ARCHITECTURAL EXPLORATION OF SI-IF MANY-DIE PROCESSORS

BY

DANIEL PETRISKO

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Adviser:

Associate Professor Rakesh Kumar

Abstract

Monolithic, single-die processors dominate today's computing landscape. High performance systems achieve massive throughput by connecting large numbers of discrete chips – CPUs, GPUs, FPGAs – through high latency, low bandwidth interconnects. However, such systems provide limited performance scaling due to high communication costs between the discrete chips. This thesis proposes an alternate path for performance scaling: integrating many dies onto a single chip using a novel assembly technology – Silicon Interconnect Fabric (Si-IF). Many-die processors have both a technical and an economic advantage over their monolithic counterparts. We demonstrate potential benefits of a many-die approach using two approaches: efficient workload coverage design space exploration using many dies and evaluating a many-die wafer-scale GPU design.

To my family, for everything

Acknowledgments

I would like to thank my advisor, Professor Rakesh Kumar, for his guidance, dedication and patience driving me to produce quality research throughout the years. I would also like to thank Professor Puneet Gupta and Saptadeep Pal of UCLA for their collaboration, insights, and suggestions. Additionally, I thank both former and current members of the PASSAT group Henry Duwe, Jose Rodrigo Sanchez Vicarte and Matt Tomei for their academic help, as well as their companionship, good humor and eventually confiscated coffeemaker. Lastly, I would like to thank all my friends and family for their constant emotional support.

Contents

List of Tables

List of Figures

List of Abbreviations

CU	Compute unit, equivalent to a streaming multiprocessor in a GPU
CPI	Clocks per instruction
DSE	Design space exploration
DRAM	Dynamic random-access memory
EDAP	Energy-delay-area product
EDP	Energy-delay product
EPI	Energy per instruction
GPU	Graphics processing unit, an ASIC either discrete from or integrated with a processor
GPM	Single die collection of CUs used in a multi-die GPU
HBM	High bandwidth memory
ILP	Integer linear programming
IPC	Instructions per clock
LLC	Last level cache, the cache which sits nearest to external memory to a processor
MCM	Multi-chip module, a processor which is comprised of several dies in a single package
Si-IF	Silicon interconnect fabric
SRAM	Static random-access memory
TDP	Thermal design power
WSI	Waferscale integration

1. Introduction

Monolithic, single-die processors dominate today's computing landscape. The SoC revolution has allowed designers to integrate multiple hardware subcomponents on the same die to improve performance and energy efficiency, but the benefits are inherently restricted due to die size limits stemming from yield concerns [25] and fabrication costs [17]. Recognizing these trends, commercial processors are beginning to move towards MCM offerings to take advantage of technology node heterogeneity or smaller die yields. Traditional MCMs contain only a few loosely coupled dies, due to the relatively low bandwidth, high latency and high energy cost of inter-die communication. Recent advancements in 2.5D die interconnection technologies [3],[9],[15] motivate system designers to seriously consider more disintegrated chiplet architectures, both to save design costs as well as to pursue an alternative scaling pathway. In a chiplet-based approach, large quantities of small subsets of dies are produced and combined to generate semi-custom chips. Because the dies are small, yields are high and so manufacturing costs are minimized. Using this technique, there is no requirement to use the same technology node for each die, further saving on design and mask costs.

In this thesis, we evaluate potential area savings, energy efficiency and cost benefits of a systematic approach to creating a catalog of multi-die processors. The key insight presented is that traditional DSE [5],[11],[14] has focused on finding a single processor which can perform best across a variety of workloads from a single domain. This methodology does not extend to the problem faced by semiconductor manufacturers today: instead, they must consider which *set* of systems will acceptably cover a *set* of potentially disparate applications domains. Furthermore, if employing chiplet-based assembly, they must consider which chiplets to make available, which can drastically influence the final system selection. We create and evaluate a simple framework for both system and chiplet selection,

using it with a sampling of systems across a variety of application domains to demonstrate its advantages over traditional, naïve approaches.

The second half of this thesis focuses on implications of many-die processors on high-performance processing. Si-IF many-die construction is an enabling technology for waferscale computing, a goal for high-performance architecture since the 1980s [12],[16]. Attempts at waferscale computing ultimately failed at the time because yield problems overwhelmed any available amount of redundancy or error-resilient architecture. By decomposing an unbuildable, monolithic waferscale die into a Si-IF many-die, yield problems disappear. The high availability of intra-die interconnections available through Si-IF enables performance comparable to a theoretical waferscale GPU [20]. We choose a GPU as a demonstration of waferscale computing for several reasons. Data-center GPU workloads demand massive memory bandwidth and computational throughput traditionally spread over many discrete GPUs, so there is a practical application for a high-performance chip which could reduce server footprints. Additionally, unsolved problems in data placement, scheduling and load-balancing make the programming model for multi-GPU computation immensely difficult to optimize [18][19]. WSGPU would maintain a traditional, single GPU programming model. Lastly, GPU applications tend to be latency tolerant, meaning the extra latency from cross-wafer communication in a WSGPU does not significantly impact overall performance. This thesis presents a WSGPU, a waferscale GPU capable of enormous computational throughput while maintaining buildability, energy-efficiency and programmability.

In summary, this thesis makes the following contributions:

- We identify the problem of multi-system design space exploration for system and chiplet selection. We present a solution to this problem: an ILP-based framework for

selecting sets of chiplets and systems to build for optimal coverage of a broad set of applications (Chapter 2).

- We present a waferscale GPU as a sample Si-IF many-die processor, providing performance and energy efficiency infeasible by any other iso-transistor construction of modern GPUs (Chapter 3).

The work presented in this thesis was a series of collaborative efforts between the author and others at UCLA. The author performed all architectural performance and power simulation. The ILP framework was not written by the author, although its input and output data were the responsibility of the author. Data regarding Si-IF specification, fabrication costs, heat-flow analysis and electrical circuit-level details were not provided by the author. Figures 3, 4, and 8 were produced independently of the author and many figures were produced by another using the author's data. Some of Chapter 1 was written as an unpublished paper, and Chapter 2 is currently under submission. Some sections originally written by the author are copied verbatim from these efforts.

2. Designing Multi-Die Systems

2.1. Motivation

2.1.1. Why Multi-Die Systems?

Microarchitectural design space exploration for a general-purpose processor is traditionally aimed at determining the microarchitectural parameter values that maximize processor performance or efficiency over a set of representative applications [5],[11],[14]. However, applications are often targeted not by a single processor, but using a family of processors where each processor in the family targets a subset of the applications.

Unfortunately, as processor design, verification, manufacturing, and management costs increase [17], it will become critical to target applications with the smaller number of systems while achieving the performance and efficiency of larger number of systems. We present the first work on systematically performing a microarchitectural design space exploration when multiple systems will be used to target applications. Our analysis is performed both for traditional processor designs as well as component IP or chiplet-based design.

Our results clearly show benefits of using multiple systems to service diverse workloads (3X improvement in EDA^2P), efficacy of our optimization techniques (1.65X improvement in EDA^2P for same number of systems) and advantages of chiplet assembly approaches over a set of metrics (over 1.5X improvement in EDA^2P).

2.1.2. Is Multi-System Selection Trivial?

Since single processor design space exploration has been studied extensively, one might wonder how effective traditional algorithms are on the multi-system selection problem. To illustrate the need for a smarter selection algorithm, we compare our system selection algorithm (fully explained in Section 2.3) to two naïve approaches. The first approach simply uses traditional design space exploration to

select the best average system, or the system that has the best geometric mean performance across all applications. The second approach iteratively applies this process to greedily select the best 8 systems to cover all applications. The third approach is our ILP-based optimization framework.

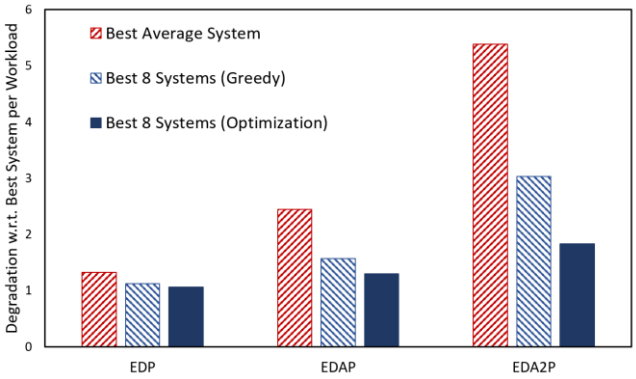

Figure 1. Optimal System Selection Algorithms for Application Coverage

We optimize for three metrics: EDP, EDAP and EDA^{2}P. EDP is a widely used metric to represent the energy efficiency of a system. EDAP and EDA^{2}P additionally correlate with logic designer costs and physical design / mask costs, respectively. Because the systems evaluated in this exercise (enumerated in Table 2) are relatively similar, energy efficiency is increased by only a small amount through optimal selection. However, the advantages of intelligent system selection show when considering EDAP and EDA^{2}P. Our optimization achieves a 21% reduction and 60% reduction, respectively, compared to an iterative greedy approach. It is important to note that a full multi-system design space exploration is impractical, because it is a variant of the NP-complete set cover problem. Therefore, a more efficient approximate selection is necessary.

Figure 1 presents the multi-system selection problem, but to fully realize the benefits of multi-die processors, one must consider chiplet selection in addition to system selection. Design costs are minimized for a multi-die processor when the number of discrete designs is minimized. If unique chiplets are required to construct each system, then the number of chiplet designs can be at worst k times the number of systems, where k is the number of chiplets per processor. Each chiplet design requires logic,

verification and physical design work, in addition to requiring a new mask. While having an unlimited catalog of chiplets will minimize EDP and maximize performance, this approach sacrifices economic advantages of multi-die construction.

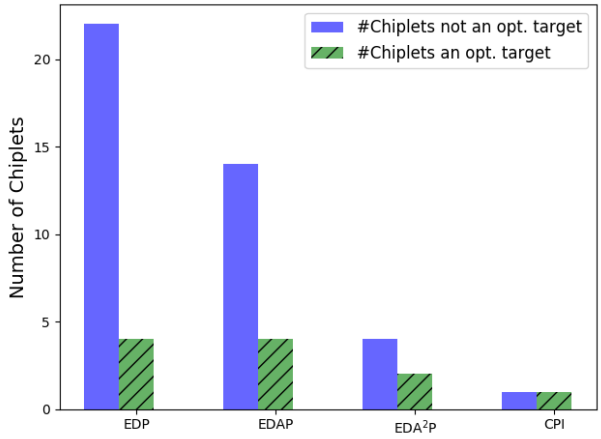

Figure 2. System Selection Optimizing for Number of Chiplets

By adding number of unique chiplets as a minimization objective, we can reduce the total design and manufacturing costs of the system set while only sacrificing minimal efficiency. Figure 2 shows the chiplet reduction when considering number of chiplets as a design metric. The chiplet-focused system selections perform within 5% of chiplet-agnostic system selection, while reducing the number of unique chiplets by 2-5x.

2.2. System Architectures

For this work, we consider that a single-core processor system is comprised of two chiplets: core with L1 cache (core+L1 chiplet) and L2 chiplet. We consider the interconnect characteristics to match those of the state-of-the-art chiplet assembly approach presented in [3],[9],[15]. With latency and energy-per-bit overheads comparable to those of traditional SoCs, core+L1 and L2 chiplet systems require no substantial microarchitectural changes compared to their single die counterparts.

We first perform simulations [24] on all core (includes L1 caches) configurations, with a constant 1MB L2 cache. We fully explore these configurations and then trim the design space to a smaller set using the clustering approach described earlier. We then simulate this reduced set of cores with a wide range of L2 cache sizes to construct a broader design space for our diverse set of workloads. Finally, we use these results to generate power, energy and area results for chiplets implemented in 22 nm, 32 nm and 45 nm technology nodes. Because we assume the same microarchitectural parameters for a chiplet across technology nodes, performance characteristics of each chiplet remain the same. We also consider systems built out of heterogeneous technology chiplets for our evaluations. This is a reasonable assumption as core microarchitectural parameters such as cache latencies are unlikely to vary within generations. In all, we considered a total of 5868 system configurations in our study.

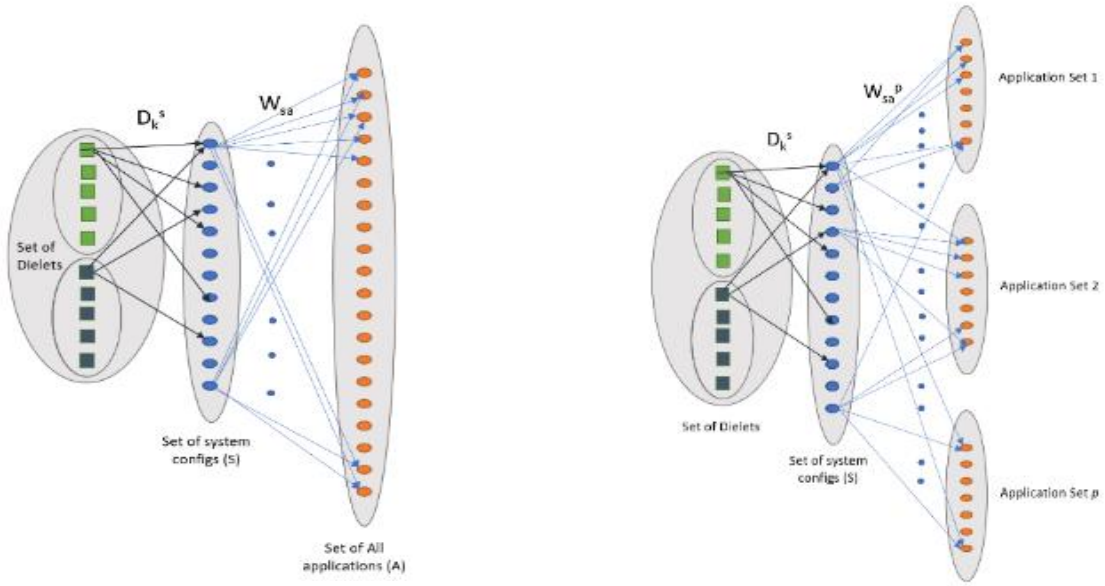

Figure 3. Multi-System Selection Problem Illustrations

We consider two major variants of the multi-system selection problem illustrated in Figure 3. For the first variant, the goal is to determine how to assign each workload from a large set of applications to a set of systems available for construction out of a set of chiplets. A system may cover more than one application. The systems are chosen such that they maximize some objective function

when running their assigned applications, subject to some constraint. For example, one may wish to minimize energy-delay product while constraining total chip area or maximize performance with a constraint on power. The second variant of the problem groups individual applications into application sets where each application-workload may be optimized using a different set of objectives or constraints. A formal description of our ILP-based optimization framework can be found in Section 2.3.

2.3. ILP Framework

By formulating the multi-system selection problem as a set of constraints with a minimization objective, we can solve using traditional ILP solvers. Table 1 and Figure 4 together show the formal specification of our ILP framework. The objective function is minimizing the sum of a normalized metric over all application-system assignments. Each metric is normalized for each application with respect to the best possible system constructible out of the set of chiplets, disregarding any additional constraints. Constraint (a) assigns one system to every application. Constraint (b) enforces a hard constraint on systems relative to other systems in the solution; e.g. performance may be no worse than 2x the performance of the best system. Constraint (c) enforces a hard constraint on systems, e.g. total area may be no more than 10 mm^2. Constraint (d) ensures that system-application mappings are present in the selected system set. Constraint (e) guarantees only systems selected to service applications are present in the selected systems set. Constraint (f) bounds the total number of systems selected. Constraints (g, h) trim the chiplet set to contain only chiplets used to construct chosen systems. Constraint (i) bounds the total number of chiplets selected.

Table 1. ILP Framework Definitions

Notation	Meaning
S	Initial set of all systems under consideration
A	Set of Applications
D	Set of all chiplets under consideration
Va	Estimated relative volume of systems running application *a*
x_s^a	0 or 1 indicating whether *s* is used to service application a
X_s	0 or 1 indicating whether system *s* is present in final system set
W_s^a	n-tuple of metrics (EDP, CPI, etc.) when application *a* runs on system *s*
N_p	p^{th} element of W_s^a, p ϵ [1, *n*]
c_s	Area of system *s*
α_a	Area constraint for application *a*
χ_{ap}	Threshold value of parameter *p* for application *a*
D_d^s	0 or 1 indicating whether chiplet *d* is a component of system *s*
y_d	0 or 1 indicating where chiplet *d* is present in the final set of chiplets
K	Maximum number of unique systems possible in the final set of systems
L	Maximum number of unique chiplets allowed in the final set of chiplets

Minimize:

$$\sum_{a\in A}\sum_{s\in S} x_s^a * W_s^a.N_j$$

Subject to:

(a) $$\sum_{s\in S} x_s^a = 1, \qquad a \in A$$

(b) $$\sum_{s\in S} x_s^a * W_s^a.N_p \le \chi_{ap}, \quad p \in [1,n]; a \in A$$

(c) $$\sum_{s\in S} x_s^a * c_s \le \alpha_a, \qquad a \in A$$

(d) $$x_s^a \le x_s, \qquad s \in S; a \in A$$

(e) $$\sum_{a\in A} x_s^a \ge x_s, \qquad s \in S$$

(f) $$\sum_{s\in S} x_s \le K$$

(g) $$y_d^t \ge D_d^s * x_s \qquad s \in S; d \in D; t \in$$

(h) $$\sum_{s\in S} D_d^s * x_s \ge y_d^t \qquad d \in D; t \in T$$

(i) $$\sum_{d\in D, t\in T} y_d^t \le L$$

Figure 4. ILP Framework Formulation

2.4. Evaluations

2.4.1. Methodology

The methodology for our design space exploration is an iterative process consisting of three main steps, illustrated in Figure 5. First, we identify a set of interesting initial system configurations to explore. This set of configurations (Table 2) must be diverse enough so that changing any parameter has a measurable impact on either power, area or performance of the system running some application. In addition, these configurations should vary enough such that their range tightly covers the range of application behavior.

Table 2. Multi-System DSE System Configurations

Issue-width	1, 2, 4	**ROB (OOO)**	32, 64 entries
I-Cache	8 kB DM, 16 kB 2-way, 32 kB 4-way, 64 kB 4-way	**L2 Cache**	256 kB 1-way, 512 kB 2-way, 1 MB 4-way, 2 MB 8-way, all 12 cycle access
Execution Units	3, 6	**ITLB-DTLB**	64-28 entries
IQ (OOO)	48, 96	**Ld/St Queue**	32 entries

For example, cache size configurations should be selected such that having any smaller cache (than the smallest configuration) will sacrifice performance while failing to reduce power or area by a significant amount; similarly, having a larger cache (than the largest configuration) will not significantly increase performance but would incur a power or area penalty. Optimally selecting these initial configurations is a problem which requires either relying on architectural intuition or performing extensive pre-processing.

Next, we perform a full factorial exploration of these initial system configurations. To evaluate performance of designs, we use Sniper [6], a fast trace-driven multi-core interval simulator. We use the ROB-centric model in Sniper, which more accurately models in-order cores and issue contention.

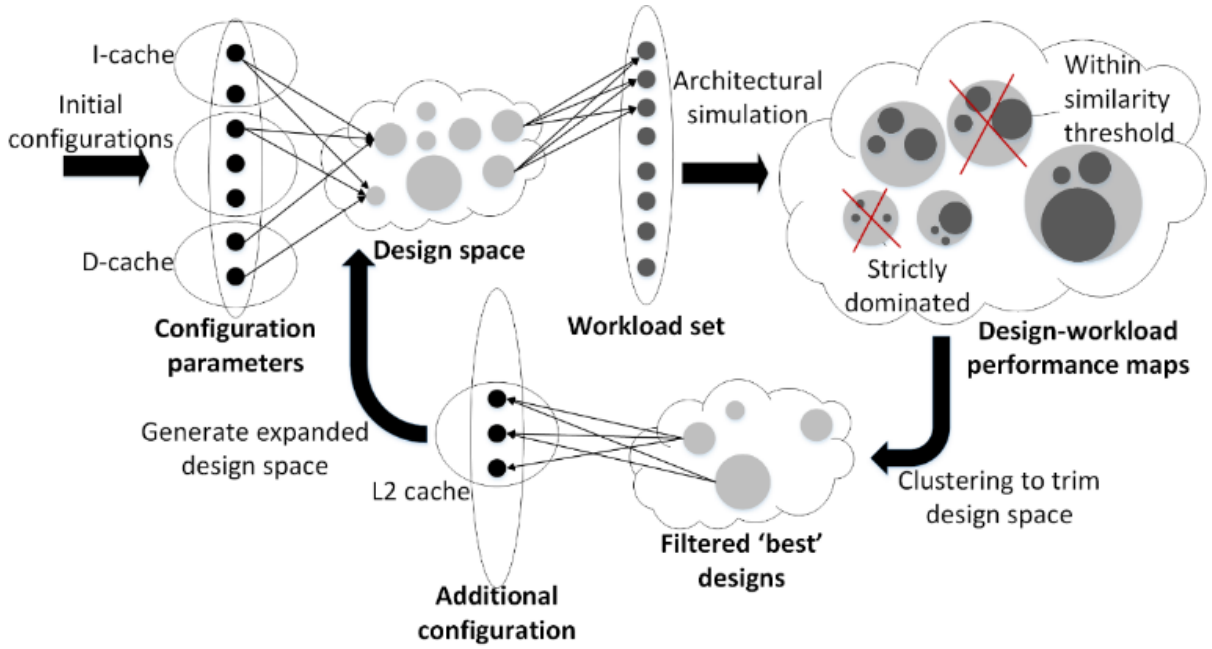

Figure 5. Multi-System DSE Methodology Cartoon

For area and power modelling, we model 22 nm, 32 nm and 45 nm processes using McPat 1.0 [13], which is deeply integrated with Sniper. We consider area as the total area of the processor, not only core and cache. Peak power in McPat is calculated based on estimation of maximum activity factors for the processor and so may vary between workloads. As a conservative estimate, we calculated the highest peak power estimated for any workload as the peak power for a given system.

To calculate average power and energy for our primary results, we exclude DRAM energy since DRAM can be easily swapped out during system assembly and selected to match the power or performance requirements of a given application domain. We performed a sensitivity analysis of our results when including DRAM and found that the trends and analysis were largely unchanged.

The design space of this initial set of systems is pruned using a canopy clustering technique such that systems which are worse (or within 5%) with respect to the rest of the systems for every metric of interest on all workloads are removed from consideration. We then add new configurations to our reduced design space. This clustering usually cuts down the design space by 3X-10X with negligible

impact on eventual optimized metrics. This is important as DSE runtime is dominated by microarchitectural simulation.

Lastly, we use the optimization algorithm described in Section 2.3 to select the best chiplets and systems to cover the given workloads.

Table 3. Multi-System DSE Benchmarks

Suite	Benchmarks
EEMBC	FFT, autocorr, convEn, binSearch, div, inSort, intAVG, intFilt, mult, rle, tea8
SPEC2006	perlbench, omnetpp, mcf, libquantum, astar, xalancbmk, leslie3d, calculix, GemsFDTD, cactusADM, dealII, soplex, lbm, povray
SPLASH-2	cholesky, fft, radix, raytrace, ocean.cont, ocean.ncont
NPB	BT, EP, MG, SP

A diverse set of workloads were chosen by subsetting 4 benchmark suites - SPEC2006 [8], EEMBC [22], SPLASH-2 [28], and NAS Parallel Benchmark (NPB) [2]. We chose 14 applications from SPEC2006 to demonstrate the diversity of the suite without capturing redundant results between benchmarks. SPEC2006 applications were simulated using 100M-warmup, 30M-detail Pinballs [21] with maxK=10 (up to ten SimPoint regions are simulated in total). Four benchmarks selected from NPB were simulated using 100M-warmup, 30M-detail Pinballs with maxK=10 and input size W. Twelve benchmarks from the telecomm and automotive suites of EEMBC were simulated by running them in a loop in detailed mode for 30M instructions. SPLASH-2 benchmarks were run single threaded with fast-forwarding until the ROI (region of interest) was simulated in full. Table 3 lists all benchmarks considered.

2.4.2. Results

In Section 2.1 we demonstrated the value of intelligent system selection, as well as the benefit of having many systems to cover applications. We then showed the necessity of explicitly optimizing to reduce the number of unique chiplets. While we have chosen a realistic set of architectural parameters,

benchmark applications and optimization goals for our analysis, different parameter configurations for our general-purpose ILP framework will select drastically different systems. Therefore, we present the results in this section not as proof of potential benefits of system selection, but instead to provide insight as to general trends which we observed from a large number of parameter sweeps.

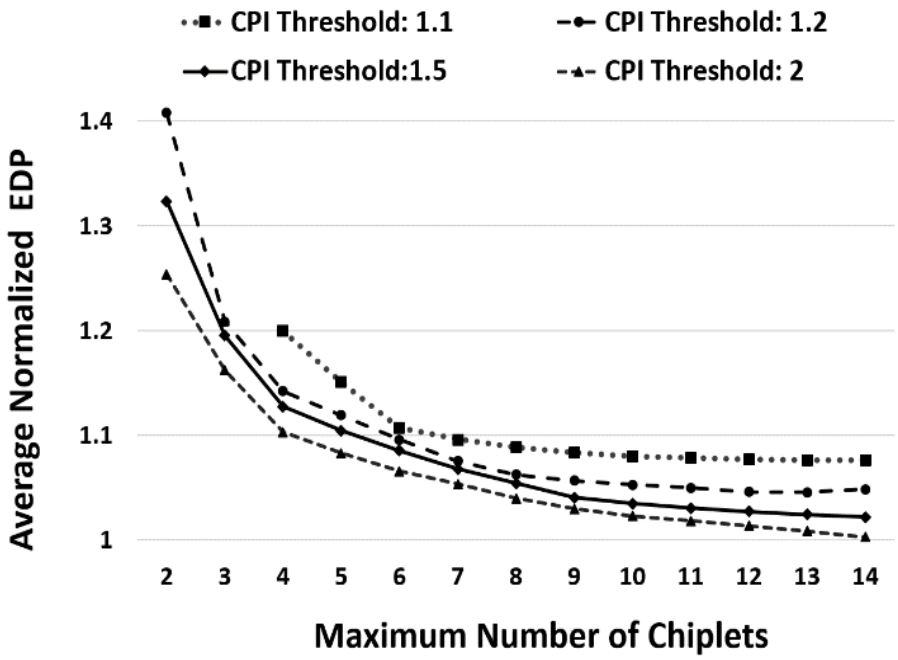

Figure 6. Number of Chiplets Needed for EDP Minimization

Figure 6 shows an EDP minimization selection, where a threshold CPI eliminates systems which perform a threshold number of times worse than the best possible system configuration. As we loosen the CPI constraint, the minimum achievable EDP is lowered. Interestingly, tighter CPI constraints lead to a lower saturation point for number of chiplets achieving noticeable EDP benefits. This is intuitive when considering the degenerate case: a threshold CPI of 1 will saturate at 2 chiplets selected because all other chiplets have been removed from viability. When deprioritizing performance, more unique combinations of chiplets can be selected to provide benefits. However, these benefits become small in single digit number of chiplets. This is because unique chiplets are able to cover large swathes of the

design space: k chiplets can be used to construct 2^k unique systems. This is a massive advantage of multi-die processor construction.

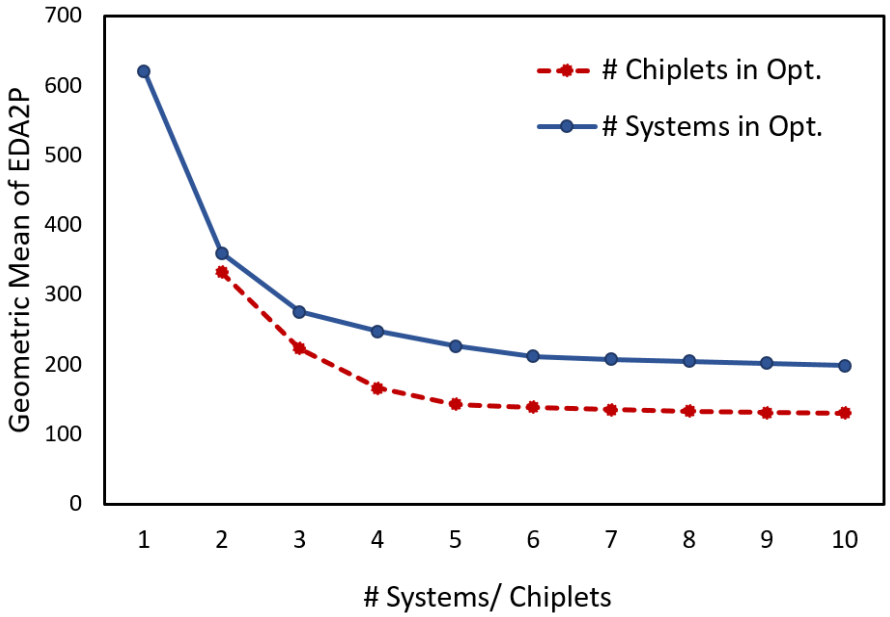

Figure 7. Systems / Chiplets Needed for EDA^{2}P Minimization

Figure 7 shows an EDA^{2}P minimization when constraining the number of unique chiplets and systems. As number of unique designs approximates manufacturing costs and EDA^{2}P approximates design cost and energy efficiency, Figure 7 demonstrates both the economical and performance benefits made possible through multi-die processors. At 10 chiplets (2^{10} possible systems), a multi-die processor achieves 1.5x improvement in EDA^{2}P. Additionally, each chiplet is smaller in area and lower in complexity than a full system, resulting in both lower design and manufacturing costs. As unique ASIC demand increases, while ASIC design and fabrication costs skyrocket, multi-die processors offer a way to provide semi-custom solutions through design reuse.

3. A Practical Waferscale GPU

3.1. Motivation

Increasing communication overheads are already threatening computer system scaling. Waferscale processors [12],[16] were considered in the 1980s as way to dramatically reduce communication overheads. However, the concept was abandoned due to yield issues inherent to conventional integration technology. We argue that a Silicon-Interconnection Fabric (Si-IF)-based integration [3],[9],[15] where pre-manufactured dies are directly bonded on to a silicon wafer, enables one to build a waferscale system without the corresponding yield issues. As such, waferscale processing needs to be revisited. We show that a waferscale GPU can provide significant performance and energy efficiency advantages over traditional multi-GPU systems (up to 3.4x and 4x average speedup with 24 and 40 GPU-equivalent waferscale GPU, as well as 8.8x and 16.9x average EDP benefits) without any change in the programming model or the scheduler. While thermal and power delivery constraints significantly impact the waferscale GPU architecture, techniques such as voltage-stacking and voltage scaling can enable high-performing waferscale GPU systems. A proof-of-concept Si-IF prototype with interconnected dies is also presented that suggests feasibility of the proposed waferscale processors.

3.2. System Architecture

Table 4. Many-die GPU System Configurations

	Scale-Out SCM-GPU	Scale-Out MCM-GPU	Waferscale GPU
CUs per GPM	64	64	64
L2 Cache per GPM	4 MB	4 MB	4 MB
DRAM (HBM)	1.5 TB/s, 100 ns, 6 pJ/bit	1.5 TB/s, 100 ns, 6 pJ/bit	1.5 TB/s, 100 ns, 6 pJ/bit
GPMs per package	1	4 (Ringbus)	All (Mesh)
GPM Interconnect	None	1.5 TB/s, 56 ns, 0.54 pJ/bit	1.5 TB/s, 20 ns, 1 pJ/bit
Package Topology	Mesh	Mesh	Single Package
Package Interconnect	256 GB/s, 96 ns, 10 pJ/bit	256 GB/s, 96 ns, 10 pJ/bit	None

We consider three constructions of multi-chip, large-scale GPU systems, described in Table 4. For each of these constructions, we consider the atomic hardware unit to be a GPM, roughly equivalent to the largest feasible GPU available today, combined with an HBM DRAM die. Each GPM has a TDP of 200 W and an area of 500 mm^2 for the GPU die, plus 70 W and 200 mm^2 for two 3D-stacked DRAM dies. Scale-Out SCM-GPU is a conventional large-scale GPU system, where each GPM is contained in its own package, connected to each other via an interconnect such as NVLink. Scale-Out MCM-GPU is a conventional extension of MCM-GPU [1], which is a multi-chip module consisting of four GPMs in a single package. Scale-Out MCM-GPU connects a set of MCM-GPUs in the same manner as Scale-Out SCM-GPU. Last, WSI is our proposed waferscale architecture, comprising a set of GPMs connected using Si-IF technology in a single package. In all architectures, GPMs are laid out in a 2D mesh topology.

The authors of MCM-GPU proposed several optimizations to reduce inter-GPM communication. One optimization was to use a First Touch paging policy, where data pages are placed into the GPMs which first access them. Another was to schedule consecutive thread blocks of a single kernel to the same GPM, rather than using a pure round-robin global scheduler. We take both optimizations as our baseline for Scale-Out MCM-GPU and WSI.

Figure 8 shows the link bandwidth, energy per bit and latency characteristics for each integration tier. There is roughly an order of magnitude loss for each characteristic when moving between integration tiers. This drastic decline in efficiency prevents conventional SCM and MCM architectures from achieving waferscale. As Si-IF provides interconnect performance between conventional on-chip and inter-die capabilities, Si-IF is an enabling technology for waferscale systems. However, Si-IF connections are still less performant than on-chip connections. An ideal WSGPU would exist not only in a single package, but on a single die. However, it has long been conventional wisdom

that yield constraints prevent manufacturing single die waferscale systems. As we explore in Section 3.4.4, architectural optimizations are needed to mitigate the performance impact of these imperfect inter-die links.

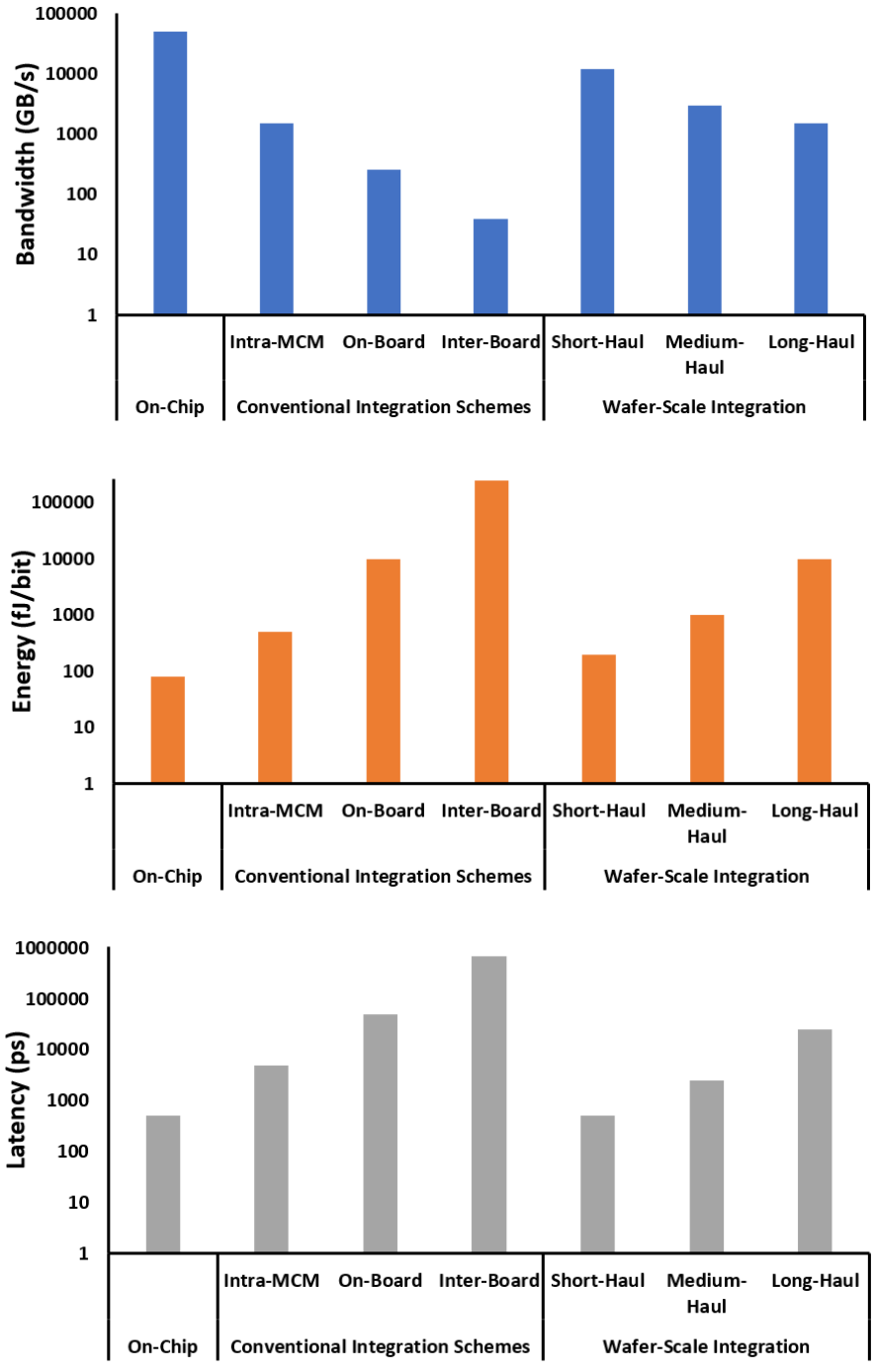

Figure 8. Interconnect Link Integration Tier Comparisons

3.3. Comparison to Traditional Architectures

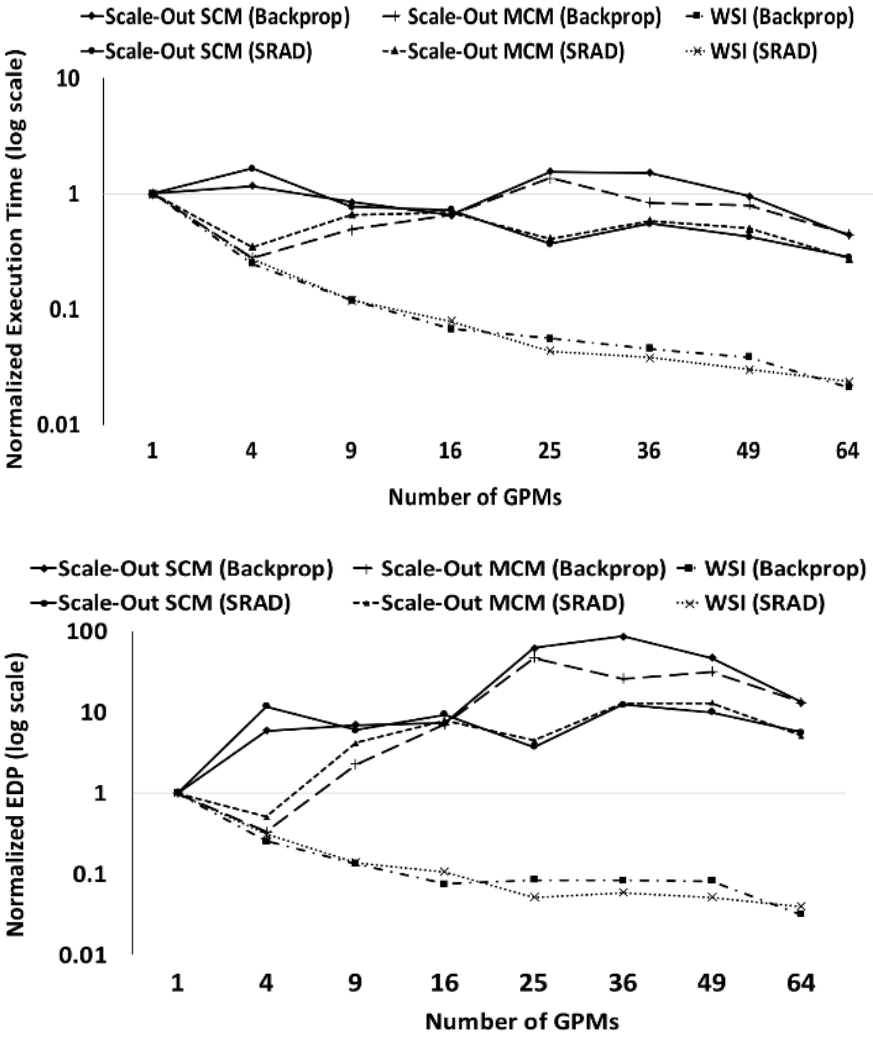

Figure 9. Waferscale GPU Scaling Motivation

Figure 9 demonstrates the limited scalability of Scale-Out SCM and Scale-Out MCM systems compared to WSI. For Backprop, we observe a 47.54x speedup for 64 GPM WSI over a single GPM system, and a 20.8x and 21.13x speedup over the highest performing Scale-Out SCM-GPU and Scale-Out MCM-GPU systems, respectively. For SRAD, we observe a 42.56x speedup for 64 GPM waferscale GPU over a single GPM system. This is compared to Scale-Out SCM-GPU and Scale-Out MCM-GPU which saturate at 3.57x and 3.65x speedup, respectively. Speedup in the Scale-Out systems is eventually limited by memory transfer latency. An additional benefit of WSI systems is that these speedups are

attainable without requiring changes to the programming model, unlike Scale-Out systems which comprise logically separate GPUs.

While all three architectures achieve performance benefits from having large numbers of GPMs, only WSI architectures scale efficiently. Scale-Out systems with many GPMs have drastically higher EDPs than either WSI systems or even single GPM systems. 64 GPM WSI has a 31.54x reduction in EDP compared to a single GPM, trending downwards, whereas both Scale-Out systems increase in EDP past 9 GPMs. For SRAD, we observe the 64 GPM waferscale GPU manages a 24.88x reduction in EDP, a sharp contrast to the order of magnitude EDP increase for Scale-Out MCM-GPU and Scale-Out SCM-GPU systems.

3.4. Evaluations

3.4.1. Benchmark Selections

Table 5. Waferscale GPU Benchmarks

Benchmark	Suite	Domain
backprop	Rodinia	Machine Learning
hotspot	Rodinia	Physics Simulation
lud	Rodinia	Linear Algebra
particlefilter-naive	Rodinia	Medical Imaging
srad	Rodinia	Medical Imaging

We evaluate our systems with five benchmarks from the Rodinia benchmark suite [7], shown in Table 5. These workloads were selected as highly parallel GPU applications with scalable input sizes which we were able to run on gem5-gpu to gather complete memory traces.

3.4.2. Methodology

Figure 10 shows our simulation methodology.

The first step in modelling a waferscale GPU is to collect memory traces from which to extract the application behavior. We create an 8 compute unit (CU) GPU in gem5-gpu Syscall-Emulation (SE)

mode, placing a memory probe on the Load-Store Queue (LSQ) of each CU. Next, we run each benchmark in fast-forward until the beginning of the ROI (region of interest), where we take a checkpoint. Finally, we run the entire ROI of the application in detailed mode, collecting a memory trace of every global read, write, and atomic operation. The files are fed into our trace-based simulator.

The trace-based simulator first parses the traces, gathering the relative timing, virtual address, type of operation (read/write/atomic) and size of the memory access, maintaining the block id of the access, but clearing any affinity to a particular compute unit. Private compute time is estimated as the time spent between non-consecutive memory accesses multiplied by the duty cycle of the compute unit which originally executed the request. Note that this private compute time is not simply raw computation, but also includes accessing block shared memory. In the view of the simulator, there is no difference between the two operations. These compute requests are grouped along with global memory accesses into thread blocks. When executing a thread block, compute requests must conservatively wait until all outstanding memory requests have completed. Conversely, new memory requests must wait until there are no outstanding compute requests to proceed. This assumption is based on the in-order execution of warps within a thread block. In a real GPU system, the local warp scheduler will overlap computation and memory accesses by switching out warps upon cache misses.

Every topology we consider in detail in this work is some version of a 2D mesh. To minimize inter-GPM traffic, when simulating an arbitrary number of GPMs, effort is made so that the resulting mesh is as close to square as possible. If the number of GPMs which can be enabled (e.g. for power reasons) is less than the dimension of the closest desirable grid, the excess GPMs are simply disabled. For instance, when simulating 40 GPMs, we choose a 7x6 mesh rather than 10x4, forbidding scheduling to the last two GPMs. The mesh itself is modelled as bidirectional links rather than as a NOC with virtual channels: besides being a much simpler (and therefore faster) implementation, it is also pessimistic

interconnection behavior. Therefore, we expect less contention in an actual waferscale GPU compared to our model.

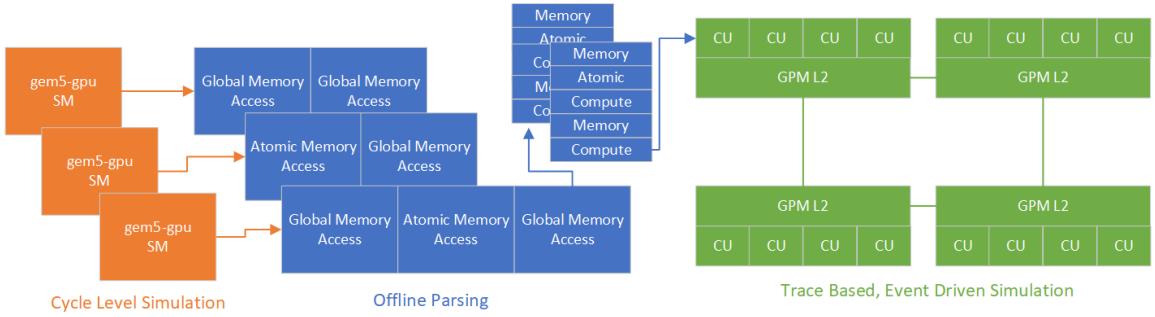

Figure 10. Trace-Based Waferscale Simulator Cartoon

3.4.3. Simulator Validation

We validated our trace-based simulator against gem5-gpu [23] for a small number of compute units. Figure 11 compares normalized performance across the two simulators for different number of CUs as well as normalized performance across the two simulators for different DRAM bandwidth values.

We observe a geometric mean of 5% and maximum error of 28% across the two simulators when number of CUs is scaled. We observe a geometric mean of 7% and maximum error of 26% across the two simulators when DRAM bandwidth is scaled. The results suggest the validity of our simulation approach (relative to gem5-gpu).

As an additional validation step, we created roofline plots [27] as visual representations of the interplay of bandwidth, data locality and computation resources in the two simulators. Visual inspection of Figure 12, roofline plots of an 8 CU system, reveals the same general characteristics and application positioning between gem5-gpu and our trace-based simulator. This correspondence builds further confidence that application characteristics, such as compute to memory ratio, data locality and bandwidth bottlenecks are preserved in our trace-based methodology.

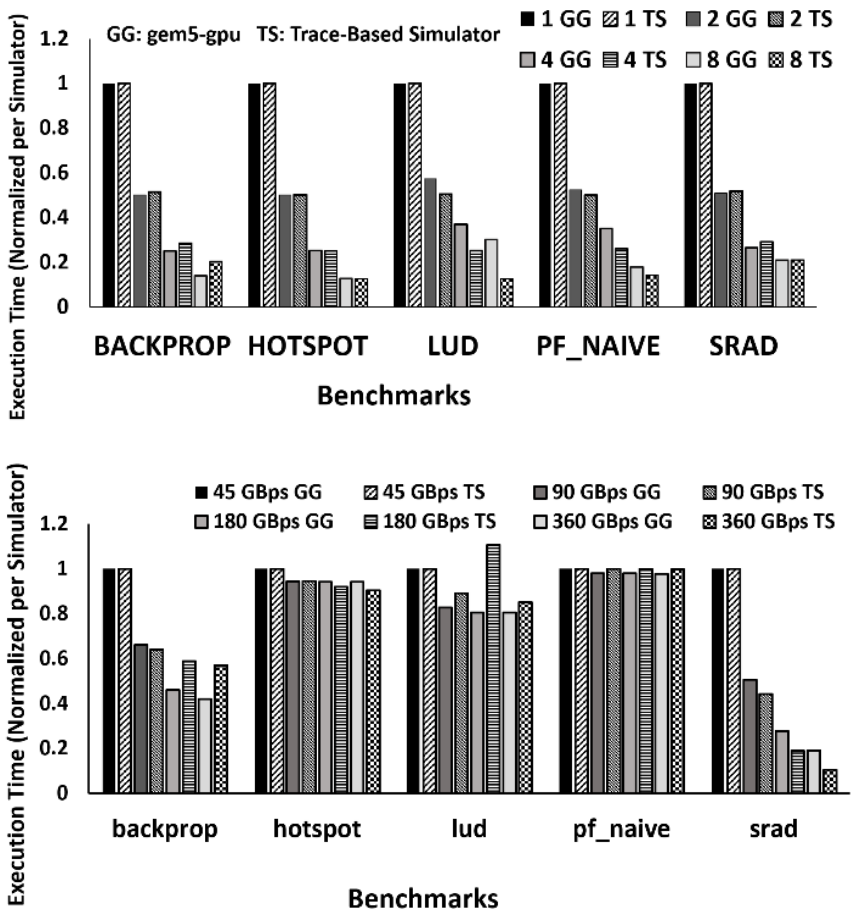

Figure 11. Trace-Based Simulator Scaling Validation

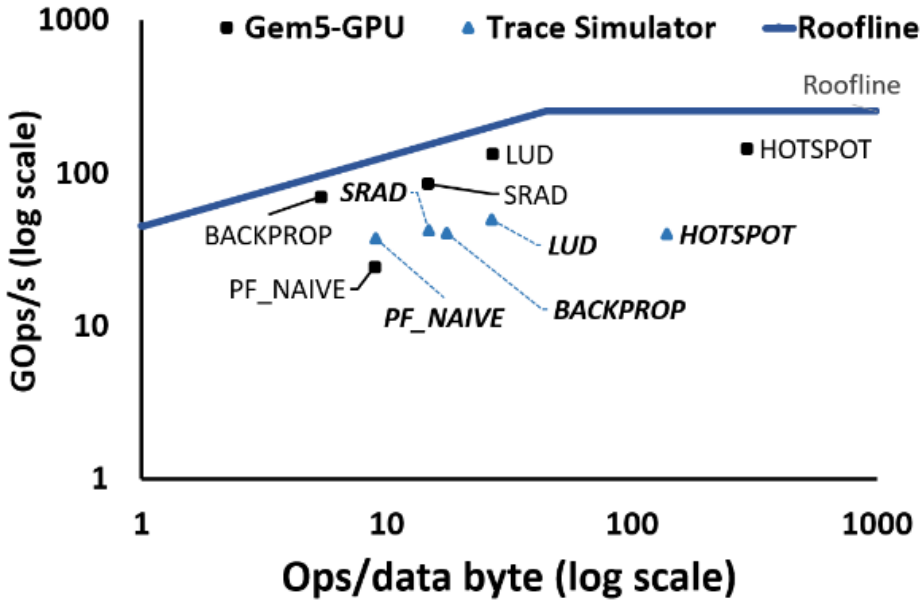

Figure 12. Trace-Based Simulator Roofline Validation

3.4.4. Scheduling Results

Performance and especially EDP of a waferscale GPU architecture depend heavily upon how computation and dataflow are distributed. In a conventional GPU, thread blocks are distributed round-robin to CUs by a centralized controller. Prior work [1],[18][19] has explored scheduling and data placement in this context. However, relatively large trans-GPU inter-GPM communication costs increase the importance of these policies. Additionally, while prior work has traditionally focused on computational load balancing and maximizing utilization of memory bandwidth, the massive memory bandwidth afforded by Si-IF and the significant energy costs of trans-WSI communication shift the focus toward enhancing data locality and minimizing data transfers between GPMs. Most GPU applications are relatively latency tolerant, so the extra latency incurred from multi-hop inter-GPM transfers tends to have limited impact on performance. However, as we show in this section, there are significant energy benefits to minimizing inter-GPM communication.

As discussed in Section 3.2, our baseline scheduler is locality-aware round-robin, where sequential thread blocks are assigned to the same GPM. This baseline scheduler is paired with the first-touch paging policy.

There are two competing forces on an ideal scheduler. To minimize I/O energy, the schedule must attempt to eliminate inter-GPM communication. However, runtime energy – and therefore execution time – must also be considered. After all, a schedule which assigns every thread block to a single GPM would minimize I/O energy at the expense of total energy. Therefore, we explore the benefits of two heuristic based schedulers, paired with a simple runtime load balancer.

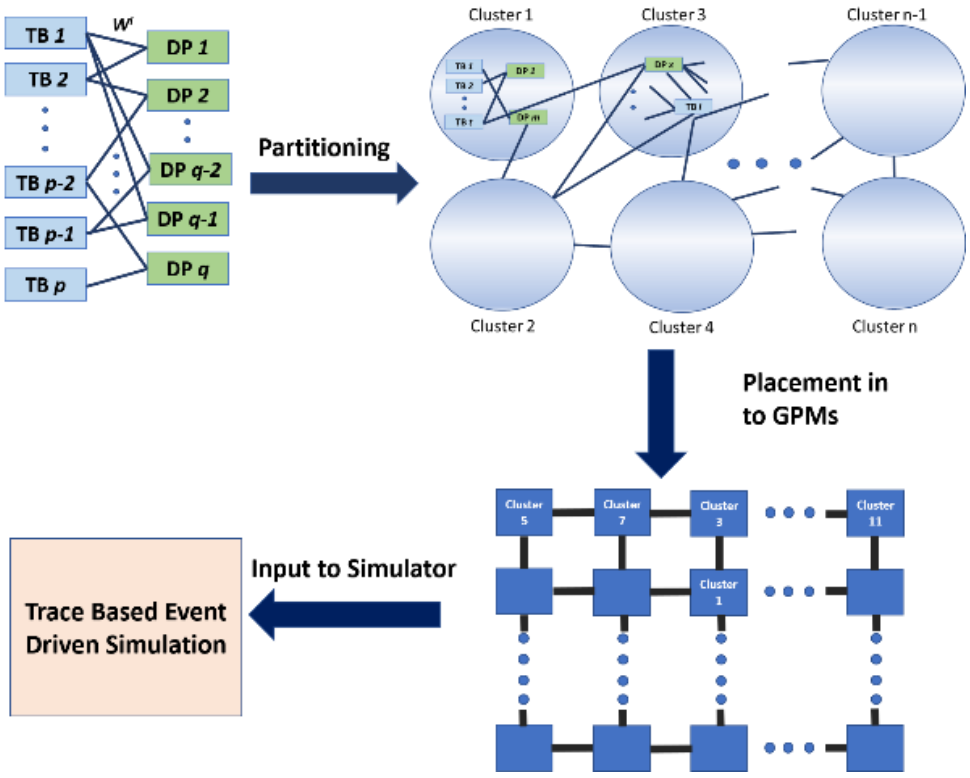

Figure 13. Waferscale GPU MinComm Scheduling Methodology

The MinComm scheduler is an offline partitioning and placement framework focused on assigning thread blocks to GPMs, as well as placing DRAM pages in optimal GPMs to minimize overall remote memory accesses. First, a memory trace is gathered from a previous run of the application. This is the same trace which is used as input into our trace-based simulator. From the trace, we construct a bipartite graph between thread blocks and DRAM pages, with edge weights corresponding to the number of times a DRAM page is accessed from a thread block. A Fiduccia-Matthessey (FM) partitioning algorithm [4] is used to cluster thread blocks into groups to assign to a single GPM. Simultaneously, clustering provides a DRAM page placement map using the opposite nodes in the bipartite graph.

While this approach minimizes the number of remote memory accesses, the average distance of remote memory accesses is suboptimal. The next step in generating the MinComm schedule is to assign each logical GPM solved for in the clustering step to a physical GPM in the 2D mesh. We use simulated annealing to map logical GPMs to appropriate physical GPMs. This step minimizes the top number of

hops, minimizing the total access latency and energy of remote memory accesses. Note that by placing highly communicative GPMs close together, the possibility of inter-GPM congestion increases. However, with the generous bandwidth provided by Si-IF, we did not notice any performance degradation due to this effect. Figure 13 illustrates the overall scheduling process.

This offline scheduling framework has the advantage of being simple and, as we show in Figure 14, achieves very close to optimal placement. Unfortunately, it requires profiling the application and would not perform as well for applications with significant input variance, such as large graph applications or applications with unusual or unbalanced memory access patterns. We leave a full study of dynamical scheduling policies to future work.

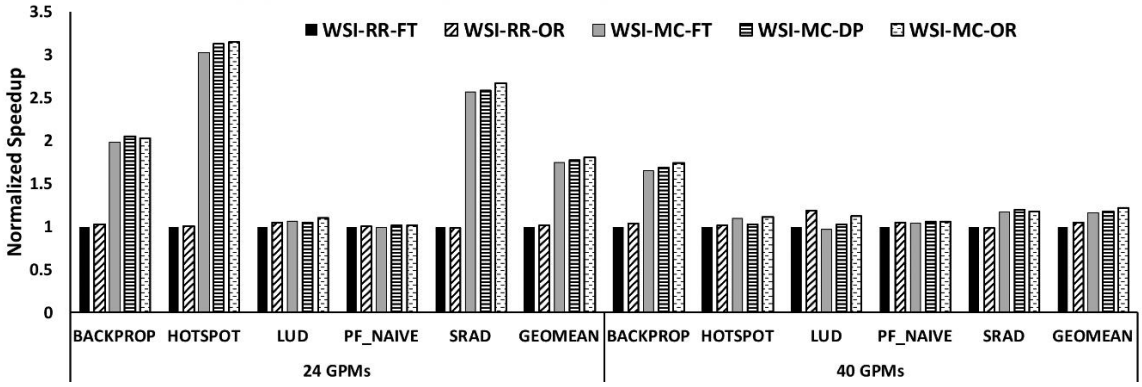

Figure 14. Waferscale GPU Scheduling Speedup Results

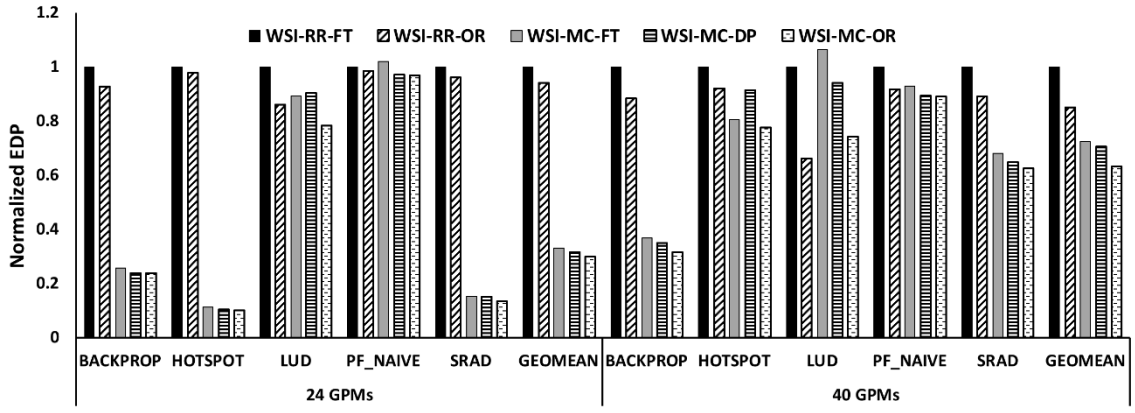

Figure 15. Waferscale GPU Scheduling EDP Results

Figure 14 shows the relative performance of various scheduling policies for 24 GPM and 40 GPM WSI systems. We compare against an oracular data placement policy which allocates all DRAM pages simultaneously into all GPMs. We observe that our baseline policy performs within 13% of the round-robin scheduler with oracular data placement, consistent with the observation made in [1]. Interestingly, we see far greater benefits from MinComm: for applications such as backprop, hotspot and srad, up to 3.13x speedups are achievable. Although communication is being minimized in both oracular round-robin and MinComm first-touch, MinComm preserves locality more strongly, making caching more effective. Overall, our offline policy with data placement results in 1.79x (24 GPMs) and 1.22x (40 GPMs) benefits over the baseline round-robin first-touch policy and comes within 4% of MinComm with oracular data placement. This results in an overall EDP benefit of 66% and 18% for 24 GPM and 40 GPM WSI systems, respectively, as shown in Figure 15.

3.4.5. Performance and EDP Results

Our motivation in Section 3.3 showed that energy efficient scaling is impractical for Scale-Out SCM and MCM systems. A natural question is whether the optimizations suggested in Section 3.4.4 would equally apply to Scale-Out systems. Figure 16 compares Scale-Out MCM systems to WSI systems with the best MinComm data placement policies. We find that waferscale GPUs have overwhelming advantages over Scale-Out MCM systems. WSI performs 14.7x and 21x better than a single MCM-GPU, as well as 10.7x (3.4x average) and 10x (4x average) better than equivalent GPM Scale-Out systems. Similarly, average EDP benefits against equivalent GPM counterparts are 8.8x and 16.9x, respectively. Lastly, a major motivation for WSI adoption is that the programming model need not change to achieve these benefits, whereas Scale-Out systems would need to be explicitly programmed as many-GPU systems, a well-known difficult problem.

While waferscale GPUs present considerable performance and energy efficiency advantages, the architecture here is by no means optimal. Optimizations such as network-on-chip design, DRAM sizing, thermal-aware floor-planning, and GPM heterogeneity could provide even greater benefits. We leave that design space exploration to future work.

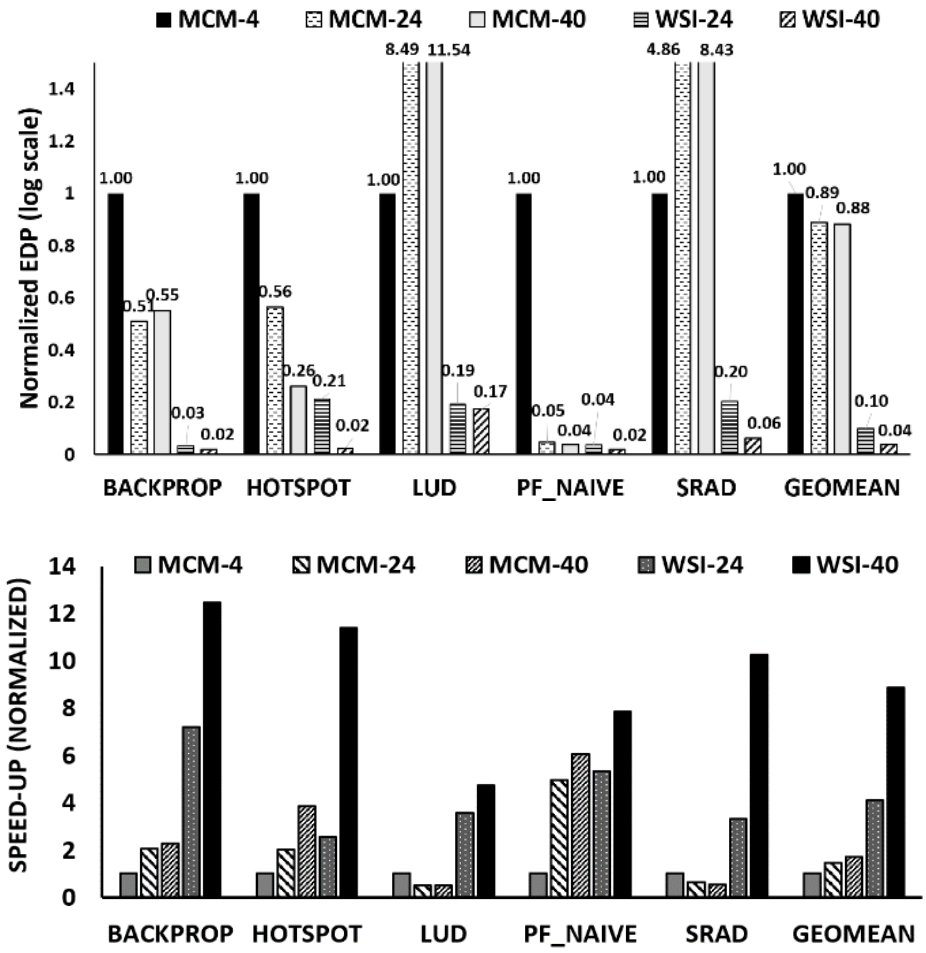

Figure 16. Performance and EDP Advantages of Waferscale GPU

4. Related Work

4.1. System-On-Chip Architectures

Due to a variety of economic and technical constraints, SoC designs dominate the modern computing landscape, from mobile devices to high performance server chips [29]. The ability to integrate hundreds of IPs from dozens of disparate companies enables previously unmanageably complex designs to be completed in years rather than decades. SoCs are designed by combining both soft and hard IP blocks along with custom logic and manufacturing the entire design on a single die.

While SoCs also enable rapid design customization and low-cost reuse of IP, Si-IF many-dies differ by enabling designers to reuse already manufactured silicon dies. Even in the case of an SoC reusing a hard IP, a chip manufacturer will have to design a mask to manufacture it. Compare to Si-IF many-dies, where each chiplet is manufactured independently and masks can be reused for each chiplet shared between chip designs.

4.2. Multi-Chip Modules

The desire for technology node heterogeneity and design reuse motivate today's multi-chip modules. New interconnection technologies such as Intel EMIB [15] connect several dies in a single package. Recent processors have taken advantage of MCM construction to package cutting-edge cores with lagging technology node eDRAM, for example. One could consider Si-IF many-dies a natural progression of modern MCM technology.

4.3. Brick and Mortar Silicon

Systems produced by the chip fabrication technique Brick and Mortar [10] are the closest related work to the many-die system design space exploration described in Chapter 2. Brick and Mortar assembly is a structured architecture consisting of manufactured silicon chips ("bricks") and an I/O cap

("mortar"), which enables designers to create made-to-order chip designs. Available bricks consist of loosely coupled pieces of silicon: processor cores, memories and accelerators. Available mortars are a high-bandwidth packet-switching network and a low-bandwidth island-style mesh. By selecting a few bricks from a catalog of mass-produced silicon and integrating them after testing, Brick and Mortar assembly promises to drastically reduce custom ASIC manufacturing costs.

There are three main differences between many-die processors using Si-IF and a Brick and Mortar assembly approach. Significantly higher bandwidth, lower latency and lower energy per bit communication enable Si-IF many-dies to consist of much more tightly coupled dies. Brick and Mortar designs only consider cores or accelerators as their finest granularity, whereas Si-IF technology allows us to separate out L1 or L2 cache dies or heavily communicating cores. Another difference is the degree of architectural flexibility. Brick and Mortar assemblies have a rigid interconnect structure and only offer a few sizes of bricks. This rigidity helps reduce manufacturing costs, assists designers with mixed process integration and design flow reuse, and further improves yield. Lastly, the authors of Brick and Mortar only discuss cost reduction – there is no mention of using Brick and Mortar to enable higher performance systems. Brick and Mortar is likely not a highly scalable technology as presented because the restricted bandwidth and loose coupling of bricks prevent large-scale integration.

4.4. Exascale APU

When considering exascale computing, there are few ways forward that do not rely on heavily integrated MCMs. In [26], the authors describe an *Exascale Heterogenous Processor (EHP)*, which uses die-stacking and chiplet technologies to integrate CPU + GPU + in-package 3D memory. The authors describe this packaging approach as one of several optimizations which will allow achievable exascale computing. They observe that out-of-chiplet traffic represents 60-95% of traffic across representative kernels, while only amounting to a 13% performance impact compared to an impractical iso-transistor

monolithic chip. They conclude that chiplet organization favorably trades longer network latency for improvement in die yield and cost. While this work reaches a similar conclusion as Si-IF many-dies, it considers chiplet organization as one of many potential optimizations for exascale computing. It does not delve into chiplet selection or chiplet interconnection architecture. Lastly, this work only considers exascale applications when evaluating power, performance and cost trade-offs. Our work focuses on applications from embedded to high-end server domains.

4.5. MCM-GPU

The authors of MCM-GPU [1] observe that with the end of Moore's law, traditional transistor-based GPU scaling will become infeasible. As an alternative scaling pathway, they propose MCM-GPU, which is a multi-die GPU composed of 4 GPMs connected via 2d mesh in a single package. Each GPM is a manufacturable GPU die with 64 CUs. The authors compare MCM-GPU to both a theoretically maximal monolithic GPU with 128 CUs and an immediately available multi-GPU setup. Due to the order of magnitude differences between chip, package, board and system communication bandwidth and energy per bit costs, MCM-GPU offer significantly improved power and performance over an optimized multi-GPU system, while offering comparable performance to an unbuildable monolithic GPU.

MCM-GPU is the baseline to which we compare WSGPU. MCM-GPU provides a scaling pathway beyond single die GPUs, while WSGPU scales to an entire wafer worth of GPMs. We compare WSGPU against a system composed of MCM-GPUs connected in a 2D mesh, which we call Scale-Out MCM-GPU. Much as inter-package communication costs between single die GPUs limited performance in multi-GPU systems, inter-MCM communication costs limit Scale-Out MCM-GPU performance and drastically increase power consumption. Much as Exascale APU is meant to provide a scaling pathway past traditional MCMs, WSGPU is meant to allow massive GPU computation beyond what is possible with MCM-GPU alone.

5. Conclusion

Si-IF many-die processors provide multiple scaling pathways for semiconductor manufacturers. By using intelligent chiplet-based design space exploration, a relatively small set of unique chiplets can provide mix-and-match designs for a wide array of semi-custom ASICs, allowing manufacturers to save money while meeting increasingly specific customer demands. Additionally, waferscale computing may finally be realizable with a Si-IF many-die design. By avoiding costly inter-package communication costs and difficult multi-GPU programming models, WSGPU can provide unmatched performance per transistor and performance per watt for large-scale GPU applications. With traditional scaling methods failing, silicon manufacturers of the future must look toward radical solutions for new scaling pathways: Si-IF many-dies offer a new way forward.

References

[1] Arunkumar, A., Bolotin, E., Cho, B., Milic, U., Ebrahimi, E., Villa, O., Jaleel, A., Wu, C.J. and Nellans, D., 2017, June. MCM-GPU: Multi-chip-module GPUs for continued performance scalability. In *Proceedings of the 44th Annual International Symposium on Computer Architecture* (pp. 320-332). ACM.

[2] Bailey, D.H., Barszcz, E., Barton, J.T., Browning, D.S., Carter, R.L., Dagum, L., Fatoohi, R.A., Frederickson, P.O., Lasinski, T.A., Schreiber, R.S. and Simon, H.D., 1991. The NAS parallel benchmarks. *The International Journal of Supercomputing Applications*, *5*(3), pp.63-73.

[3] Bajwa, A.A., Jangam, S., Pal, S., Marathe, N., Bai, T., Fukushima, T., Goorsky, M. and Iyer, S.S., 2017, May. Heterogeneous Integration at Fine Pitch (≤ 10 µm) Using Thermal Compression Bonding. In *Electronic Components and Technology Conference (ECTC), 2017 IEEE 67th* (pp. 1276-1284). IEEE.

[4] Caldwell, A.E., Kahng, A.B. and Markov, I.L., 1999, January. Design and implementation of the fiduccia-mattheyses heuristic for vlsi netlist partitioning. In *Workshop on Algorithm Engineering and Experimentation* (pp. 182-198). Springer, Berlin, Heidelberg.

[5] Calborean, H. and Vinţan, L., 2010, June. An automatic design space exploration framework for multicore architecture optimizations. In *Roedunet International Conference (RoEduNet), 2010 9th* (pp. 202-207). IEEE.

[6] Carlson, T.E., Heirmant, W. and Eeckhout, L., 2011, November. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for (pp. 1-12). IEEE.

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

[8] Henning, J.L., 2006. SPEC CPU2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News*, *34*(4), pp.1-17.

[9] Jangam, S., Pal, S., Bajwa, A., Pamarti, S., Gupta, P. and Iyer, S.S., 2017, May. Latency, Bandwidth and Power Benefits of the SuperCHIPS Integration Scheme. In *Electronic Components and Technology Conference (ECTC), 2017 IEEE 67th* (pp. 86-94). IEEE.

[10] Kim, M.M., Mehrara, M., Oskin, M. and Austin, T., 2007, June. Architectural implications of brick and mortar silicon manufacturing. In *ACM SIGARCH Computer Architecture News* (Vol. 35, No. 2, pp. 244-253). ACM.

[11] Kumar, R., Tullsen, D.M. and Jouppi, N.P., 2006, September. Core architecture optimization for heterogeneous chip multiprocessors. In *Proceedings of the 15th international conference on Parallel architectures and compilation techniques* (pp. 23-32). ACM.

[12] Lea, R.M., 1990, January. WASP: a wafer-scale massively parallel processor. In *Wafer Scale Integration, 1990. Proceedings.,[2nd] International Conference on* (pp. 36-42). IEEE.

[13] Li, S., Ahn, J.H., Strong, R.D., Brockman, J.B., Tullsen, D.M. and Jouppi, N.P., 2009, December. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore

architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on* (pp. 469-480). IEEE.

[14] Li, Y., Lee, B., Brooks, D., Hu, Z. and Skadron, K., 2006, February. CMP design space exploration subject to physical constraints. In *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on* (pp. 17-28). IEEE.

[15] Mahajan, R., Sankman, R., Patel, N., Kim, D.W., Aygun, K., Qian, Z., Mekonnen, Y., Salama, I., Sharan, S., Iyengar, D. and Mallik, D., 2016, May. Embedded Multi-die Interconnect Bridge (EMIB)--A High Density, High Bandwidth Packaging Interconnect. In *Electronic Components and Technology Conference (ECTC), 2016 IEEE 66th* (pp. 557-565). IEEE.

[16] McDonald, J.F., Rogers, E.H., Rose, K. and Steckl, A.J., 1984. The trials of wafer-scale integration: Although major technical problems have been overcome since WSI was first tried in the 1960s, commercial companies can't yet make it fly. *IEEE Spectrum*, *21*(10), pp.32-39.

[17] "McKinsey on semiconductors, 2014." [Online]. Available: http://www.mckinsey.com/industries/semiconductors/our-insights

[18] Milic, U., Villa, O., Bolotin, E., Arunkumar, A., Ebrahimi, E., Jaleel, A., Ramirez, A. and Nellans, D., 2017, October. Beyond the socket: NUMA-aware GPUs. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture* (pp. 123-135). ACM.

[19] Murthy, K. and Mellor-Crummey, J., 2015, October. Communication avoiding algorithms: Analysis and code generation for parallel systems. In *Parallel Architecture and Compilation (PACT), 2015 International Conference on* (pp. 150-162). IEEE.

[20] Pal, S., Petrisko, D., Bajwa, A.A., Gupta, P., Iyer, S.S. and Kumar, R., 2018, Feb. A Case for Packageless Processors. *High Performance Computer Architecture (HPCA), 2018 IEEE International Symposium on. IEEE, 2018.*

[21] Patil, H. and Carlson, T.E., 2014, February. Pinballs: portable and shareable user-level checkpoints for reproducible analysis and simulation. In *Proceedings of the Workshop on Reproducible Research Methodologies (REPRODUCE)*.

[22] Poovey, J.A., Conte, T.M., Levy, M. and Gal-On, S., 2009. A benchmark characterization of the EEMBC benchmark suite. *IEEE micro*, *29*(5).

[23] Power, J., Hestness, J., Orr, M.S., Hill, M.D. and Wood, D.A., 2015. gem5-gpu: A heterogeneous cpu-gpu simulator. *IEEE Computer Architecture Letters*, *14*(1), pp.34-36.

[24] Tange, O., 2011. Gnu parallel-the command-line power tool. *The USENIX Magazine*, *36*(1), pp.42-47.

[25] Tirkel, I., 2013. Yield learning curve models in semiconductor manufacturing. *IEEE Transactions On Semiconductor Manufacturing*, *26*(4), pp.564-571.

[26] Vijayaraghavany, T., Eckert, Y., Loh, G.H., Schulte, M.J., Ignatowski, M., Beckmann, B.M., Brantley, W.C., Greathouse, J.L., Huang, W., Karunanithi, A. and Kayiran, O., 2017, February. Design and Analysis of an APU for Exascale Computing. In *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on* (pp. 85-96). IEEE.

[27] Williams, S., Waterman, A. and Patterson, D., 2009. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, *52*(4), pp.65-76.

[28] Woo, S.C., Ohara, M., Torrie, E., Singh, J.P. and Gupta, A., 1995, July. The SPLASH-2 programs: Characterization and methodological considerations. In *ACM SIGARCH computer architecture news* (Vol. 23, No. 2, pp. 24-36). ACM.

[29] Zhang, T., Benini, L. and De Micheli, G., 2001. Component selection and matching for IP-based design. In *Design, Automation and Test in Europe, 2001. Conference and Exhibition 2001. Proceedings* (pp. 40-46). IEEE.