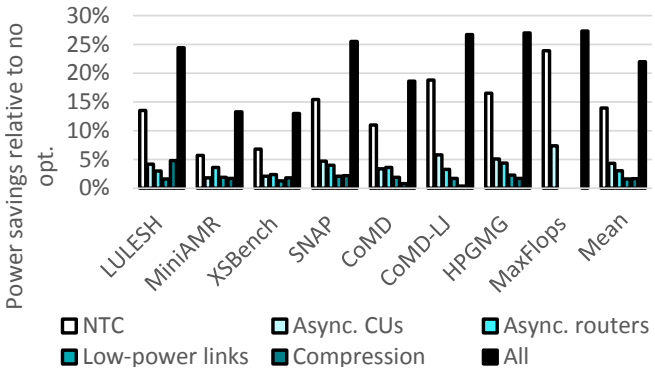

Figure 12. Power savings from optimizations

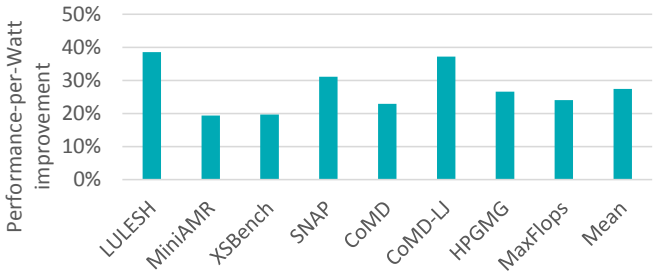

Figure 13. Energy-efficiency benefit from optimizations

circuits are variability tolerant and resilient against errors. Recent advances in our NTC research allows operating the CUs near the threshold voltage at as high as 1 GHz while still obtaining 14% power savings on average across the workloads examined. Given the lower stability of SRAM cells, this paper does not apply NTC to the memory circuits, which would be a potential future research direction.

Asynchronous Compute Units: Most computing devices have synchronous implementations, in which data are stored in registers at the edge of a clock. Although synchronous designs have lower design complexity than asynchronous designs, the former typically has higher dynamic power because of high switching activities and clock-tree complexity. We estimate average system-power savings of 4.3% by carefully applying asynchronous-circuit techniques to only the ALUs and crossbars of the GPU SIMD units.

Asynchronous Routers: When we extend asynchronous circuits to interconnect routers, our study indicates an average power savings of 3.0% can be attained.

Low-Power Links: Operating interconnect links in a low-power mode can lead to average power savings of 1.6%.

DRAM Traffic Compression: Our research shows that a substantial portion of the EHP interconnect power is spent on the long-distance interconnects between the last-level cache (LLC) and the in-package memory. One way to reduce the data-movement power is to apply data compression to the network messages. We estimate average power savings of 1.7% from compression. *LULESH* benefits the most from this optimization, given its high memory intensity.

Finding 1: Fig. 12 shows the power saved when each

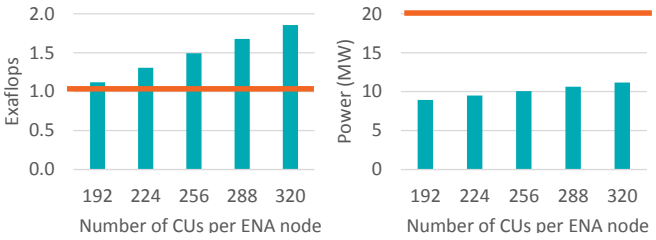

Figure 14. MaxFlops performance and power

of the above techniques is deployed individually and in combination. The power savings range from 13% to 27% when all techniques are deployed together. Note that the baseline power without optimizations already includes power savings from DVFS.

Finding 2: Reduction in power can lead to performance improvements by allowing a higher-performing hardware operating point while staying under the power budget. Fig. 13 shows the improvement in performance per Watt when comparing the new best-mean configuration with optimizations (288 CUs, 1100 MHz, 3 TB/s) to the one before optimizations (320 CUs, 1000 MHz, 3 TB/s). The trend in the efficiency gains across the kernels as seen in Fig. 13 is not the same as the power-savings trend in Fig. 12 because the change in the best-mean configuration (lower CU count and higher frequency) affects different kernels differently.

Takeaway: Optimizing power consumption throughout various node components can improve the node energy efficiency substantially, and may also be necessary if the target applications exercise different parts of the node differently.

F. Exascale Target

In this section, we briefly discuss how our architectural design choices help achieve the overall exaflop performance target within a 20MW power budget for the entire exascale machine. We analyze the the *MaxFlops* application, which represents our most compute-intensive, double-precision floating-point kernel.

Fig. 14 depicts the performance and power scaling trend observed when varying the number of CUs while fixing the GPU frequency and memory bandwidth to 1 GHz and 1 TB/s, respectively. As expected, we see a linear scaling trend with additional CUs. With 320 CUs per ENA, we expect to reach up to 18.6 double-precision teraflops per ENA or *1.86 double-precision exaflops* with a total of 100,000 ENA nodes. This scenario consumes *11.1 MW* of power, although this number should only be considered for a peak-compute scenario. Full-scale applications will tend to exercise more than just the compute resources (e.g., caches, interconnects, internal and external memory), but we expect the design to provide a highly-capable exascale system.

VI. Discussion

The previous section provided preliminary evidence that our proposed ENA meets and possibly even exceeds the aggressive exascale performance, power, and thermal goals for a peak-compute scenario (*MaxFlops*). Ideally, we would like similar levels of compute efficiency for the other more challenging memory-intensive and balanced applications. Furthermore, exascale computing, though challenging, is merely the next HPC milestone. Post-exascale supercomputers will be needed to continue to enhance capabilities as HPC applications become more and more complex.

We briefly discuss some important research directions that the computer-architecture community should pursue. Although these directions are critical for HPC, the technologies developed will also help improve processing in many other domains including mobile, general-purpose, gaming, and cloud computing.

Table II
Performance benefit of dynamic resource reconfiguration

Application	Best App-Specific Config. (CUs / MHz / TB/s)	Perf. Benefit over Best-Mean Config. (%)	
		Without Power Opt.	With Power Opt.
LULESH	256 / 1100 / 4	31.2	38.0
MiniAMR	256 / 1200 / 4	47.3	54.3
XSBench	224 / 1400 / 5	44.9	47.5
SNAP	384 / 700 / 5	18.2	30.2
CoMD	192 / 1500 / 6	40.3	49.8
CoMD-LJ	224 / 1300 / 6	29.6	39.3
HPGMG	352 / 900 / 7	34.9	37.9
MaxFlops	384 / 925 / 1	10.7	19.9

Dynamic Resource Reconfiguration: As Section IV described, not all HPC applications behave in the same manner. A statically fixed hardware configuration would lead to missed opportunities. A run-time technique that adjusts the hardware configuration based on application phases is likely to improve the performance and power despite potential additional management overheads. Although there is already a large body of work in this area, continuous research is necessary to further enhance the efficiency of dynamic reconfiguration.

One of the challenges of dynamic reconfiguration is to *identify* when a kernel reaches a phase beyond which more compute resources and/or higher frequency either (1) do not increase performance but instead lead to higher power consumption (balanced kernels) or (2) cause memory contention, adversely degrading performance (memory-bound kernels). Once such phase is identified, the reconfiguration technique needs to *reduce* the capability of the compute resources via DVFS and/or clock or power gating to operate at an energy-optimal point. Table II shows the potential benefit of an oracle technique that chooses the highest-performing hardware configuration for each kernel while staying under the power budget and area budget of up to 384 CUs per node. As much as 54% additional performance improvement for the ENA can be gained compared to the statically set best-mean configuration of 320 CUs, 1000 MHz, and 3 TB/s, especially when additional power optimizations are enabled.

High-density and Low-energy Memory: As discussed in Section V-C, the higher read and write access energy of NVM (compared to DRAM) leads to a significant power overhead for memory-intensive applications. However, NVM's very low leakage power coupled with high memory densities is desirable to reduce the static power of systems with large memory capacities. Further advances in memory technologies are required to improve not only the density but also the energy efficiency. Additionally, there are ample research opportunities for efficient management of heterogeneous memories and multi-level memories to ensure that more frequently accessed data reside in a faster, lower-access-energy memory.

Resiliency Solutions: As discussed in Section II, supporting resiliency without significant area and performance overheads is critical given the aggressive exascale performance goals, the low mean-time-to-failure requirement, and the increased rate of silent errors that future technology nodes may cause. In addition, some of the power-saving techniques this paper evaluated substantially reduce the operating voltage, potentially increasing error rates. Further research is needed to explore techniques that have low performance and area overheads while providing robust resiliency.

Power Optimizations: This paper considered only a handful of promising power-optimization techniques. Other types of optimizations are also necessary to target the remaining components with high power consumption, such as memory and SerDes links. A management technique must be developed to orchestrate a variety of power optimizations, triggering only those deemed appropriate for a given application phase with negligible performance impact.

VII. Conclusion

In this work, we presented a vision for an exascale node architecture along with the reasoning behind our design choices. We also presented quantitative analysis of these design choices, leveraging results from several research efforts, to make a case for why this approach makes sense given our exascale performance, power, resiliency, and cost goals. Though we have made significant progress toward these goals, we are still actively exploring many research directions to solve the remaining challenges as well as pushing forward to post-exascale systems. We hope that the computer-architecture community will benefit from this work and continue to contribute toward solving these important challenges.

Acknowledgments

AMD, the AMD Arrow logo, Radeon, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

References

[1] "Top500 News Release June 2008," https://www.top500.org/lists/2008/06/.

[2] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark Silicon and the End of Multicore Scaling," in *Proc. of the Int'l Symp. on Computer Architecture (ISCA)*, 2011.

[3] "Pathforward Draft Technical Requirements," https://asc.llnl.gov/pathforward/docs/Attachment_4_PathForward_Draft_Technical_Requirements.docx.

[4] "The Green500 List - June 2016," http://www.green500.org/lists/green201606.

[5] "NVIDIA Wins $18 Million DOE Grant for Exascale Computing Research," https://blogs.nvidia.com/blog/2014/11/14/nvidia-wins-doe-grant-for-exascale-computing-research/.

[6] "Intel Federal LLC To Propel Supercomputing Advancement For The U.S Government," https://newsroom.intel.com/news-releases/intel-federal-llc-to-propel-supercomputing-advancements-for-the-u-s-government/.

[7] "Cray To Explore Alternative Processor Technologies For Supercomputing," http://investors.cray.com/phoenix.zhtml?c=98390&p=irol-newsArticle&ID=1990117.

[8] "MoChi Architecture," http://www.marvell.com/architecture/mochi/.

[9] A. Kannan, N. E. Jerger, and G. H. Loh, "Enabling Interposer-based Disintegration of Multi-core Processors," in *Proc. of the Int'l Symp. on Microarchitecture (MICRO)*, 2015.

[10] "Heterogeneous System Architecture (HSA): Architecture and Algorithms," Tutorial at the Int'l Symp. on Computer Architecture (ISCA), 2014.

[11] "ROCm: Open Platform For Development, Discovery and Education around GPU Computing," gpuopen.com/compute-product/rocm.

[12] JEDEC, "High Bandwidth Memory (HBM) DRAM," http://www.jedec.org/standards-documents/docs/jesd235.

[13] S. Puthoor, A. M. Aji, S. Che, M. Daga, W. Wu, B. M. Beckmann, and G. Rodgers, "Implementing Directed Acyclic Graphs with the Heterogeneous System Architecture," in *Proc. of the Workshop on General Purpose Processing Using Graphics Processing Unit (GPGPU)*, 2016.

[14] B. A. Hechtman, S. Che, D. R. Hower, Y. Tian, B. M. Beckmann, M. D. Hill, S. K. Reinhardt, and D. A. Wood, "QuickRelease: A throughput-oriented approach to release consistency on GPUs," in *Proc. of the Int'l Symp. on High Performance Computer Architecture (HPCA)*, 2014.

[15] D. R. Hower, B. A. Hechtman, B. M. Beckmann, B. R. Gaster, M. D. Hill, S. K. Reinhardt, and D. A. Wood, "Heterogeneous-race-free Memory Models," in *Proc. of the Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2014.

[16] M. S. Orr, S. Che, A. Yilmazer, B. M. Beckmann, M. D. Hill, and D. A. Wood, "Synchronization Using Remote-Scope Promotion," in *Proc. of the Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2015.

[17] J. Alsop, M. S. Orr, B. M. Beckmann, and D. A. Wood, "Lazy Release Consistency for GPUs," in *Proc. of the Int'l Symp. on Microarchitecture (MICRO)*, 2016.

[18] J. Power, A. Basu, J. Gu, S. Puthoor, B. M. Beckmann, M. D. Hill, S. K. Reinhardt, and D. A. Wood, "Heterogeneous System Coherence for Integrated CPU-GPU Systems," in *Proc. of the Int'l Symp. on Microarchitecture (MICRO)*, 2013.

[19] K. Saban, "Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency," Xilinx, White Paper, 2011.

[20] "AMD Radeon R9 Series Gaming Graphics Cards with High Bandwidth Memory," http://www.amd.com/en-us/products/graphics/desktop/r9#.

[21] P. V. et al., "A 4x4x2 Homogeneous Scalable 3D Network-on-Chip Circuit with 326MFlit/s 0.66pJ/b Robust and Fault-Tolerant Asynchronous 3D Links," in *Proc. of the Int'l Solid-State Circuits Conference (ISSCC)*, 2016.

[22] S. Das and G. Sadowski, "Reconfigurable links for self-timed on-chip communication," in *Proc. of the Int'l Workshop on Network on Chip Architectures (NoCArc)*, 2016.

[23] A. Majumdar, L. Piga, I. Paul, J. L. Greathouse, W. Huang, and D. H. Albonesi, "Dynamic gpgpu power management using adaptive model predictive control," in *Proc. of the Int'l Symp. on High Performance Computer Architecture (HPCA)*, 2017.

[24] K. Dev, S. Reda, I. Paul, W. Huang, and W. Burleson, "Workload-aware power gating design and run-time management for massively parallel gpgpus," in *Proc. of the Symp. on VLSI (ISVLSI)*, 2016.

[25] J. Wadden, A. Lyashevsky, S. Gurumurthi, V. Sridharan, and K. Skadron, "Real-world Design and Evaluation of Compiler-managed GPU Redundant Multithreading," in *Proc. of the Int'l Symp. on Computer Architecuture (ISCA)*, 2014.

[26] M. Oskin and G. H. Loh, "Software-managed Approach to Die-Stacked DRAM," in *Proc. of the Int'l Symp. on Parallel Architectures and Compilation Techniques (PACT)*, 2015.

[27] M. Meswani, S. Balgodurov, D. Roberts, J. Slice, M. Ignatowski, and G. Loh, "Heterogeneous Memory Architectures: A HW/SW Approach for Mixing Die-stacked and Off-package Memories," in *Proc. of the Int'l Symp. on High-Performance Computer Architecture (HPCA)*, 2015.

[28] "Fast Forward 2 R&D Draft Statement of Work," https://asc.llnl.gov/fastforward/rfp/04_DraftSOW_04-03-2014.pdf.

[29] G. Kim, J. Kim, J. H. Ahn, and Y. Kwon, "Memory Network: Enabling Technology for Scalable Near-Data Computing," in *Proc. of the Workshop on Near-Data Processing*, 2014.

[30] G. Kim, J. Kim, J.-H. Ahn, and J. Kim, "Memory-centric system interconnect design with hybrid memory cubes," in *Proc. of the Int'l Symp. on Parallel Architectures and Compilation Techniques (PACT)*, 2013.

[31] J. T. Pawlowski, "Hybrid Memory Cube: Breakthrough DRAM Performance with a Fundamentally Re-Architected DRAM Subsystem," in *Hot Chips 23*, 2011.

[32] J. Sim, A. R. Alameldeen, Z. Chishti, C. Wilkerson, and H. Kim, "Transparent Hardware Management of Stacked DRAM as Part of Memory," in *Proc. of the Int'l Symp. on Microarchitecture (MICRO)*, 2014.

[33] M. Meswani, G. H. Loh, S. Blagodurov, D. Roberts, J. Slice, and M. Ignatowski, "Toward Efficient Programmer-managed Two-level Memory Hierarchies in Exascale Computers," in *Proc. of the Int'l Workshop on Hardware-Software Co-Design for High Performance Computing*, 2015.

[34] G. H. Loh and M. D. Hill, "Efficiently Enabling Conventional Block Sizes for Very Large Die-Stacked DRAM Caches," in *Proc. of the Int'l Symp. on Microarchitecture (MICRO)*, 2011.

[35] M. K. Qureshi and G. H. Loh, "Fundamental Latency Trade-offs in Architecting DRAM Caches," in *Proc. of the Int'l Symp. on Microarchitecture (MICRO)*, 2012.

[36] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The Gem5 Simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, 2011.

[37] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *Proc. of the Int'l Symp. on Performance Analysis of Systems and Software (ISPASS)*, 2009.

[38] J. L. Greathouse, A. Lyashevsky, M. Meswani, N. Jayasena, and M. Ignatowski, "Simulation of Exascale Nodes through Runtime Hardware Monitoring," in *the Workshop on Modeling & Simulation of Exascale Systems and Applications (ModSim)*, 2013.

[39] B. Su, J. L. Greathouse, J. Gu, M. Boyer, L. Shen, and Z. Wang, "Implementing a Leading Loads Performance Predictor on Commodity Processors," in *Proc. of the USENIX Annual Technical Conf. (USENIX ATC)*, 2014.

[40] B. Su, J. Gu, L. Shen, W. Huang, J. L. Greathouse, and Z. Wang, "PPEP: Online Performance, Power, and Energy Prediction Framework and DVFS Space Exploration," in *Proc. of the Int'l Symp. on Microarchitecture (MICRO)*, 2014.

[41] V. Adhinarayanan, I. Paul, J. L. Greathouse, W. Huang, A. Pattnaik, and W. chun Feng, "Measuring and modeling on-chip interconnect power on real hardware," in *Proc. of the Int'l Symp. on Workload Characterization (IISWC)*, 2016.

[42] G. Wu, J. L. Greathouse, A. Lyashevsky, N. Jayasena, and D. Chiou, "Gpgpu performance and power estimation using machine learning," in *Proc. of the Int'l Symp. on High Performance Computer Architecture (HPCA)*, 2015.

[43] A. Majumdar, G. Wu, K. Dev, J. L. Greathouse, I. Paul, W. Huang, A. K. Venugopal, L. Piga, C. Freitag, and S. Puthoor, "A taxonomy of gpgpu performance scaling," in *Proc. of the Int'l Symp. on Workload Characterization (IISWC)*, 2015.

[44] B. M. Beckmann and A. Gutierrez, "The AMD gem5 APU Simulator: Modeling Heterogeneous Systems in gem5," Tutorial at the Int'l Symp. on Microarchitecture (MICRO), 2015.

[45] "Proxy Applications for Co-Design," proxyapps.lanl.gov/.

[46] "MaxFlops: Workload Description," http://ft.ornl.gov/doku/shoc/pflops.

[47] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: a compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. VLSI Syst.*, vol. 14, no. 5, 2006.

[48] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *Proc. of the Int'l Symp. on Computer Architecture (ISCA)*, 2012.

[49] Y. Eckert, N. Jayasena, and G. H. Loh, "Thermal Feasibility of Die-Stacked Processing in Memory," in *Proc. of the Workshop on Near-Data Processing*, 2014.

[50] M. Skach, M. Arora, C.-H. Hsu, Q. Li, D. Tullsen, L. Tang, and J. Mars, "Thermal Time Shifting: Leveraging Phase Change Materials to Reduce Cooling Costs in Warehouse-scale Computers," in *Proc. of the Int'l Symp. on Computer Architecture (ISCA)*, 2015.