DCRA: A Distributed Chiplet-based Reconfigurable Architecture for Irregular Applications

Marcelo Orenes-Vera, Esin Tureci, Margaret Martonosi, David Wentzlaff
Princeton University, Princeton, New Jersey, USA
{movera, esin.tureci, mrm, wentzlaf} @princeton.edu

***Abstract*—In recent years, the growing demand to process large graphs and sparse datasets has led to increased research efforts to develop hardware- and software-based architectural solutions to accelerate them. While some of these approaches achieve scalable parallelization with up to thousands of cores, adaptation of these proposals by the industry remained slow. To help solve this dissonance, we identified a set of questions and considerations that current research has not considered deeply.**

Starting from a tile-based architecture, we put forward a Distributed Chiplet-based Reconfigurable Architecture (DCRA) for irregular applications that carefully consider fabrication constraints that made prior work either hard or costly to implement or too rigid to be applied. We identify and study pre-silicon, package-time and compile-time configurations that help optimize DCRA for different deployments and target metrics. To enable that, we propose a practical path for manufacturing chip packages by composing variable numbers of DCRA and memory dies, with a software-configurable Torus network to connect them. We evaluate six applications and four datasets, with several configurations and memory technologies, to provide a detailed analysis of the performance, power, and cost of DCRA as a compute node for scale-out sparse data processing. Finally, we present our findings and discuss how DCRA—together with our framework for design exploration—can help guide architects to build scalable and cost-efficient systems for irregular applications.

I. Introduction

Irregular applications are those traversing data structures such as trees, graphs, and sparse matrices, whose data-dependent access patterns lead to unpredictable memory accesses. They include applications doing graph analytics, sparse tensor algebra, particle simulations, and database operations. The increasing relevance of these applications has led the computer architecture community to develop many promising software and hardware techniques to accelerate them [4], [13], [19], [41], [58], [59], [63], [66], [73]. However, there is still an unmet demand for systems that can accelerate these applications at scale [26], [75].

Motivated by the idea of building such systems, we faced the following research questions: (a) what is the design space of architectures for irregular applications; (b) how do different target metrics (e.g., time-to-solution, energy, cost) affect design choices; and (c) how different applications and datasets are affected by these decisions.

Prior work Dalorex managed to scale billion-edge graphs across thousands of cores by having the entire graph stored on-chip. It followed a similar approach to the Cerebras Wafer-Scale Engine [8] of having a huge amount of SRAM on-chip, distributed evenly across a tiled grid of processing elements (PEs). While this approach achieves the largest memory bandwidth per PE—a precious resource for data-intensive applications—it sets the ratio of compute and memory per PE at silicon time, which then constrains the minimal level of parallelization at which to run a given dataset. Memory bandwidth is only part of the solution to scaling out irregular applications, as the network also plays a key role when scaling to a large number of PEs. Building large networks on a 2D silicon limits the options to high-diameter 2D topologies, which only scale bisection bandwidth—also a precious resource—with the square root of the number of PEs. Thus, massively scaling irregular applications would eventually require having chips distributed on a more beefy network in the 3D space [38], [43], [48] and using problem partitioning to create locality within each node [35], [36], [72]. But how big each node should be and the design tradeoffs to consider when architecting it, are open questions that we aim to answer in this paper.

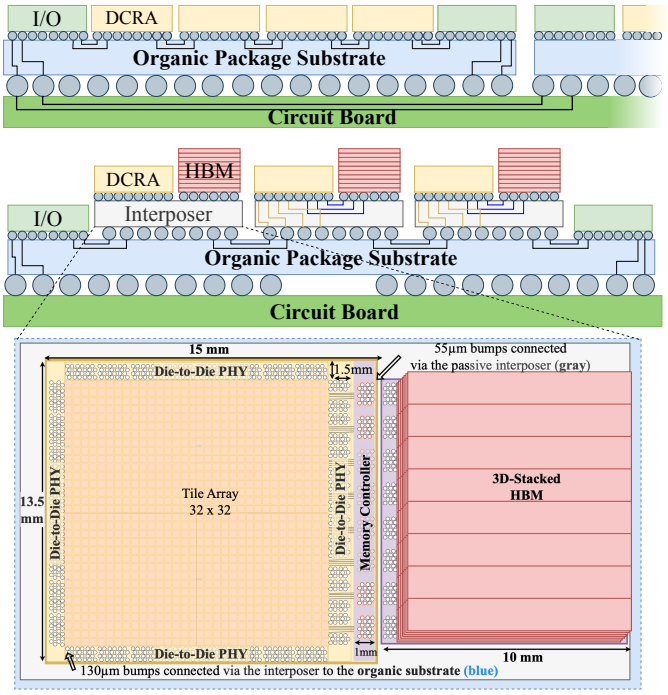

Fig. 1. Two possible integrations of the same 32x32-tile DCRA die (dimensions depicted with 256KB SRAM/tile). Top: two packages on a board, each featuring only DCRA dies, optimized for time-to-solution as it maximizes parallelization (lower data footprint per die). Bottom: a single package with DCRA dies and stacked DRAM, optimized for performance-per-dollar.

Despite the large corpus of work on irregular memory access patterns and graph processing (cite a bunch), there is still a lack of studies on the limits of parallelization within each node and the tradeoffs between performance, power, area (PPA) and cost involved in building it.

Our work: To answer these research questions, we revisit recent works in manycore architectures and graph processing through the lens of modern technology trends to: (1) understand the manufacturing process and constraints of the architectures proposed by prior work; (2) study how the balance of hardware resources dedicated to compute, memory, and network affects the energy, performance, and cost of running irregular applications; (3) propose DCRA, a distributed chiplet-based reconfigurable architecture that allows taking some of these design decisions post-silicon—at chip packaging time; (4) analyze several pre-silicon options and packaging- and compile-time configurations of DCRA in terms of performance, energy and cost efficiency for six irregular applications.

Designing computing nodes composed of tens of thousands of cores requires taking cost constraints into account. We investigated the fabrication cost and resource utilization of monolithic architectures like Dalorex and found that ***it is hard to strike a balance for the amount of SRAM per PE*** tile. A small SRAM per tile results in higher peak compute throughput for the same area, but it forces the spread of large datasets across more PEs, putting more pressure on the network and resulting in low utilization, for communication-intensive applications. In contrast, with a large SRAM per tile, the dataset can be accommodated across fewer tiles and get closer to the ideal compute utilization, but the performance per dollar will be lower when we want to achieve maximum parallelization. A similar problem applies to processing-in-memory (PIM) architectures like Tesseract [4] and GraphQ [93]; the ratio of PEs per DRAM bank is fixed during fabrication, leading to a similar tradeoff at a different scale.

Because the cost of fabricating separate silicon dies for products with different performance targets could thwart performance-per-dollar goals, we propose DCRA, a composable chiplet architecture where crucial design decisions like the ratio of memory to compute resources, total on-chip memory capacity or off-chip bandwidth are taken more cost-efficiently, during chip-packaging.

Fig. 1 depicts two packaging configurations where the same DCRA dies are integrated to create different chip products, each optimized for a target metric, i.e., time-to-solution (above) and performance-per-dollar (below). DRAM devices can be interleaved between DCRA dies to increase the capacity of a tile's local memory since the DRAM storage is exclusively assigned to the tiles of the adjacent DCRA die. The IO dies on the sides of the package determine the off-chip bandwidth. Multiple chips could be integrated into a board, which would define what we consider a *compute node*.

The rest of the paper is organized as follows. §II reviews the state-of-the-art in manycore and graph architectures and motivates why a chiplet-based reconfigurable architecture can help meet different PPA targets cost-effectively for irregular applications. §III describes the DCRA architecture, including our design contributions towards supporting nodes of configurable sizes optimized for irregular applications, where memory and compute are interleaved on-chip, connected by a software-defined Torus network. It also describes the design innovations that enabled some of these configurations. §IV describes the chiplet model and the cost model that we built on top of the simulator used for Dalorex, and the applications and datasets used in our evaluation. §V presents our characterizations of the DCRA pre-silicon options, as well as the post-silicon and software-time configurations, and shows the performance-per-dollar and energy-efficiency benefits of DCRA over prior work. Before concluding, Fig. 12 presents the insights we gained from these characterizations and provides guidance for future designs.

II. Background and Motivation

Memory accesses from graph and sparse linear algebra applications exhibit little spatial or temporal locality, resulting in poor cache behavior and intense traffic throughout the memory hierarchy [52]. Prior work aiming to accelerate these workloads mitigate memory latency via decoupling, prefetching and pipelining techniques [2], [13], [25], [32], [58], [59], [63], [67], [73], [74], [84], [88]. Fifer [59] and Polygraph [13] increase utilization further through spatiotemporal parallelization, while Hive [73] provides ordered parallelization. These works improve the execution of irregular applications via hardware-software co-designs, without exploring much of the hardware design space in terms of balancing compute and memory resources. Tesseract [4] and GraphQ [94] explored tight integrations of compute and memory resources by integrating PEs on the logic layer of a 3D stacked memory [71], while Dalorex [66] proposes having all the memory on SRAM, scattered evenly across the PEs, to increase even further the memory bandwidth. In addition, Dalorex builds on prior software techniques and adds hardware support to offer a programming model where the program is split into tasks at pointer indirection so that all tasks execute on the PEs that have the data in their local memory. Thanks to all of this prior work, as a community, we have a better understanding of how to map and orchestrate irregular applications to the hardware. We argue that the next step toward realizing some of these hardware-software innovations at scale is ***exploring the architecture design space while considering physical implementation constraints and cost***.

For the rest of this section, we explain how chiplets can help with cost-effectiveness beyond improving silicon yield (§II-A) and review the state-of-the-art in manycore architectures (§II-B) to motivate the need for a new reconfigurable architecture for irregular applications.

A. Chiplets and Fabrication Cost

The semiconductor industry is increasingly utilizing multiple dies in a chip package [5], [22], [50], [57], [83] not

TABLE I
POST-SILICON CONFIGURABILITY OF THE NETWORK-ON-CHIP (NOC) TOPOLOGY, MEMORY CAPACITY AND # OF PROCESSING ELEMENTS (PES) PER CHIP, AND WHETHER THE PES CAN EXECUTE SOFTWARE INSTRUCTIONS.

Reconfigurable Aspects / Prior Work	Memory Size/BW per chip	#Processing Elements per chip	Configurable Network Topology	Software Programmable Processors
Fifer [15]	✓	✗	✗	✗
Tesseract [4]	✓	✗	✗	✓
Dalorex [66]	✗	✗	✗	✓
PolyGraph [13]	✓	✗	✗	✗
Decades [7], [18]	✗	✗	✗	✓
ESP [21]	✗	✗	✗	✓
Manticore [92]	✓	✗	✗	✓
Simba [76]	✗	✓	✓	✓
Intel's PIUMA [1]	✓	✗	✗	✓
Graphcore [41]	✗	✗	✓	✓
Sambanova [14]	✓	✓	✓	✗
Cerebras [8]	✗	✗	✓	✓
Groq [24]	✗	✗	✗	✓
Tesla Dojo [24]	✓	✓	✗	✓
Esperanto [15]	✗	✗	✗	✓
Google TPU [34]	✓	✗	✓	✗
Nvidia A100 [10]	✓	✗	✗	✓
DCRA	✓	✓	✓	✓

only to increase silicon lithography yield but also to enable reusing components across different chip products. From the silicon manufacturing perspective, the Non-Recurring-Engineering (NRE) cost of a wafer mask is so high with respect to the silicon wafer itself that the cost per wafer is 18× larger when manufacturing 100 wafers, instead of 100,000 [33]. Since the volume of silicon production of each chip architecture generation is not as high in the HPC market as in the consumer electronic market, we argue that the mass fabrication of a chiplet that serves as a building block for scalable architectures is a valuable proposition. Moreover, from the supply-chain [60] and the embodied-carbon perspective [3], [9], [23], it is more agile and sustainable to integrate existing chiplets differently to meet the demands of various products than fabricating a new chip for each one. Thus, we set out to design a chiplet that can be fabricated at high volume to amortize NRE costs, and then integrated into a Multi-Chip Module (MCM) package at the scale and configuration that best meets the performance, power, and cost requirements of the target product, within the irregular application domain. Beyond being able to create chips for different markets (e.g., single-die for automobile chips or full nodes for HPC clusters), graph-search-oriented supercomputers could integrate larger or smaller chip packages per node, depending on the target performance and cost.

B. Design Space of Manycore Systems

In recent years, the widespread demand for large deep-learning models has accelerated the development of manycore and dataflow systems that can massively parallelize compute-intensive workloads. Although the demand for graph and sparse linear algebra workloads, as well as target data sizes, are growing, we have not seen systems exhibiting as high scalability as dense systems. This is because the irregular data-access patterns and the low arithmetic intensity (operations/byte) challenge the network and the memory systems optimized for dense workloads.

Table I shows a selection of the recent manycore systems starting with those focusing on sparse workloads and continuing with those focusing on AI. While some of these manycores have good attributes for sparse data processing (large SRAM capacity [24], [41], [64], [66], [85] or on-chip DRAM [4], [10], [14], [34], [92]), the ratio between on-chip memory and PUs is optimized for dense computation, and the interconnect is designed for dataflow communication. And precisely these resources—the network-on-chip (NoC) and the on-chip memory capacity and bandwidth—are the most important ones for irregular applications. Note that we use chip and package interchangeably, and chiplet to refer to a die that can be integrated into a chip package with others.

Memory Configurability: Based on our experiments, there is no universally optimal memory-hierarchy configuration for irregular applications that optimizes for all important target metrics (e.g., time-to-solution, and performance per cost or power unit). This is exacerbated by the fact that graphs and sparse datasets for different application domains can have very different degree distributions, local structure and size and therefore may present different optimization landscapes. Therefore, we argue that—given the NRE costs—it is more cost-efficient to design a chiplet architecture that allows for optimizations during packaging than to fabricate a new chip for each configuration. Fig. 1 illustrates that DCRA allows opting whether to integrate DRAM on the package, and it enables interleaving DRAM devices between columns of DCRA dies—not just on the edges of the package as in existing chiplet-based designs. §III describes how DCRA allows for this flexibility, which can enable different cost-optimal design points given the variance in HBM prices with availability [16].

Network Configurability: One of the key attributes that improved the performance of irregular applications in Dalorex was using a torus NoC, which helped to reduce network congestion caused by irregular communication patterns. A torus NoC can be implemented without a long wrap-around wire by *folding* it into the 2D space, i.e., having even-indexed tiles connected, representing one bisection of the network and odd-indexed tiles representing the other bisection [17]. However, this sets the size of the torus NoC at silicon time. Since we want to support integrating a scaling number of chiplets and keep having them connected in a torus, we propose a *reconfigurable* torus NoC where the topology is defined prior to execution (see §III-A).

Compute Configurability: The NoC and memory configurations are geared towards being able to feed the PEs with data, which is much more challenging for irregular applications than for dense ones. The number of PEs per chip and per node will ultimately shape the topology of the cluster-level network,

the size of the dataset that can be processed, and at what speed. Thus, being able to defer this decision to post-silicon, is another motivation for having this level of configurability. We also advocate for the flexibility of ISA-programmable PEs over the efficiency gains of ASICs or CGRAs because the compute area and power needed for irregular applications is low in comparison with the memory and network resources—as we show in §V.

III. The DCRA Architecture

Having motivated the need for a reconfigurable architecture for irregular applications to meet different PPA and cost targets, this section describes how DCRA architecture achieves that. First, §III-A describes how our Torus NoC design can be configured to span multiple dies and packages. Then, §III-B describes how we manage the tile's SRAM as a cache and/or as a scratchpad, and provide prefetching support by leveraging the task-based execution model. §III-C describes how the DRAM dies can be interleaved in between the DCRA dies. Finally, Table II summarizes the configurable knobs and serves as an outline for our evaluations in §V.

Dalorex's task-based execution model: We use the latest work on sparse architectures, Dalorex [66], as a starting point to build our base design. We take their task-based execution model, enabled by the Task-scheduling Unit (TSU), which is the nexus between the NoC router and the Processing Unit (PU) executing the tasks. When the PU executes a task it can potentially spawn new tasks by writing task-invocation parameters into the output queues (OQs)—one per task type. The router eventually takes them from the OQs and places them into the NoC. Task invocations are routed to the tile that owns the data they operate on because the dataset layout into the partitioned global address space (PGAS) is statically known. At the destination tile, the router places a task-invocation message into the tile's input queue (IQ), so that the TSU can schedule the task to the PU. Since there is one IQ per type of task, the TSU also prioritizes the tasks based on their type, with a heuristic informed by the occupancy of the task queues.

This execution model—where every tile only operates on local data—shifts the problem of irregular memory-access patterns to irregular inter-tile communication. Dalorex observed that when dealing with irregular traffic a 2D torus provides a more uniform utilization than a 2D mesh [66]. To make implementation practical on 2D silicon, the 2D torus is folded (as described in §II-B). However, Dalorex assumes a monolithic wafer-scale design and does not solve the problem of setting up a 2D torus spanning different grid sizes.

A. Designing a reconfigurable 2D Torus

Motivated by our need to scale irregular communication while providing a chiplet-based integration, we design a Torus NoC whose topology can be configured in software. For example, we can confine the Torus within a die, or have it span multiple dies and packages within the board of a compute node. (Table III shows the interconnect energy and latency assumed for our evaluation for hops at each level.) Fig. 2 shows that the routers at the edges of each die can be configured to connect to a router on the next die or wrap around by connecting to the adjacent tile (Tiles 0 and 63). Moreover, we can reconfigure a 2D-torus NoC into two 2D-mesh NoCs by not connecting the wrap-around links on the routers at the edges. This allows for a 2D-mesh NoC to be used for streaming data from I/O to the tiles, and then reconfigure it to a 2D-Torus NoC for the rest of the execution.

Reducing long-distance communication: To reduce the number of hops across chiplets, DCRA has two hierarchical NoCs, one that connects every tile and one that hops once per die, as shown in Fig. 2. Each NoC topology is individually configured. While bringing the data inside the package from the disk, they both would be configured as a mesh to increase I/O. During the execution, both may become torus, or the die-NoC may also remain open to stream I/O data.

After doing a literature review, we found that—to the best of our knowledge—our proposal is the first 2D-Torus NoC topology for which the size of the network can be configured at runtime, being able to span one torus across multiple dies, or have multiple torus networks.

Off-Chip Bandwidth: DCRA employs I/O chiplets to connect chip packages. This allows DCRA dies to remain agnostic to a specific off-chip protocol (e.g., PCIe 6.0 [77]). It also defers choices like off-chip bandwidth to packaging time, to decide based on the requirements of the final product. The off-chip bandwidth could be as high as the I/O-DCRA bandwidth.

B. Reconfigurable SRAM Management

In Dalorex, all the dataset arrays had to fit within the aggregated SRAM memory of the grid of tiles. While this is suited for large levels of parallelization that aim to achieve the fastest time-to-solution, being forced to scale out with the dataset size can become very costly (shown in §V-D). Thus, we need to support data caches that are backed up by DRAM.

Keeping the same tile design as in Dalorex, we manage the SRAM scratchpad vastly differently. Inspired by prior work on reconfigurable caches [11], [12], [42], DCRA offers to the software the appearance of a tile's local address space, which is mapped to the SRAM as a cache and/or as a scratchpad.

Cache mode: The cache mode allocates a portion of the SRAM as a direct-mapped cache, for a given address range of the tile's local address space (a segment). It stores cache-line tags and the valid/dirty bits in SRAM too, so the area overhead of the cache is only the logic for tag comparison. To minimize the hardware and energy overhead of using the SRAM as a cache, we make the caches directly mapped. The cacheline width equals the bitline width of the DRAM memory controller (512 bits in our experiments).

Scratchpad mode: When scaling out the parallelization of a dataset, if the memory footprint per tile fits in the local SRAM, the data cache would not be configured. Instead, the data segment would map directly into the SRAM scratchpad. A data segment can also be mapped directly to the SRAM scratchpad, even when the data cache is enabled.

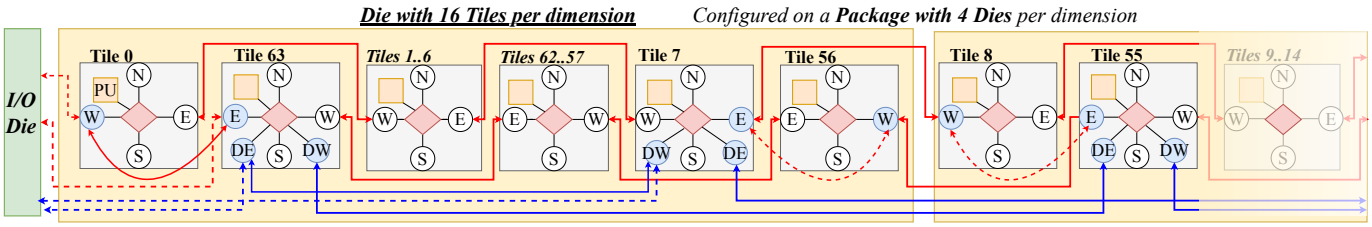

Fig. 2. Horizontal links within a DCRA die and across dies. The red links show the NoC that connects every tile (*tile-NoC*), while the blue links show the NoC that connects to one tile per die (*die-NoC*). Because of the die-NoC, the routers at the die edges are radix-9, while the rest are radix-5. The ports shadowed in blue are runtime reconfigurable; any tile subgrid within a node board may become torus (including across packages). The dies on the edges of a package will interface with the I/O die. All I/O links are configured when loading the dataset to maximize I/O bandwidth. During program execution, both NoCs may become torus, or tile-NoC torus and die-NoC mesh, to keep streaming from I/O.

Data cache misses and evictions: Upon a miss, the data cache fetches the full cacheline from DRAM without checking for coherence since the data arrays in the data segment are not shared. This fetch uses a physical address since each DRAM module is private to each DCRA die with 1-1 mapping between tiles and DRAM vaults. Each tile's local SRAM is backed up by a DRAM storage of size $DRAM\_capacity/tiles\_per\_die$. (The details of the DRAM devices assumed for the evaluation are described in Table III.) The data cache has a dirty bit per line to write back to DRAM upon eviction. Since the data cache of each tile only contains the part of the dataset that the tile is responsible for, there are no coherence issues for modified data.

Prefetching: The PU has a very simple in-order pipeline, which stalls waiting for data on a D$ miss. Since the first parameter of every task message contains an array index, and the TSU knows to which array it corresponds (used for NoC routing), we use this information to prefetch the data. A task may access more than one array using the same index, so we add another pointer to the TSU's per-task table so that it prefetches both when needed. Those tables also have an extra bit saying whether the PU keeps prefetching data during the task execution. Since pointer indirections are split into tasks, when tasks access multiple array elements, they often do it with a streaming pattern, To prefetch these, we enable PU's next-line prefetcher for tasks that access multiple elements.

C. Reconfigurable On-chip Memory Capacity

The memory hierarchy determines the bandwidth available to computing units and, thus, at which level of arithmetic intensity the chip becomes memory-bound. To increase memory bandwidth, prior work has proposed having DRAM on the chip. Prior work proposed 3D integrations where processing cores are on the base plane of the 3D-stacked memory, accessing the TSVs directly [4], [39], [93], [94]. However, this may render impractical. There are very few foundries that can fabricate HBM technology, and the design seems to only change with every generation of JEDEC specification [31] (e.g., HBM2, HBM2e, HBM3) which vendors conform to. So it would be expensive—if possible—to fabricate an HBM die with a custom base layer. Moreover, the base layer of the HBM stacks is already filled with MBIST logic and PHY [27], [47], [61], [69]. Alternatively, other works have proposed integrations of off-the-shelf HBM dies with compute dies by having the compute dies surrounded by HBM devices on the edges of the chip [5], [10], [29], [51], [56], [57], [89], [92]. DCRA is the first proposal for having horizontally integrated HBM dies (2.5D) interleaved across columns of compute dies (as shown in Fig. 3). This allows the design of DCRA dies to be agnostic to the number of DCRA and DRAM dies that are eventually integrated on-chip.

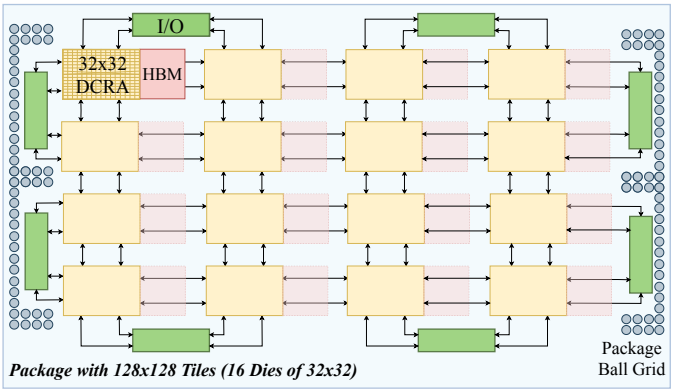

Fig. 3. Top view of a chip package with 128x128 tiles. The number of DCRA dies would determine the compute capacity, while including HBM dies would determine the compute-to-memory ratio of the chip architecture. The number of I/O dies (and their bandwidth) determines the off-chip bandwidth.

Interleaving DRAM & DCRA dies: Figs.1 and 3 depict the integration of DCRA, and HBM dies via a passive silicon interposer. This provides higher DCRA-HBM bandwidth than a silicon bridge [51] or substrate integration [5], [44]. The interposer only contains the wires between a DCRA die, and its HBM die since HBM access is exclusive. The links connecting DCRA dies are routed through the organic substrate, which also contains the power delivery and redistribution layer.

Memory controller: As depicted in Fig. 1, the memory controller lives on the DCRA die, and thus, in case of not integrating DRAM, it becomes *dark silicon*. Although in our evaluation, we consider it as such, future work could investigate using an embedded FPGA logic that allows hardening a memory controller of choice (for HBM or DDR, opting for high bandwidth or high capacity) or other computing logic (in the case DRAM is not integrated).

D. Choosing the Right Configuration

Now that we have established the design of the DCRA architecture, we are ready to discuss the design space exploration that we performed in our evaluation to find the adequate

TABLE II
RECONFIGURABLE PARAMETERS OF DCRA

Tapeout-time Design Decisions
1. # of Tiles per die
2. # of PUs per tile (and their operating and max. frequency)
3. SRAM capacity per tile (KB)
4. NoC Width (and the operating and max. frequency)
Packaging-time Design Decisions
5. # of DCRA dies per package
6. # of DRAM dies per package and capacity of each (GB)
7. # of I/O dies per package (Off-Package BW)
Compile Time Configurations
8. Size of the input and output queues (for every task type)
9. Size and Place of the grid that the workload uses (grid of dies)
10. The address space for which the data is cached
11. Size of the D$ (in data elements)

pre-silicon options for the sparse applications that we target. Table II shows the pre-silicon options, and post-silicon and compile-time configurations that can be selected for the DCRA architecture. In §V we present our evaluation results for most of these features, in the order that they are presented in table II. Moreover, in Fig. 12 we provide guidance on how to choose the pre-silicon and packaging-time configuration of DCRA based on the target deployment of the final product.

IV. EVALUATION FRAMEWORK

This section describes the methodology we use to evaluate DCRA, starting with the applications and datasets we use, followed by the simulation infrastructure we built, including the cost model we use to evaluate the cost-effectiveness of different configurations.

Applications: We evaluate the performance of DCRA on four graph workloads, SPMV, and histogram [82].*Breadth-First Search (BFS)* determines the number of hops from a root vertex to all vertices reachable from it; *Single-Source Shortest Path (SSSP)* finds the shortest path from the root to each reachable vertex; *PageRank* ranks websites based on the potential flow of users to each page [45]; *Weakly Connected Components (WCC)* finds and labels each set of vertices reachable from one to all others in at least one direction (using graph coloring [79]); *Sparse Matrix-Vector Multiplication (SPMV)* multiplies a sparse matrix with a dense vector. *Histogram* counts the values that fall within a series of intervals. We report traversed edges per second as $TEPS = m/time$ where m is the number of edges connected to the vertices in the graph starting from the search key. When we report TEPS for SPMV and Histogram, we consider the non-zero elements to multiply, and elements to process, respectively.

Datasets: We use three sizes of the RMAT [49] graphs—standard on the Graph500 list [55]—RMAT-22, RMAT-25 and RMAT-26, which are named after their number of vertices. For example, RMAT-26 (abbreviated as R26 in §V) contains 2^{26}, i.e., 67M vertices (V) and 1.3B edges (E), and has a memory footprint of 12GB. We also use the Wikipedia (WK) graph (V=4.2M, E=101M) in our evaluation to exercise different graph topologies. For SPMV we use the same datasets as a graph can be seen as a square sparse matrix with V rows and columns and E non-zero elements. The graphs (as sparse matrices) are stored in the Compressed Sparse Row (CSR) format without any partitioning, i.e, the dataset contains three input arrays, one for the values of the non-zeros, one for the column indices of those non-zeros, and one for the pointers to the beginning of each row in the previous two arrays. The output array has size V, and its meaning depends on the application, e.g., for Histogram, it is the count of the column indices of the non-zeros.

Simulation: We built our evaluation framework on top of the functional simulator from Dalorex [66]—a cycle-accurate simulator for the NoC and based on performance models for the PUs—which we extended to support chiplets, caches and DRAM. We chose this to make it easier to compare over Dalorex and because faithfully modeling the NoC is the most critical part for these large parallelizations of data-dependant applications. Table III summarizes some of our additions to the energy, latency, and area model for communication links and memory technology. The full set of energy, area and performance parameters we used for our evaluation can be found in our repository [65], which also includes a Readme file will all the scripts used to generate the artifacts and plot the evaluation figures in this paper. Since we test different frequencies for the NoC and the PUs on Figs. 4 and 7, our area and energy numbers scale with frequency and voltage [30]. For the rest of the experiments, our operating frequency is 1Ghz.

Cost Model: We also added a cost model to the simulator to study the cost-effectiveness of different DCRA configurations. The cost model—similarly to the energy model—is decoupled from the runtime simulation process, i.e., cost and energy can be re-calculated post-simulation for different parameters. This is useful to study how price variations can change the cost-effectiveness of different configurations.

Silicon cost: All of our evaluations consider 7nm technology for the DCRA chiplets; we assume that a 300mm wafer with this transistor process costs $6,047 [33]. We obtain the cost per die by dividing the wafer cost by the number of good dies, which we calculate using 0.2mm scribes, 4mm edge loss, and 0.07 defects per mm^2. (We integrate and validate [54] die yield calculations in our cost model using Murphy's model.) When comparing the cost-effectiveness of our results, we do not include the Non-Recurring Engineering (NRE) cost of the DCRA dies since all the options use the same technology.

Packaging cost: In terms of packaging, all our results featuring grid sizes over 128x128 use multiple packages (of 64x64 tiles each, based on our rationale from §III-D). We assume the cost of the 65nm silicon interposer connecting a DCRA die with HBM (including bonding) to be 20% of the price of a DCRA die [86]; the cost of an organic substrate to be 10% of the price of an equal-sized DCRA die, and the bonding to add an additional 5% overhead [46], [81].

DRAM cost: Regarding *DRAM*, we assume an 8GB HBM2E device with eight 64GB/s memory channels. While

TABLE III
ENERGY (E), BANDWIDTH, LATENCY, AND AREA OF LINKS AND MEMORY DEVICES ASSUMED FOR THE EVALUATION.

Memory Model Parameters	**Values**
SRAM Density	3.5 MB/mm^2 [90]
SRAM R/W Latency & E.	0.82ns & 0.18 / 0.28 pJ/bit [90]
Cache Tag Read & cmp. E.	6.3 pJ [90], [91]
HBM2E 4-high Density	8GB/110mm^2 (75 MB/mm^2) [47]
Mem.Channels & Bandwidth	8 x 64 GB/s [47]
Mem.Ctrl-to-HBM RW Latency & E.	50ns & 3.7pJ/bit [37], [68]
Bitline Refresh Period & E.	32ms & 0.22pJ/bit [20], [80]
Wire & Link Model Parameters	**Values**
MCM PHY Areal Density	690 Gbits/mm^2 [6]
MCM PHY Beachfront Density	880 Gbits/mm [6]
Si. Interposer PHY Areal Density	1070 Gbits/mm^2 [6]
Si. Interposer PHY Beachfront Density	1780 Gbits/mm [6]
Die-to-Die Link Latency & E.	4ns & 0.55pJ/bit (<25mm) [62]
NoC Wire Latency & E.	50 ps/mm & 0.15pJ/bit/mm [39]
NoC Router Latency & E.	500ps & 0.1pJ/bit
I/O Die RX-TX Latency	20ns [77]
Off-Package Link E.	1.17pJ/bit (upto 80mm) [89]

this cost is not disclosed, we made an educated guess using public sources [33], [70]. We assume 7.5$/GB, which is more affordable than when HBM was released in 2017. One could expect this price to decrease over time as more vendors fabricate HBM or the process matures [47], [53], [61], [69].

V. RESULTS

Unlike dense-data workloads, data traversal algorithms have highly varying demands that depend on data size and structure (e.g., skewness), making the peak compute performance of current systems severely under-utilized. DCRA is an architecture for efficiently processing data traversal workloads with several pre-silicon options as well as packaging- and compile-time configurations that make this process cost-efficient.

We start in §V-A by demonstrating the benefits of the default options of DCRA such as the utilization of torus topology within and across chiplets. §V-B then analyze pre-silicon options, such as SRAM size per tile and PU frequency, where we can make tradeoffs depending on the target set of application domains. Next, §V-C analyzes pre-silicon options where decisions may be made depending on the skewness of dataset sparsity, such as PUs per tile, tiles per chiplet, and NoC frequency. In §V-D we evaluate packaging-time configurations such as the target-metric-dependent choice of including HBM or not on the chip package. In **??** we evaluate compile-time configurations, like the sizes of the tiles' input and output queues, and how the level of parallelization for a given dataset affects throughput per dollar and watt.

A. Default DCRA Options

Fig. 4 shows the performance improvement of DCRA with different NoC configurations with 64x64 tiles (across 16 chiplets) with 512KB SRAM per tile. Since at that parallelization level (2^{22} vertices across 2^{12} tiles) the network is the bottleneck, we observe a large performance impact when

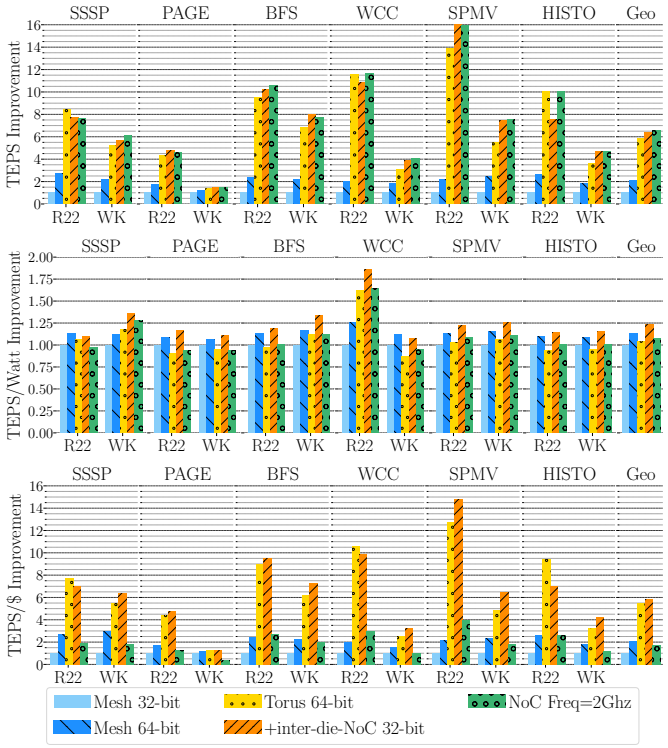

Fig. 4. Performance, energy efficiency, and performance per dollar improvements of different network choices over a baseline of a 32-bit 2D-Mesh. All configurations use 64x64 tiles (across 16 chiplets), with 512KB SRAM/tile.

increasing bisection bandwidth and reducing the diameter of the network. When doubling the width of the Mesh NoC, the performance also doubles, nearly equally across the board. The largest impact comes from using a Torus as a topology, 2.6× on geomean, but up to 8× for SPMV. This is expected since not only does Torus increase the bisection bandwidth and reduce the diameter of the network, but also improves the uniformity of the network traffic. This is aligned with what was observed by Dalorex, and it was the reason why we put effort into designing our reconfigurable network topology where the size of the folded torus can span one or multiple chiplets. However, the Torus is more power-hungry than the Mesh, and even though it increases throughput, the energy efficiency is 92% of the Mesh. This is overcome by using our 2-level hierarchical Torus topology where there is a torus that connects adjacent tiles (even across chiplets) and another torus that connects to one tile per chiplet (*inter-node-die*). This addition improves the energy efficiency by 19% and performance by 9% on geomean, over the regular Torus. In terms of cost efficiency, the DCRA hierarchical Torus has 5.1× higher throughput-per-dollar than the 32-bit Mesh on geomean, for this set of applications and datasets.

Finally, Fig. 4 also tests the performance of increasing the network frequency from 1GHz to 2GHz (as a double-pumped NoC [28], [87]). After the Torus addition, at this scaling level (64x64), the NoC is not the bottleneck anymore, and so the performance improvement is only 3% on geomean. However, it is interesting to show the energy and cost impact of this

double-pumped NoC. The reason why the cost increases three-fold is that we set in the area model that the NoC area doubles for this, but also the area of the IO dies on the chip package (need to handle twice the bandwidth). However, this type of NoC could render useful to parallelize huge graphs (potentially skewed) across larger systems, where the NoC is critical.

Default: We choose the hierarchical Torus as the default NoC topology for DCRA, as it provides the best performance per dollar and energy efficiency. While the 2Ghz NoC option would only render cost-efficient when the configuration or application benefits from extra bandwidth (e.g. when having more than one PU per tile §V-C).

B. Target Application Dependent Pre-Silicon Options

One of the main considerations for production systems is the range of application types a system will be serving. It is not uncommon for workloads that process sparse data to be processed alongside dense-data processing workloads or kernels. However, there are also target systems that mostly focus on sparse data processing [26]. While these application domains have different arithmetic-intensity and data-communication needs, there are several factors like SRAM size and maximum PU frequency where we can decide one option or another depending on whether the target application domain is sparse-only or also dense.

Fig. 5 shows the performance, energy efficiency, and performance per dollar improvements of different SRAM sizes and different numbers of tiles per DCRA chiplet, but the same number of tiles in total (1024, organized as 32x32). Since a DCRA chiplet is always attached to a single 8-channel HBM device, the number of tiles per chiplet determines the ratio of tiles per HBM channel (T/C). Since these sparse applications have a low arithmetic intensity, i.e., 1.44, 0.8, 1.8, 0.88, 1.52, 0.8 FLOPs/byte, for SSSP, PageRank, BFS, WCC, SPMV, and Histogram, respectively (nearly identical across these datasets), they require a large amount of memory bandwidth. As a result, Fig. 5 shows a strong performance increase with SRAM size, 2.6× on geomean when increasing from 64KB to 512KB, with the same chiplet configuration of 32x32 tiles. Note that these improvements are on geomean across two datasets sizes. While the RMAT-22 dataset (shortened as R22) has a data footprint ranging from 512MB for Histogram to 1GB for SPMV (0.5-1MB per tile), R25 has 8× larger footprint. We evaluated this to see the throughput per dollar difference when the dataset nearly fits on SRAM, and when it does not. Across applications, SSSP and SPMV, with the largest footprint, have the largest performance improvements. The case of Pagerank is special, as the barrier-synchronization at each epoch is causing low utilization towards the end of each epoch due to work imbalance (shown in Fig. 6).

The hit-rate of the data cache increases from 88% to 96% on geomean across datasets (from 81% to 95% considering only R25). Since the effective bandwidth of a tile is: $SRAM-bandwidth \times hit-rate + DRAM-bandwidth \times (1 - hit-rate)$, when the DRAM-bandwidth per tile is low (as it is shared across 128 tiles), the hit-rate has a large impact on the effective bandwidth. Changing the number of tiles per chiplet to 16x16 quadruples the DRAM bandwidth per tile, unleashing further performance improvements, 1.44× on geomean across datasets (1.7× considering only R25) when keeping the same SRAM size of 512KB. However, this configuration has nearly half the performance per dollar on geomean, due to the higher cost of having four times more HBM devices.

Default: Having observed these results, we choose the 512KB/tile and 32x32 tiles per chiplet as the default configuration of DCRA despite having a third of the performance for SSSP and SPMV for R25, because of three reasons. (1) the extra cost of more HBM only renders beneficial when the application is memory-bound and with a large data footprint per tile. However, this footprint decreases as we scale the parallelization, aiming for the fastest time-to-solution. Moreover, applications with higher arithmetic intensity will not benefit as much from this bandwidth, making the per-dollar performance worse. (2) having more tiles better amortizes the area of the PHY that connects to other tiles; its size, $255mm^2$, still achieves a good fabrication yield. (3) the SRAM size determines the minimal parallelization required for a given dataset for the chip integrations of DCRA that do not include HBM—making 512KB beneficial in that case too.

Targeting sparse-dense: If we were to tape out a different configuration for a deployment that would both use sparse and dense applications, having a 128 or 256KB SRAM would represent a better tradeoff, as 256KB achieves only 18% worse performance than the 512KB on geomean, and equal energy and cost efficiency. This 32x32 tile per die and 256KB case is the one depicted in Fig. 1.

C. Data- or Application-Dependent Pre-Silicon Options

For systems focusing on sparse data processing only, one important consideration is target-data properties such as skewness of sparsity. In single-data-owner execution schemes like Dalorex, this can create work-imbalance especially when increasing the parallelization level. This can be accommodated in tile-based systems as well by having multiple PUs per tile, and having them share an input queue (IQ) [13]. That way, the hotspots caused by dataset skewness are softened by a shared work queue across PUs, provided that the memory operations are atomic across the tile's PUs.

Fig. 6 evaluates three configurations with similar area and memory and network resources to try to measure only the impact of the tile granularity. The configuration of 4 PUs per tile has, thus, 4 times fewer tiles but 4 times larger. These experiments aim to isolate the impact of having more PUs sharing the same IQ, which leads to better load balancing. (Note that we do not model SRAM bank conflicts on the SRAM but we do consider larger SRAM access latencies for larger SRAM sizes at a rate of one extra nanosecond for each four-fold increase in SRAM size.) Pagerank benefits the most from work balancing, 2.5× with 16 PUs/tile over the baseline of 1 PU per tile, since this reduces the time wasted at the end of each epoch synchronization. The other barrier-less graph implementations or single-pass applications (Histogram and

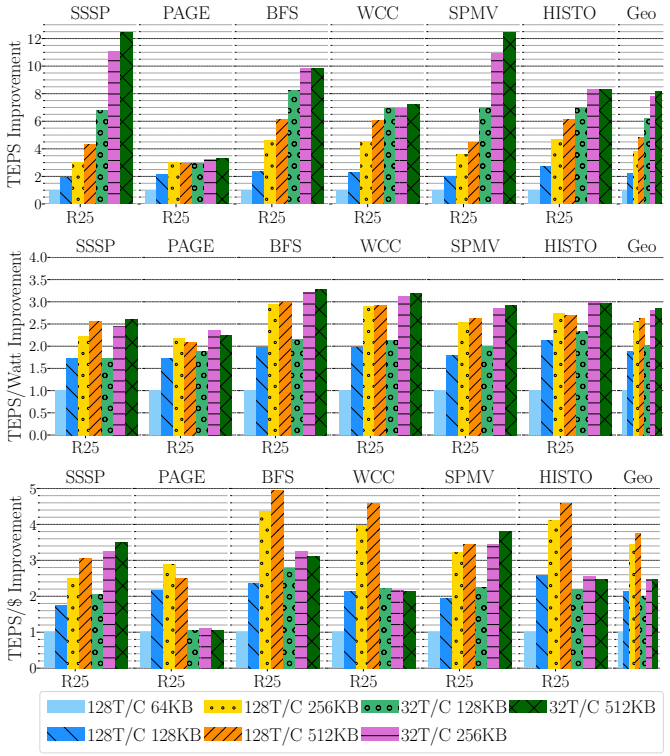

Fig. 5. Performance, energy efficiency, and performance per dollar improvements of different SRAM sizes and number of tiles per HBM channel, over a baseline of 64KB SRAM and 128 tiles per HBM channel (T/C). A DCRA chiplet is always attached to a single 8-channel HBM device, and thus, the number of tiles per chiplet determines the ratio of tiles per HBM channel.

SPMV) do not benefit as much. WCC is the application that experiences the largest differences across datasets, while the other applications have similar performance across datasets.

Default: A single PU per tile is DCRA default configuration because despite the benefits of balancing work better across PUs, if we aim to keep the same memory bandwidth per PU, the SRAM size grows too large and so does the distance between the PUs to any given memory bank, increasing both latency and power. This loss in energy efficiency outperforms the benefits for all barrier-less applications and only renders it interesting for Pagerank. In general, 4 PUs per tile results in a better balance of performance improvements and energy efficiency than the 16 PUs per tile configuration.

Targeting sparse-dense or synchronization-intensive applications: We could expect 4 PUs per tile to offer a better balance when targeting both sparse and dense applications and having opted for a smaller SRAM size per PU (e.g., 128 or 256KB). Alternatively, if we expect to focus on synchronization-intensive applications or running skewed datasets, 4 PUs per tile could be a better option.

Another option to adapt the chip to the application domain is supporting different PU frequencies. Dynamic-voltage frequency scaling (DVFS) [40] is a common technique to reduce power when the chip is too hot or when we want to save energy. We argue that it can be also used to adapt for different application domains, but it requires the PU to be designed and synthesized for a target maximum frequency. Thus, this

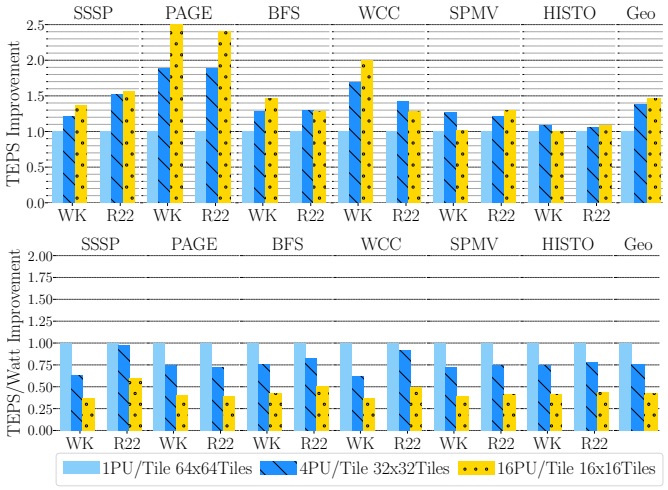

Fig. 6. Performance and energy efficiency improvements for configurations with 4 and 16 PUs per tile, over the baseline of 1 PU per tile. All three configurations have 64x64 PUs and the same number of chiplets, but decreasing number of tiles per chiplet, i.e., larger tiles. To make configurations have the same bisection bandwidth and total SRAM we scale up these resources as the number of tiles is reduced.

maximum frequency is a pre-silicon option.

Fig. 7 shows the performance improvement and energy-efficiency of DCRA with different PU frequencies for our sparse application domain. We already know that the dense workloads will scale well with PU frequency, as they have higher arithmetic intensity, but with this experiment, we want to see how these sparse applications behave.

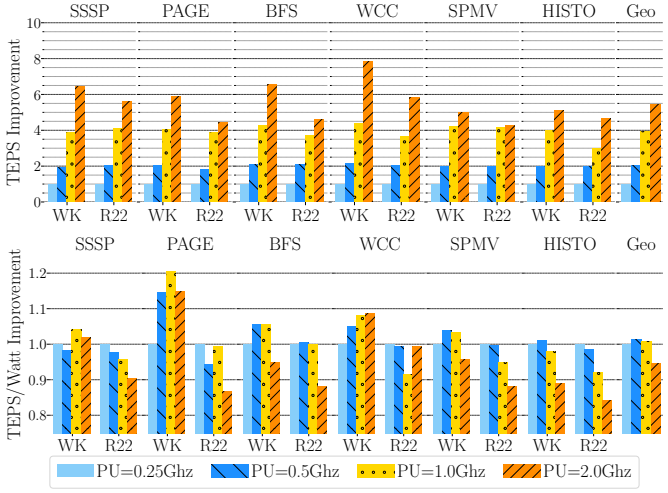

Fig. 7. Performance and energy efficiency improvements for different PU frequencies, over the baseline of 250 MHz. All configurations use 64x64 tiles, one PU per tile, and 512KB SRAM per tile.

Fig. 7 shows that the performance improvement is linear with PU frequency until the 1GHz point, where it starts to saturate for most applications: 2 Ghz achieves a geomean 38% improvement over 1 Ghz. It is noteworthy that WCC achieves full linear scaling up to 2 Ghz for R22, showing that it is not constrained by memory or the network on this scaling level (64x64 PUs).

Default and sparse-dense: The default maximum PU frequency of DCRA is 1 Ghz. However, if the target deployment considers the possibility of processing compute-bound applications, the 2 Ghz should be chosen, because the area overhead (assuming a pessimistic 50% area increase) in the overall DCRA die is 10%. The decrease in energy efficiency with the 2 Ghz version can be mitigate at runtime by using DVFS to reduce the frequency and thus power.

D. Target-metric-dependent Packaging-Time Configurations

Optimization of production systems for different target metrics such as fastest time-to-solution, energy or cost is an important factor both at pre-silicon as well as at packaging. At this moment, we have covered all of our pre-silicon options, and so we now focus on post-silicon configurations, particularly, on the tradeoffs of integrating HBM or not.

Fig. 8 shows the throughput-per-dollar and energy efficiency of Dalorex and DCRA using HBM, over a baseline of DCRA with SRAM only. For Dalorex's monolithic integration, only one chip fits in a wafer, and so we assume that for its cost.

Although Fig. 8 does not show absolute throughput, it is shown in Fig. 11 for R26. In terms of throughput-per-dollar, the configuration including HBM wins across the board, with a couple of exceptions. Note that this throughput-per-dollar results could change significantly if either HBM becomes cheaper (currently twice the cost of our selected 32x32-tile DCRA die with 512KB/tie) or another DRAM or 3D-stacked SRAM [78] technology becomes cheaper.

The DCRA-SRAM configuration wins in energy efficiency for R25 but not for R26. We found that the energy efficiency stays more or less stable across the parallelization levels (with some ups and downs). This trend is more clearly shown in Fig. 11, and the reasons are quite fascinating: (1) the energy of SRAM remains fairly constant across the parallelization levels, as SRAM banks are powered off when idle, as we do not need to stride arrays across them for a single PU/tile; (2) the HBM does it similarly, to the point where it is switched off when no longer used, and the SRAM becomes the main level of memory. The memory energy becomes less dominant over time as the data cache hit-rate increases; (3) the NoC energy increases at a similar rate at which the memory energy decreases. (4) the PU energy remains stable since in this task-based execution model, PUs do not run a main thread but rather react to incoming tasks provided by the router (which triggers the clock gate of the PU.)

Fig. 9 shows the breakdown of the energy used by PUs, memory, and NoC for DCRA-SRAM and DCRA-HBM the configurations shown in Fig. 8. PUs use a small fraction in both cases, partly because PUs are powered off when idle.

DCRA-SRAM energy: Since this version uses 16× more tiles than DCRA-HBM, it spends more energy on wires and routers (Fig. 11 shows the increase in message routing hops).

DCRA-HBM energy: As we saw in Fig. 5, the HBM integration saturates the memory controller bandwidth, which makes the energy usage dominated by DRAM for small parallelization levels.

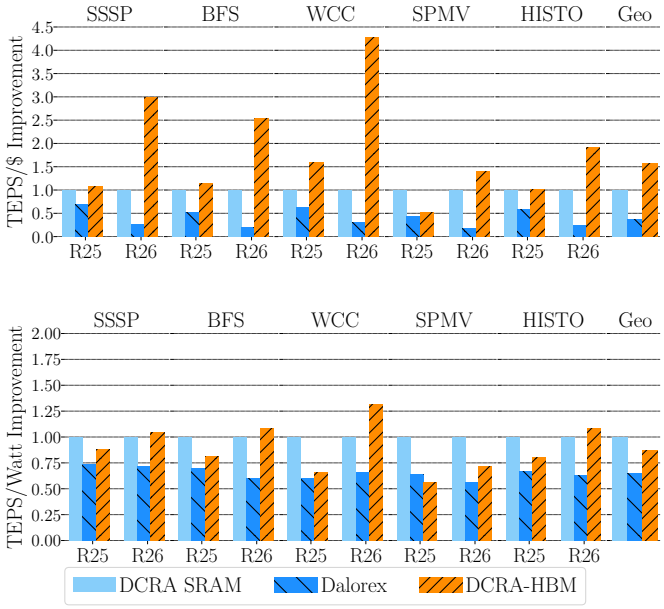

Fig. 8. Comparison of throughput/$ and energy efficiency of Dalorex and DCRA with HBM, over a baseline of DCRA with SRAM only. Dalorex uses 2MB SRAM per tile ([66]) while DCRA uses 512KB; DCRA-HBM integrates a 8GB HBM2 device per chiplet of 32x32 tiles, providing a memory-per-PU ratio of 8MB/PU. We parallelize the datasets across the smallest configuration for which it fits on chip. For R25 that size is 32x32 for DCRA-HBM, 64x64 for Dalorex and 128x128 for DCRA-SRAM, while for R26 we use our next scaling step with 4 times as many tiles.

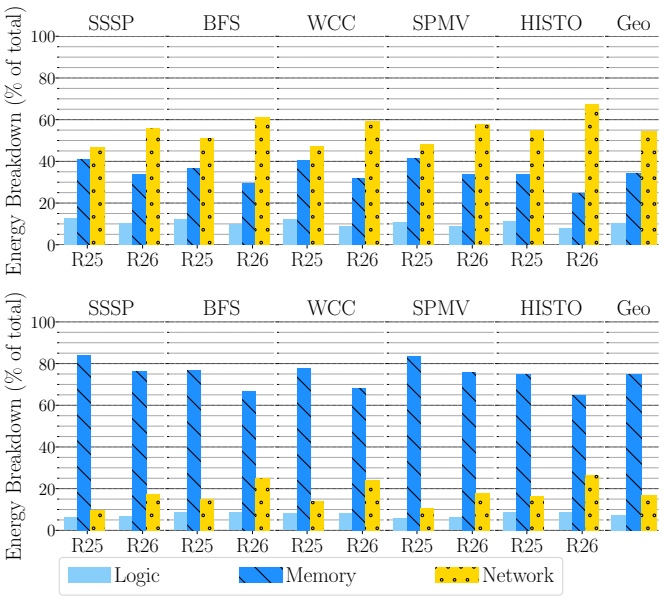

Fig. 9. Breakdown of the energy consumed by computing logic, memory, and NoC communication (including routing and wire energy), for Fig.8's DCRA-SRAM (top) and DCRA-HBM (bottom). The Y-axis shows the % of the total energy spent on each component.

E. Compile-time Configurations

Once a DCRA system is taped-out and packaged according to the considerations we described above, there remain several compile-time configurable features. These include (a) the input and output queue sizes, (b) the portion of the tile's SRAM

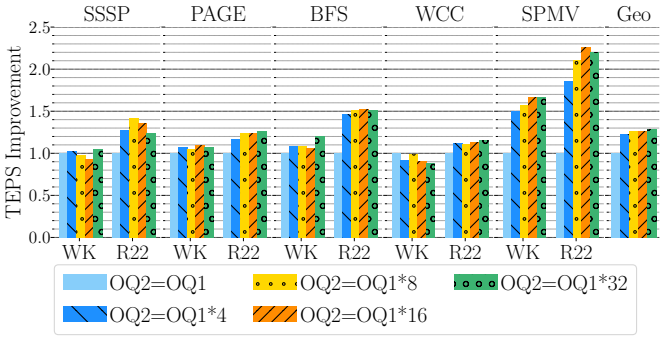

Fig. 10. Performance improvement of increasing the size of the OQ2 over that of OQ1, normalized over the baseline of OQ2=OQ1. OQ1 size is 12 task invocation messages. All configurations use 64x64 tiles, one PU/tile, and 512KB/tile. We do not show Histogram because it only has two tasks (one OQ between them).

that is used as cache and a scratchpad, (c) the PU and NoC frequency (up to the maximum frequency considered during design), and (d) the network topology.

SRAM and frequency were discussed in the sections above; we would like to add that the part of the SRAM that is not used for caching space is invested for program memory, data arrays (mapped as a scratchpad), or larger queues. This last option is the one we focus on in this section, which can have a different impact across applications and datasets.

Queue sizes are especially important in the task-based execution model that we use because the parallelism achieved is equal to the number of PUs with tasks ready to execute in the IQs. To provide enough work for others, the OQs should be sufficient. Fig. 10 shows the performance gained from increasing the size of the OQ of the task that pushes the vertex updates for the graph applications (or the updates to the output array in SPMV and Histogram) in the Dalorex implementation that we use. We increase the size of this OQ2 over the size of the OQ for the first task (OQ1), which pushes the lookup for the edge list. Because the ratio of the number of task invocations that go to each of these tasks is similar to the average number of edges per vertex, we can use it to choose the size of OQ2. Fig. 10 showcases that the performance improvement is more significant for R22 than for WK (which only improves for SPMV). R22 has 32 edges per vertex on average, while WK has 25. We observe that this ratio is also very different across applications, so this is another learning we can use to guide this decision. A more thorough study varying the size of OQ1, as well as that of the IQs for a particular application, would help optimize this software configuration parameter.

Parallelization Level: Another compile-time decision is, of course, the size of the subgrid of tiles that we use to parallelize the application, from the total number of tiles that the system has. This is related to the NoC topology because the configurable routers described in Fig. 2 need to be set to wrap around accordingly. The 2D Torus can be as large as a computing node (a board of one or more packages). Beyond that level, the nodes are connected via the cluster-level network.

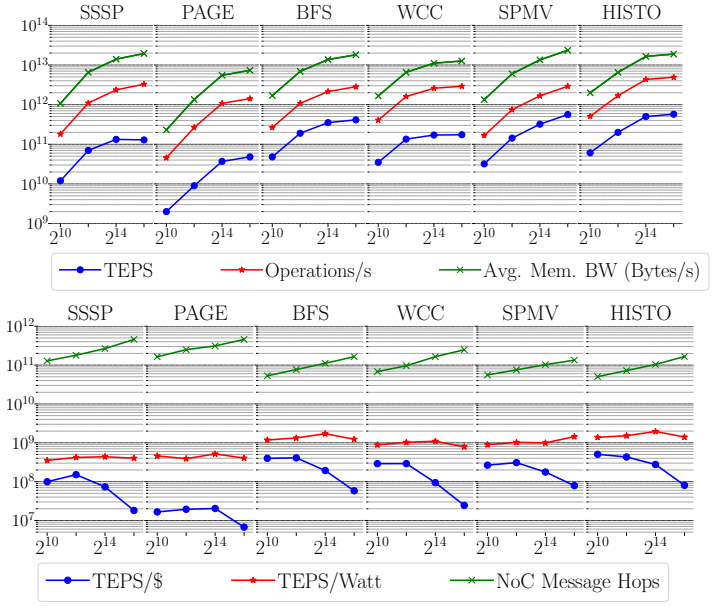

Fig. 11. Throughput in operations and floating point operations (FLOPs) per second, and the average on-chip memory bandwidth needed to achieve that. The X-axis is the size of the DCRA grid used when analyzing strong scaling R26, ranging from 1024 to over 64K PUs. The bottom plot shows throughput as a function of power and cost, in addition to the total number of message hops during the execution.

Evaluating how the performance scales with the size of the node can help us understand the tradeoffs of using a larger or smaller node. Fig. 11 (top) shows the throughput and the average memory bandwidth needed to achieve that, for R26. The gap between the floating-point and regular operations shows the characteristics of the application, as well as the gap between the memory bandwidth and the throughput. Analyzing Fig. 11, we learn the following principles that guide our decision tree of Fig. 12.

Strong scaling for faster time-to-solution: We parallelize a dataset with 2^{26} vertices with up to 2^{16} tiles. At that level of parallelization, the entire dataset fits on the 32GB of SRAM across all 16 chips. Fig. 11 (bottom, green line) shows that the total number of message hops increases as we parallelize across more tiles. This means that to do the same amount of total work, the PUs need to communicate more. Although the throughput increases until this point, we do not expect it to scale much further without applying communication-mitigation techniques like graph partitioning or reduction trees.

Throughput-per-watt remains stable until 64K tiles: Fig. 11 (bottom, red line) shows that the throughput-per-watt remains stable across different parallelization levels. From Fig. 9 we observe that as we scale out, the energy consumption shifts from HBM towards the NoC as the number of tiles increases. This is because the HBM device is powered down when the entire dataset fits in SRAM (at the last scaling point); SRAM then behaves as the main memory level of the chip. Beyond this point, we can only expect the efficiency to drop, as the traffic continues to increase with the number of tiles.

Throughput-per-dollar likes small grids: Fig. 11 (bottom, blue line) shows that if throughput-per-dollar is the target

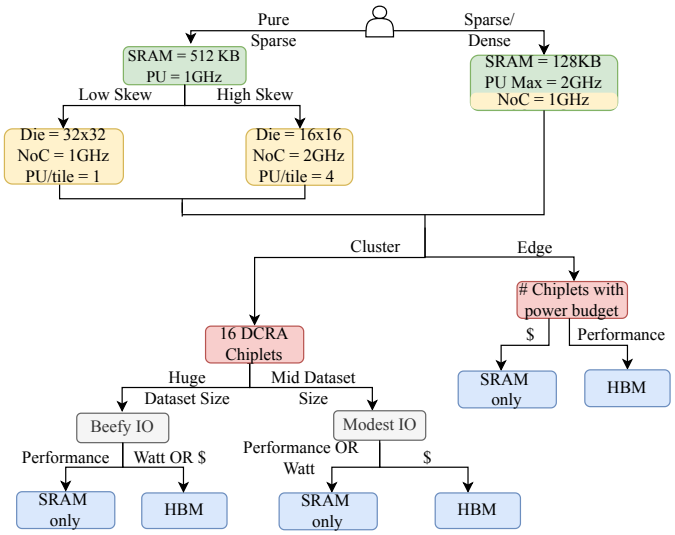

Fig. 12. Based on the target deployment (pure-sparse applications vs a balance between sparse and dense) we decide whether to have a max operating frequency of 1Ghz or 2Ghz, and whether to employ 512KB of SRAM or 128KB, respectively. In addition, a high-skew target datasets (that create execution hotspots), choosing four PUs per tile consuming from the same task queue can help mitigate that (in that case, we double the NoC frequency to balance the injection throughput to the tiles). Depending on whether the target is an HPC cluster or an edge deployment, HBM and SRAM can be integrated differently. The HPC tradeoffs were discussed above. However, we would expect the edge-device tradeoffs to be different. In the HPC cluster, we expect that we can always scale out to use more chips, whereas in the edge we may be more constrained by size and power. At the edge, if we care most about performance we should integrate HBM, and the cost-efficient alternative is to use SRAM, and when the dataset does not fit we have to resort to swap back and forth from DDR or flash. Additionally, the IO bandwidth between chips and outside the node could be provisioned depending on whether the cluster is expecting to run huge graphs collectively, or every node processing its graph. For that, graph partitioning techniques play an important role.

metric, staying within 2^{12} tiles (64x64) is the best option. This is because we stop seeing linear speedups beyond 2^{12}, while the cost always grows linearly. Note that while the TEPS/$ trend is determined by the sublinear scaling, the absolute values depend on the cost. While we can power down HBM to save energy, we cannot avoid the HBM cost when we do not use it. Thus, the 2^{16} datapoint, for which the dataset fits on SRAM, would achieve a higher TEPS/$ than the one displayed if we had chosen an integration without HBM. However, as we saw in Fig. 8, TEPS/$ for SRAM-only at 2^{16} is still lower than staying within the 2^{12} scaling point with HBM.

VI. Conclusion

DCRA proposes a chiplet-based architecture that allows making key design decisions, such as on-chip memory and off-chip interconnect, post-silicon. This enables the same DCRA design to be mass-produced (saving NRE costs) and later integrated differently to create products with distinct target metrics. We have discussed the configurations that warrant different targets—some of which are evaluated in §V. To enable such configurations, DCRA includes our design of a 2D-Torus NoC topology which can span across a varying number of dies as its size is defined in software.

We put much effort into designing a chiplet-based architecture that is highly configurable, allowing it to potentially be applied to graph and sparse application domains as well as others to optimize the chip at packaging time for different target metrics, such as time-to-solution, energy efficiency, or cost. Fig. 12 summarizes our guidelines for integrating DCRA for different use cases and constraints.

We have created the scripts for the artifact evaluation of the experiments presented in this paper. Our repository [65] includes the simulation framework implementing DCRA; we are planning to document it with the publication of this paper to allow researchers to explore further configurations and optimizations of DCRA.

References

[1] Sriram Aananthakrishnan, Nesreen K Ahmed, Vincent Cave, Marcelo Cintra, Yigit Demir, Kristof Du Bois, Stijn Eyerman, Joshua B Fryman, Ivan Ganev, Wim Heirman, et al. Piuma: programmable integrated unified memory architecture. *arXiv preprint arXiv:2010.06277*, 2020.

[2] Maleen Abeydeera and Daniel Sanchez. Chronos: Efficient speculative parallelism for accelerators. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1247–1262, 2020.

[3] Bilge Acun, Benjamin Lee, Fiodar Kazhamiaka, Kiwan Maeng, Udit Gupta, Manoj Chakkaravarthy, David Brooks, and Carole-Jean Wu. Carbon explorer: A holistic framework for designing carbon aware datacenters. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 118–132, 2023.

[4] Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi. A scalable processing-in-memory accelerator for parallel graph processing. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, pages 105–117, 2015.

[5] AMD. AMD rome, 2018. https://en.wikichip.org/wiki/amd/cores/rome.

[6] Shahab Ardalan, Bapi Vinnikota, Tawfik Arabi, and Elad Alon. What is the right die-to-die interface? a comparison study, 2022. https://www.opencompute.org/events/past-events/hipchips-chiplet-workshop-isca-conference.

[7] Jonathan Balkind, Katie Lim, Fei Gao, Jinzheng Tu, David Wentzlaff, Michael Schaffner, Florian Zaruba, and Luca Benini. OpenPiton+Ariane: The first open-source, SMP Linux-booting RISC-V system scaling from one to many cores. In *Third Workshop on Computer Architecture Research with RISC-V, CARRV*, volume 19, 2019.

[8] Cerebras Systems Inc. The second generation wafer scale engine. https://cerebras.net/wp-content/uploads/2021/04/Cerebras-CS-2-Whitepaper.pdf.

[9] Vidya A Chhabria, Chetan Choppali Sudarshan, Sarma Vrudhula, and Sachin S Sapatnekar. Towards sustainable computing: Assessing the carbon footprint of heterogeneous systems. *arXiv preprint arXiv:2306.09434*, 2023.

[10] Jack Choquette and Wish Gandhi. Nvidia A100 GPU: Performance & innovation for GPU computing. In *2020 IEEE Hot Chips 32 Symposium (HCS)*, pages 1–43. IEEE Computer Society, 2020.

[11] Chia Chen Chou, Aamer Jaleel, and Moinuddin K Qureshi. Cameo: A two-level memory organization with capacity of main memory and flexibility of hardware-managed cache. In *47th IEEE/ACM International Symposium on Microarchitecture*, pages 1–12. IEEE, 2014.

[12] Emilio G Cota, Paolo Mantovani, and Luca P Carloni. Exploiting private local memories to reduce the opportunity cost of accelerator integration. In *Proceedings of the 2016 International Conference on Supercomputing*, pages 1–12, 2016.

[13] Vidushi Dadu, Sihao Liu, and Tony Nowatzki. Polygraph: Exposing the value of flexibility for graph processing accelerators. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 595–608. IEEE, 2021.

[14] Murali Emani, Venkatram Vishwanath, Corey Adams, Michael E Papka, Rick Stevens, Laura Florescu, Sumti Jairath, William Liu, Tejas Nama, and Arvind Sujeeth. Accelerating scientific applications with sambanova reconfigurable dataflow architecture. *Computing in Science & Engineering*, 23(2):114–119, 2021.

[15] Esperanto Technologies. Esperanto's et-minion on-chip RISC-V cores. https://www.esperanto.ai/technology/.

[16] Kim Eun-jin. Samsung and SK Hynix Enjoy a Rush of Orders for New Memories, 2023. https://www.businesskorea.co.kr/news/articleView.html?idxno=109380.

[17] Kai Feng, Yaoyao Ye, and Jiang Xu. A formal study on topology and floorplan characteristics of mesh and torus-based optical networks-on-chip. *Microprocessors and Microsystems*, 37(8):941–952, 2013.

[18] Fei Gao, Ting-Jung Chang, Ang Li, Marcelo Orenes-Vera, Davide Giri, Paul J. Jackson, August Ning, Georgios Tziantzioulis, Joseph Zuckerman, Jinzheng Tu, Kaifeng Xu, Grigory Chirkov, Gabriele Tombesi, Jonathan Balkind, Margaret Martonosi, Luca Carloni, and David Wentzlaff. Decades: A 67mm2, 1.46tops, 55 giga cache-coherent 64-bit RISC-V instructions per second, heterogeneous manycore soc with 109 tiles including accelerators, intelligent storage, and efpga in 12nm finfet. In *Custom Integrated Circuits Conference (CICC)*, pages 1–2, 2023.

[19] Gerasimos Gerogiannis, Serif Yesil, Damitha Lenadora, Dingyuan Cao, Charith Mendis, and Josep Torrellas. Spade: A flexible and scalable accelerator for spmm and sddmm. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–15, 2023.

[20] Saugata Ghose, Abdullah Giray Yaglikçi, Raghav Gupta, Donghyuk Lee, Kais Kudrolli, William X Liu, Hasan Hassan, Kevin K Chang, Niladrish Chatterjee, Aditya Agrawal, et al. What your dram power models are not telling you: Lessons from a detailed experimental study. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(3):1–41, 2018.

[21] Davide Giri, Kuan-Lin Chiu, Giuseppe Di Guglielmo, Paolo Mantovani, and Luca P. Carloni. ESP4ML: Platform-based design of systems-on-chip for embedded machine learning. In *DATE*. IEEE Press, 2020.

[22] Wilfred Gomes, Altug Koker, Pat Stover, Doug Ingerly, Scott Siers, Srikrishnan Venkataraman, Chris Pelto, Tejas Shah, Amreesh Rao, Frank O'Mahony, et al. Ponte vecchio: A multi-tile 3d stacked processor for exascale computing. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 65, pages 42–44. IEEE, 2022.

[23] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. Chasing carbon: The elusive environmental footprint of computing. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 854–867. IEEE, 2021.

[24] Linley Gwennap. Groq rocks neural networks. *Microprocessor Report, Tech. Rep., jan*, 2020.

[25] Tae Jun Ham, Lisa Wu, Narayanan Sundaram, Nadathur Satish, and Margaret Martonosi. Graphicionado: A high-performance and energy-efficient accelerator for graph analytics. In *Proceedings of the 49th Annual International Symposium on Microarchitecture*, MICRO, 2016.

[26] William Harrod. Agile: The future of data centric computing, 2022. https://www.youtube.com/watch?v=qIM_RBXX6O0.

[27] High-Bandwidth Memory (HBM), 2015. https://www.amd.com/Documents/High-Bandwidth-Memory-HBM.pdf.

[28] Yatin Hoskote, Sriram Vangal, Arvind Singh, Nitin Borkar, and Shekhar Borkar. A 5-ghz mesh interconnect for a teraflops processor. *IEEE micro*, 27(5):51–61, 2007.

[29] Intel. Intel Kaby Lake G, 2018. https://en.wikichip.org/wiki/intel/cores/kaby_lake_g.

[30] ISCA Submission 210. DCRA Simulation Framework and Artifacts, 2023. https://github.com/francisca210/dcra.git.

[31] JEDEC. Standard high bandwidth memory specification jesd235a, 2015.

[32] Mark C. Jeffrey, Suvinay Subramanian, Cong Yan, Joel Emer, and Daniel Sanchez. A scalable architecture for ordered parallelism. In *Proceedings of the 48th International Symposium on Microarchitecture*, MICRO-48, page 228–241, New York, NY, USA, 2015. Association for Computing Machinery.

[33] Scotten W. Jones. Lithovision: Economics in the 3d era. https://semiwiki.com/wp-content/uploads/2020/03/Lithovision-2020.pdf.

[34] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12, 2017.

[35] George Karypis and Vipin Kumar. Metis: A software package for partitioning unstructured graphs. *Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Version*, 4(0), 1998.

[36] George Karypis, Kirk Schloegel, and Vipin Kumar. Parmetis: Parallel graph partitioning and sparse matrix ordering library. 1997.

[37] Dae-Hyun Kim, Byungkyu Song, Hyun-a Ahn, Woongjoon Ko, Sunggeun Do, Seokjin Cho, Kihan Kim, Seung-Hoon Oh, Hye-Yoon Joo, Geuntae Park, Jin-Hun Jang, Yong-Hun Kim, Donghun Lee, Jaehoon Jung, Yongmin Kwon, Youngjae Kim, Jaewoo Jung, Seongil O, Seoulmin Lee, Jaeseong Lim, Junho Son, Jisu Min, Haebin Do, Jaejun Yoon, Isak Hwang, Jinsol Park, Hong Shim, Seryeong Yoon, Dongyeong Choi, Jihoon Lee, Soohan Woo, Eunki Hong, Junha Choi, Jae-Sung Kim, Sangkeun Han, Jongmin Bang, Bokgue Park, Janghoo Kim, Seouk-Kyu Choi, Gong-Heum Han, Yoo-Chang Sung, Won-Il Bae, Jeong-Don Lim, Seungjae Lee, Changsik Yoo, Sang Joon Hwang, and Jooyoung Lee. A 16gb 9.5gb/s/pin lpddr5x sdram with low-power schemes exploiting dynamic voltage-frequency scaling and offset-calibrated readout sense amplifiers in a fourth generation 10nm dram process. In *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, volume 65, pages 448–450, 2022.

[38] John Kim, Wiliam J Dally, Steve Scott, and Dennis Abts. Technology-driven, highly-scalable dragonfly topology. *ACM SIGARCH Computer Architecture News*, 36(3):77–88, 2008.

[39] Seongguk Kim, Subin Kim, Kyungjun Cho, Taein Shin, Hyunwook Park, Daehwan Lho, Shinyoung Park, Kyungjune Son, Gapyeol Park, and Joungho Kim. Processing-in-memory in high bandwidth memory (pim-hbm) architecture with energy-efficient and low latency channels for high bandwidth system. In *2019 IEEE 28th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, pages 1–3, 2019.

[40] Wonyoung Kim, Meeta S Gupta, Gu-Yeon Wei, and David Brooks. System level analysis of fast, per-core dvfs using on-chip switching regulators. In *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, pages 123–134. IEEE, 2008.

[41] Simon Knowles. Graphcore. In *2021 IEEE Hot Chips 33 Symposium (HCS)*, pages 1–25. IEEE, 2021.

[42] Snehasish Kumar, Hongzhou Zhao, Arrvindh Shriraman, Eric Matthews, Sandhya Dwarkadas, and Lesley Shannon. Amoeba-cache: Adaptive blocks for eliminating waste in the memory hierarchy. In *45th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 376–388. IEEE, 2012.

[43] Kartik Lakhotia, Maciej Besta, Laura Monroe, Kelly Isham, Patrick Iff, Torsten Hoefler, and Fabrizio Petrini. Polarfly: A cost-effective and flexible low-diameter topology. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15. IEEE, 2022.

[44] John H Lau. Status and outlooks of flip chip technology. *IPC EXPO Proceedings, February 2017*, pages 1–20, 2017.

[45] Page Lawrence, Brin Sergey, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[46] Chang-Chi Lee, Cp Hung, Calvin Cheung, Ping-Feng Yang, Chin-Li Kao, Dao-Long Chen, Meng-Kai Shih, Chien-Lin Chang Chien, Yu-Hsiang Hsiao, Li-Chieh Chen, Michael Su, Michael Alfano, Joe Siegel, Julius Din, and Bryan Black. An overview of the development of a gpu with integrated hbm on silicon interposer. In *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, pages 1439–1444, 2016.

[47] Dong Uk Lee, Ho Sung Cho, Jihwan Kim, Young Jun Ku, Sangmuk Oh, Chul Dae Kim, Hyun Woo Kim, Woo Young Lee, Tae Kyun Kim, Tae Sik Yun, et al. 22.3 a 128gb 8-high 512gb/s hbm2e dram with a pseudo quarter bank structure, power dispersion and an instruction-based at-speed pmbist. In *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 334–336. IEEE, 2020.

[48] Charles E Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE transactions on Computers*, 100(10):892–901, 1985.

[49] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Reseach (JMLR)*, 11:985–1042, March 2010.

[50] Cedric Lichtenau, Alper Buyuktosunoglu, Ramon Bertran, Peter Figuli, Christian Jacobi, Nikolaos Papandreou, Haris Pozidis, Anthony Saporito, Andrew Sica, and Elpida Tzortzatos. AI accelerator on IBM telum processor: industrial product. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pages 1012–1028, 2022.

[51] Ravi Mahajan, Robert Sankman, Neha Patel, Dae-Woo Kim, Kemal Aygun, Zhiguo Qian, Yidnekachew Mekonnen, Islam Salama, Sujit

Sharan, Deepti Iyengar, et al. Embedded multi-die interconnect bridge (emib)–a high density, high bandwidth packaging interconnect. In *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, pages 557–565. IEEE, 2016.

[52] Aninda Manocha, Tyler Sorensen, Esin Tureci, Opeoluwa Matthews, Juan L Aragón, and Margaret Martonosi. Graphattack: Optimizing data supply for graph applications on in-order multicore architectures. *ACM Transactions on Architecture and Code Optimization (TACO)*, 18(4):1–26, 2021.

[53] Micron. High Bandwidth Memory with ECC, 2018. https://media-www.micron.com/-/media/client/global/documents/products/data-sheet/dram/hbm2e/8gb_and_16gb_hbm2e_dram.pdf.

[54] MooreElite. Die yield calculator. https://isine.com/resources/die-yield-calculator/.

[55] Richard C. Murphy, Kyle B. Wheeler, Brian W. Barrett, and James A. Ang. Introducing the Graph 500. http://www.graph500.org/specifications, 2010.

[56] Samuel Naffziger, Noah Beck, Thomas Burd, Kevin Lepak, Gabriel H. Loh, Mahesh Subramony, and Sean White. Pioneering chiplet technology and design for the amd epyc™ and ryzen™ processor families. In *Proceedings of the 48th Annual International Symposium on Computer Architecture*, ISCA '21, page 57–70. IEEE Press, 2021.

[57] Nevine Nassif, Ashley O Munch, Carleton L Molnar, Gerald Pasdast, Sitaraman V Lyer, Zibing Yang, Oscar Mendoza, Mark Huddart, Srikrishnan Venkataraman, Sireesha Kandula, et al. Sapphire rapids: The next-generation intel xeon scalable processor. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 65, pages 44–46. IEEE, 2022.

[58] Quan M Nguyen and Daniel Sanchez. Pipette: Improving core utilization on irregular applications through intra-core pipeline parallelism. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 596–608. IEEE, 2020.

[59] Quan M. Nguyen and Daniel Sanchez. Fifer: Practical acceleration of irregular applications on reconfigurable architectures. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21, page 1064–1077, New York, NY, USA, 2021. Association for Computing Machinery.

[60] August Ning, Georgios Tziantzioulis, and David Wentzlaff. Supply chain aware computer architecture. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–15, 2023.

[61] Chi-Sung Oh, Ki Chul Chun, Young-Yong Byun, Yong-Ki Kim, So-Young Kim, Yesin Ryu, Jaewon Park, Sinho Kim, Sanguhn Cha, Donghak Shin, et al. 22.1 a 1.1 v 16gb 640gb/s hbm2e dram with a data-bus window-extension technique and a synergetic on-die ecc scheme. In *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 330–332. IEEE, 2020.

[62] Open Compute Group. Bunch of wires phy specification. https://opencomputeproject.github.io/ODSA-BoW/bow_specification.html.

[63] Marcelo Orenes-Vera, Aninda Manocha, Jonathan Balkind, Fei Gao, Juan L Aragón, David Wentzlaff, and Margaret Martonosi. Tiny but mighty: designing and realizing scalable latency tolerance for manycore socs. In *ISCA*, pages 817–830, 2022.

[64] Marcelo Orenes-Vera, Ilya Sharapov, Robert Schreiber, Mathias Jacquelin, Philippe Vandermersch, and Sharan Chetlur. Wafer-scale fast fourier transforms. In *Proceedings of the 37th International Conference on Supercomputing*, ICS '23, page 180–191, New York, NY, USA, 2023. Association for Computing Machinery.

[65] Marcelo Orenes-Vera, Esin Tureci, Margaret Martonosi, and David Wentzlaff. DCRA simulation framework and artifacts, 2023. https://github.com/morenes/dcra.git.

[66] Marcelo Orenes-Vera, Esin Tureci, David Wentzlaff, and Margaret Martonosi. Dalorex: A data-local program execution and architecture for memory-bound applications. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 718–730. IEEE, 2023.

[67] Muhammet Mustafa Ozdal, Serif Yesil, Taemin Kim, Andrey Ayupov, John Greth, Steven Burns, and Ozcan Ozturk. Energy efficient architecture for graph analytics accelerators. *ACM SIGARCH Computer Architecture News*, 44(3):166–177, 2016.

[68] Mike O'Connor, Niladrish Chatterjee, Donghyuk Lee, John Wilson, Aditya Agrawal, Stephen W Keckler, and William J Dally. Fine-grained dram: Energy-efficient dram for extreme bandwidth systems. In *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 41–54. IEEE, 2017.

[69] Myeong-Jae Park, Ho Sung Cho, Tae-Sik Yun, Sangjin Byeon, Young Jun Koo, Sangsic Yoon, Dong Uk Lee, Seokwoo Choi, Jihwan Park, Jinhyung Lee, et al. A 192-gb 12-high 896-gb/s hbm3 dram with a tsv auto-calibration scheme and machine-learning-based layout optimization. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 65, pages 444–446. IEEE, 2022.

[70] Andy Patrizio. High-bandwidth memory (hbm) delivers impressive performance gains. https://semiengineering.com/whats-next-for-high-bandwidth-memory/.

[71] J Thomas Pawlowski. Hybrid memory cube (hmc). In *2011 IEEE Hot Chips 23 Symposium (HCS)*, pages 1–24. IEEE, 2011.

[72] François Pellegrini. Scotch and pt-scotch graph partitioning software: an overview. *Combinatorial Scientific Computing*, pages 373–406, 2012.

[73] Gilead Posluns, Yan Zhu, Guowei Zhang, and Mark C. Jeffrey. A scalable architecture for reprioritizing ordered parallelism. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, page 437–453, New York, NY, USA, 2022. Association for Computing Machinery.

[74] Shafiur Rahman, Nael Abu-Ghazaleh, and Rajiv Gupta. Graphpulse: An event-driven hardware accelerator for asynchronous graph processing. In *2020 53rd Annual IEEE/ACM Symposium on Microarchitecture (MICRO)*, pages 908–921. IEEE, 2020.

[75] Agam Shah. Chipmakers Looking at New Architecture to Drive Computing Ahead, 2022. https://www.hpcwire.com/2022/11/23/chipmakers-looking-at-new-architecture-to-drive-computing-ahead/.

[76] Yakun Sophia Shao, Jason Clemons, Rangharajan Venkatesan, Brian Zimmer, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, Stephen G. Tell, Yanqing Zhang, William J. Dally, Joel Emer, C. Thomas Gray, Brucek Khailany, and Stephen W. Keckler. Simba: Scaling deep-learning inference with multi-chip-module-based architecture. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '52, page 14–27, New York, NY, USA, 2019. Association for Computing Machinery.

[77] Debendra Das Sharma. Pci express 6.0 specification: A low-latency, high-bandwidth, high-reliability, and cost-effective interconnect with 64.0 gt/s pam-4 signaling. *IEEE Micro*, 41(1), 2020.

[78] Kota Shiba, Tatsuo Omori, Kodai Ueyoshi, Shinya Takamaeda-Yamazaki, Masato Motomura, Mototsugu Hamada, and Tadahiro Kuroda. A 96-mb 3d-stacked sram using inductive coupling with 0.4-v transmitter, termination scheme and 12: 1 serdes in 40-nm cmos. *IEEE Transactions on Circuits and Systems I*, 68(2):692–703, 2020.

[79] George M. Slota, Sivasankaran Rajamanickam, and Kamesh Madduri. BFS and coloring-based parallel algorithms for strongly connected components and related problems. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium, Phoenix, AZ, USA, May 19-23, 2014*, pages 550–559. IEEE Computer Society, 2014.

[80] Kyomin Sohn, Won-Joo Yun, Reum Oh, Chi-Sung Oh, Seong-Young Seo, Min-Sang Park, Dong-Hak Shin, Won-Chang Jung, Sang-Hoon Shin, Je-Min Ryu, Hye-Seung Yu, Jae-Hun Jung, Hyunui Lee, Seok-Yong Kang, Young-Soo Sohn, Jung-Hwan Choi, Yong-Cheol Bae, Seong-Jin Jang, and Gyoyoung Jin. A 1.2 v 20 nm 307 gb/s hbm dram with at-speed wafer-level io test scheme and adaptive refresh considering temperature distribution. *IEEE Journal of Solid-State Circuits*, 52(1):250–260, 2017.

[81] Dylan Stow, Yuan Xie, Taniya Siddiqua, and Gabriel H. Loh. Cost-effective design of scalable high-performance systems using active and passive interposers. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 728–735, 2017.

[82] John A Stratton, Christopher Rodrigues, I-Jui Sung, Nady Obeid, Li-Wen Chang, Nasser Anssari, Geng Daniel Liu, and W-m Hwu. Parboil: A revised benchmark suite for scientific and commercial throughput computing. Technical Report IMPACT-12-01, University of Illinois at Urbana-Champaign, 2012.

[83] Raja Swaminathan and John Wuu. Chiplet's march to amd 3d v-cache and beyond, 2022. https://www.opencompute.org/events/past-events/hipchips-chiplet-workshop-isca-conference.

[84] Nishil Talati, Kyle May, Armand Behroozi, Yichen Yang, Kuba Kaszyk, Christos Vasiladiotis, Tarunesh Verma, Lu Li, Brandon Nguyen, Jiawen Sun, John Magnus Morton, Agreen Ahmadi, Todd Austin, Michael O'Boyle, Scott Mahlke, Trevor Mudge, and Ronald Dreslinski. Prodigy: Improving the memory latency of data-indirect irregular workloads using hardware-software co-design. In *2021 IEEE International Symposium*

on High-Performance Computer Architecture (HPCA), pages 654–667. IEEE, 2021.

[85] Emil Talpes, Douglas Williams, and Debjit Das Sarma. Dojo: The microarchitecture of tesla exa-scale computer. In *2022 IEEE Hot Chips 34 Symposium*, pages 1–28. IEEE Computer Society, 2022.

[86] Tianqi Tang and Yuan Xie. Cost-aware exploration for chiplet-based architecture with advanced packaging technologies. *arXiv preprint arXiv:2206.07308*, 2022.

[87] Sriram R Vangal, Jason Howard, Gregory Ruhl, Saurabh Dighe, Howard Wilson, James Tschanz, David Finan, Arvind Singh, Tiju Jacob, Shailendra Jain, et al. An 80-tile Sub-100-W teraflops processor in 65-nm CMOS. *IEEE Journal of solid-state circuits*, 43(1):29–41, 2008.

[88] Tianrui Wei, Nazerke Turtayeva, Marcelo Orenes-Vera, Omkar Lonkar, and Jonathan Balkind. Cohort: Software-oriented acceleration for heterogeneous socs. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS 2023, page 105–117, New York, NY, USA, 2023. Association for Computing Machinery.

[89] John Wilson. High-bandwidth density, energy-efficient, short-reach signaling that enables massively scalable parallelism, 2022. https://www.opencompute.org/events/past-events/hipchips-chiplet-workshop-isca-conference.

[90] Yoshisato Yokoyama, Miki Tanaka, Koji Tanaka, Masao Morimoto, Makoto Yabuuchi, Yuichiro Ishii, and Shinji Tanaka. A 29.2 mb/mm2 ultra high density sram macro using 7nm finfet technology with dual-edge driven wordline/bitline and write/read-assist circuit. In *2020 IEEE Symposium on VLSI Circuits*, pages 1–2, 2020.

[91] F. Zaruba and L. Benini. The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-ghz 64-bit RISC-V core in 22-nm fdsoi technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(11):2629–2640, Nov 2019. https://github.com/openhwgroup/cva6.

[92] Florian Zaruba, Fabian Schuiki, and Luca Benini. Manticore: A 4096-core RISC-V chiplet architecture for ultraefficient floating-point computing. *IEEE Micro*, 41(2):36–42, 2020.

[93] Mingxing Zhang, Youwei Zhuo, Chao Wang, Mingyu Gao, Yongwei Wu, Kang Chen, Christos Kozyrakis, and Xuehai Qian. Graphp: Reducing communication for pim-based graph processing with efficient data partition. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 544–557. IEEE, 2018.

[94] Youwei Zhuo, Chao Wang, Mingxing Zhang, Rui Wang, Dimin Niu, Yanzhi Wang, and Xuehai Qian. Graphq: Scalable pim-based graph processing. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 712–725, 2019.