UNIVERSITY OF CALIFORNIA

Los Angeles

Scale-Out Packageless Processing

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical and Computer Engineering

by

Saptadeep Pal

2021

ABSTRACT OF THE DISSERTATION

Scale-Out Packageless Processing

by

Saptadeep Pal
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2021
Professor Puneet Gupta, Co-Chair
Professor Subramanian Srikantes Iyer, Co-Chair

Demand for increasing system performance is far outpacing the capability of conventional methods for performance scaling. Traditionally, performance and energy scaling has relied on transistor and silicon scaling. However, developing chips, often very large ones in the advanced technology nodes is becoming very challenging and costly. Moreover, system performance is often limited by inter-die connections. Today, dies with different functionality are packaged and integrated using PCBs. Unlike silicon features, package and PCB features have barely scaled (about 4-5x) over the past few decades. This severely limits performance and efficiency of processor systems. Moreover, next-generation of applications driven by artificial intelligence, and other data intensive applications are driving the demand for very large scale-out systems. Traditional scale-out system building and integration methodologies are failing to deliver the performance these applications demand. As a result of the above trends, future performance, power, and cost improvements cannot come from improvements in transistor technology alone. Then, how do we enable "System scaling"?

In this dissertation, first we show that packages inhibit system scaling as it reduces the potential memory bandwidth of a processor by at least one order of magnitude, allowable thermal design power (TDP) by up to 70%, and area efficiency by a factor of 5 to 18. We therefore propose *packageless processors* - processors where packages have been removed and dies directly mounted on a silicon board using a novel integration technology, Silicon Interconnection Fabric (Si-IF). We show that Si-IF-based packageless processors outperform their packaged counterparts by up to

58% (16% average), 136% (103% average), and 295% (80% average) due to increased memory bandwidth, increased allowable TDP, and reduced area respectively. We also extend the concept of packageless processing to the entire processor and memory system, where the area footprint reduction was up to 76%. To guide technology direction for dielet integration substrate technologies, we also developed a die-to-die interconnect pathfinding tool to explore the effects of physical trade-offs such as bump pitch, wire pitch, I/O ESD capacitance etc. We show that incessant reduction of bump and wire pitch below $10\mu m$ wouldn't be helpful for interconnect performance and we need to develop techniques and technologies to minimize reliance on large ESD structures in the chiplet I/Os as ESD capacitance starts dominating performance and energy cost of these die-to-die interconnect links. Next, we show that fine pitch chiplet integration technologies allow us to disintegrate large SoCs in to chiplets with minuscule hit in performance. This opens up the opportunity to build a chiplet eco-system, where application-optimized systems can be built by selecting a subset of chiplets from a chiplet pool. Such an eco-system however needs us to find the suitable minimal set of chiplets to build in order to target a variety of workloads efficiently. To that end, we developed the first chiplet selection framework to target a large variety of applications. We show that up to 35% improvement in EDP can be obtained from application-specific system customization and when total cost of design and manufacturing is considered, up to 72% benefit in cost is possible over SoCs.

Part 2 of the dissertation focuses on scale-out processing systems. To target scale-out systems, we propose chiplet-based waferscale processors to dramatically reduce communication overheads. The Si-IF technology can be used to build scale-out processors up to a size of an entire wafer. However, building such a large consolidated waferscale system has its own challenges. Using a waferscale GPU as a case study, we showed that while a 300 mm wafer can house about 100 GPU modules (GPMs), only a much scaled down GPU architecture with about 40 GPMs can be built when physical concerns are considered. We analyzed the design space of power-delivery network, cooling and trade-offs of yield and inter-GPM network topologies, and proposed optimized waferscale GPU architecture. We also optimized thread scheduling and data placement policies. Overall, our simulations show that an optimized waferscale architecture can provide up to 19x speedup compared to traditionally integrated systems. Then, we architected and designed a 14,336-

core shared memory waferscale system in order to understand the design challenges of waferscale processors. Several aspects of the design were built from the ground up due to the scale of the system:power delivery and on-chip regulation methods, reliable waferscale clock distribution, waferscale fault-tolerant network design, chiplet and waferscale system test mechanisms, and multiple physical and architectural techniques to enhance system yield. The chiplets were taped out in TSMC N40-LP process and a smaller prototype system has been functionally verified. Next, we have focused on understanding the scalability characteristics of deep learning (DL) training applications and exploring the cross-stack impact of hardware-software-technology co-design at-scale. With the aid of an optimal operation-to-device placement tool, we have proposed a framework which allows us to figure out when to use model parallelism with data parallelism instead of data parallelism alone in order to minimize end-to-end training time. Next, we developed a system-technology co-optimization tool which explores the cross-stack impact of technology scaling, model scaling and architectural innovations on end-to-end DL training time. Using this tool, we can perform rapid-yet-accurate design space exploration and find optimal architectures under given logic, memory, and inter-chip interconnect technology parameters.

Together, the techniques and methodologies developed in this dissertation lays the ground work for a revolutionary new way of thinking about system scaling. Packageless processing and scale-out waferscale architectures can indeed provide orders of magnitude improvement in performance and energy efficiency required by next-generation of applications. Moreover, the cross-stack pathfinding tools provide rapid-assessment frameworks to understand bottlenecks across different levels in the system and helps guide technology optimal decisions for processing systems.

The dissertation of Saptadeep Pal is approved.

Anthony John Nowatzki

Mani B. Srivastava

Subramanian Srikantes Iyer, Committee Co-Chair

Puneet Gupta, Committee Co-Chair

University of California, Los Angeles

2021

To my parents, brother and friends who inspire me to keep on pushing forward,

and without whom none of this would have happened

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This dissertation is the culmination of many years of hard work, the support and guidance of my advisors and the doctoral committee, and the love and encouragement of my family and friends. This journey has been a roller-coaster ride, from being a fresh undergraduate student to finishing my masters and subsequently the doctoral program. Tough times lead to sleepless nights, but the results of the continued perseverance at solving problems and coming up with new solutions has been nothing less than mind-boggling. Now, standing at the end of the tunnel, I can definitely say that the last six years of investment has been worth the returns. Overall, this experience has been extremely fulfilling and I owe gratitude to a number of people without whom my doctoral studies would not have been possible.

I would first like to thank my doctoral committee: Profs. Puneet Gupta (chair and advisor), Subramanian S. Iyer (co-chair and co-advisor), Mani Srivastava, and Tony Nowatzki for their outstanding teaching and research guidance. Prof. Gupta has been a great advisor, mentor and a sounding board. I am grateful to him for the constant support, patience, guidance and providing me with all the resources. This dissertation would not have been possible without his excellent help and foresight. He helped me in becoming a successful researcher. I would like to thank Prof. Subramanian (Subu) S. Iyer for being an instrumental co-advisor who introduced me to the world of electronics packaging. Subu's relentless focus on hands-on engineering and building systems has pushed me to do things which I may not have thought about. Most of the dissertation stands on the collaborative work I have done with members across two laboratories at UCLA and it has been an absolute privilege. I would also be indebted to Prof. Rakesh Kumar at UIUC. Venturing into the realms of computer systems architecture research would not have been possible without him. He has been an ardent listener and a advisor and I have learnt many life lessons from him.

I would then like to thank all my internship mentors and collaborators. At Nvidia Corp., I worked under the guidance of Eiman Ebrahimi, Arslan Zulfiqar and David Nellans. The knowledge and insights from this internship immensely helped me with my research in systems architecture, in particular deep learning systems. Over the past two years since 2019, I have closely worked with Newsha Ardalani on developing DeepFlow and have learnt a lot over the course of the project.

She has been an excellent collaborator, friend and mentor. Thanks also to Qualcomm Innovation Fellowship for inviting me to present at Qualcomm Research (San Diego) in 2019 and 2020. I would also like to thank members of CDEN and CHIPS, who has provided very important feedback that has helped shape some of my research.

I would like to acknowledge the support of my lab mates: Dr. Mark Gottscho, Dr. Yasmine Badr, Dr. Shaodi Wang, Dr. Wei-Che Wang, Dr. Irina Alam, Tianmu Li, Wojciech Romaszkan, Shurui Li, Alexander Graening, SivaChandra Jangam, Krutikesh Sahoo, Steven Moran, and many from the CHIPS laboratory. I am thankful for their friendship and their inputs during technical discussions. I would also sincerely thank my collaborators form the PASSAT group at UIUC - Daniel Petrisko, Nicholas Cebry and Jingyang Liu. Particularly, I would thank Matthew Tomei with whom I have spent countless days and nights, brainstorming ideas, writing and debugging code and papers, and ofcourse pitching a startup company where we wanted to make novel waferscale systems. Finally, I will be forever grateful to Irina Alam for being my biggest support, friend and family here in LA in the past five years, for always inspiring and motivating me, and providing valuable inputs through lengthy discussions, and for proofreading all my papers.

Lastly and most importantly, I would like to thank my parents, Nupur Pal and Swapan Kumar Pal, and my brother Debapriya Pal for their unconditional love, unwavering support, and encouragement. I can never find enough words to express my gratitude towards them for their sacrifices and for the key role that they played in shaping my future. None of my accomplishments would have been possible without them.

Copyrights and Re-use of Published Material

This dissertation contains significant material that has been previously published or is intended to be published in the future. Chapters 2 (Packageless Processors) contain material that were published in [9]. Chapter 3 includes material published in [10]. Chapter 4 (DSE for chiplet-assemmbly) appeared in [10]. Chapters 5 (Waferscale-GPU) appeared in [11] and 6 (Waferscale Design methodology) appeared in [12]. Chapter 7 (DL Multi-GPU Parallelism) includes material published in [13]. Chapter 8 (DeepFlow) is part of a paper which is under submission.

Some of the work in my PhD that was conducted in collaboration with other individuals (where I contributed, but did not lead) are not included in the body of this dissertation.

VITA

2015	B.Tech., Indian Institute of Technology, Patna, India.
2015	Institute Silver Medal, Indian Institute of Technology, Patna, India.
2015, 2019	UCLA GuruKrupa Fellowship
2017	M.S., Electrical Engineering, UCLA
2017	Intel Best Student Paper Award, ECTC
2018	PhD Intern, NVIDIA Corporation
2019	Qualcomm Innovation Fellowship
2019	Winner at EPFL Engineering PhD Summit on Intelligent Systems
2019	Best Technical Presentation, UCLA ECE Annual Research Review

PUBLICATIONS

Saptadeep Pal, Jingyang Liu, Irina Alam, Nicholas Cebry, Haris Suhail, Shi Bu, Subramanian S. Iyer, Sudhakar Pamarti, Rakesh Kumar, and Puneet Gupta. "Designing a 2048-Chiplet, 14336-Core Waferscale Processor." in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 1183-1188.

Saptadeep Pal, Daniel Petrisko, Adeel A. Bajwa, Puneet Gupta, Subramanian S. Iyer and Rakesh Kumar, "A Case for Packageless Processors," in *2018 IEEE International Symposium on High*

Performance Computer Architecture (HPCA), Vienna, Austria, 2018, pp. 466-479.

Saptadeep Pal, Daniel Petrisko, Matthew Tomei, Puneet Gupta, Subramanian S. Iyer, and Rakesh Kumar, “Architecting waferscale processors-a GPU case study." in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 250-263.

Saptadeep Pal, Irina Alam, Krutikesh Sahoo, Haris Suhail, Rakesh Kumar, Sudhakar Pamarti, Puneet Gupta, and Subramanian S. Iyer. "I/O Architecture, Substrate Design, and Bonding Process for a Heterogeneous Dielet-Assembly based Waferscale Processor." in *2021 IEEE 71st Electronic Components and Technology Conference (ECTC)*, pp. 298-303.

Saptadeep Pal, Daniel Petrisko, Rakesh Kumar, and Puneet Gupta. "Design space exploration for chiplet-assembly-based processors." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28, no. 4 (2020): 1062-1073.

Saptadeep Pal, and Puneet Gupta. "Pathfinding for 2.5 D interconnect technologies." in *2020 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*, pp. 1-8. IEEE, 2020.

Saptadeep Pal, Eiman Ebrahimi, Arslan Zulfiqar, Yaosheng Fu, Victor Zhang, Szymon Migacz, David Nellans, and Puneet Gupta. "Optimizing multi-GPU parallelization strategies for deep learning training." *IEEE Micro* 39, no. 5 (2019): 91-101.

SivaChandra Jangam, **Saptadeep Pal**, Adeel A. Bajwa, Sudhakar Pamarti, Puneet Gupta and Subramanian S. Iyer, “Latency, Bandwidth and Power Benefits of the SuperCHIPS Integration Scheme,” in *2017 IEEE 67th Electronic Components and Technology Conference (ECTC)*, Orlando, FL, 2017, pp. 86-94.

CHAPTER 1

Introduction

Conventional computing is at a tipping point. On one hand, applications are fast emerging that have higher performance, bandwidth, and energy efficiency demands than ever before. On the other hand, the end of Dennard scaling as well as the slowdown of Moore's law diminishes the prospect of easy performance, bandwidth, or energy efficiency scaling in the future. Traditional framework of getting better performance and efficiency by developing and manufacturing large system-on-chips (SoCs) at the latest technology nodes is becoming prohibitively costly, more so for low- to medium-volume products. Moreover, fueled by the exponential growth of deep learning and big data analytics, scale-out systems composed of many SoCs are in great demand. Traditional integration technologies and methodologies of building scale-out systems, however, are failing to deliver the performance these applications demand. As a result of the above trends, future performance, power, and cost improvements cannot come from improvements in SoC technology alone. Then, how do we enable "System scaling"? To address this growing need, the research in this dissertation focuses on developing technology solutions and frameworks for system-technology co-optimization (STCO) which intersects the broad areas of computer architecture, VLSI design, and packaging. A novel system-level heterogeneous integration platform that enables packageless processing is introduced in Section 1.1. Section 1.2 briefly highlights the benefits of the novel chiplet-based waferscale processor architectures proposed and developed as part of the research in this dissertation. Section 1.3 then explains the importance of developing a cross-stack pathfinding frameworks for STCO. Overall, *Scale-Out Packageless Processing* provides a radically different dimension to system scaling and is described in Section 1.4 that summarizes all research projects covered in this dissertation.

1.1 A Novel Heterogeneous Integration Platform for Packageless Processing

Increasing SoC design and manufacturing cost in the latest technology processes is now making the industry look at alternative approaches to system design. An alternative approach, system-in-package (SiP), where multiple heterogeneous chiplets are integrated inside a package is gaining traction. SiP technologies, however, often lead to performance and/or energy efficiency overheads and are yet to deliver on their promise. Moreover, package-to-package links on printed circuit boards (PCBs) used to build integrated systems suffer from poor bandwidth, latency, and energy characteristics and often bottlenecks system performance. This is because of a striking observation that in the last two decades while silicon chips have dimensionally scaled by >1000X, package and PCB features have merely managed 5X. This absence of "system scaling" can severely limit the performance of processor systems. Though this realization has motivated the push toward 3D and 2.5D integration schemes, they however do not address the root cause.

This dissertation proposes a radically different approach – eliminating packages altogether and integrating all system components on a silicon board. To that end, we developed and prototyped a novel integration technology – Silicon Interconnect Fabric (Si-IF), as a potential replacement for PCB-based integration as the enabling technology for packageless processing. Si-IF technology allows integration of multiple bare silicon dies (chiplets) on a silicon substrate using very fine pitch copper pillar based I/O. The Si-IF is fabricated using a conventional and mature back end of line (BEOL) process which can have up to four levels of conventional copper dual damascene based interconnect. Si-IF allows I/O pitches to be less than 10 μm and wire pitches to be as low as few hundred nanometers to a few microns. This enables us to build near on-chip-like parallel interfaces between chiplets. The custom designed area and energy-efficient I/O circuitry helps in delivering <0.1 pJ/bit energy efficiency of inter-chiplet communication on Si-IF. Besides, different chiplets can come from different technologies, for e.g. Si, GaN, InP, SiC etc. In fact, an SoC can be disintegrated and multiple component chiplets can now come from different technology nodes to optimize for cost and power. For e.g., a processor core can be in 32nm technology while L2 cache chiplet can be implemented in 22nm. Overall, Si-IF technology dramatically reduces the overhead of splitting an SoC into multiple chiplets and opens a foundational platform to minimize system integration cost where hard-IPs, aka. chiplets can now be integrated to build a system without the penalties associated with conventional packaging.

The dissertation also proposes packageless processor architecture where we leveraged the unique properties of the Si-IF platform to (a) significantly improve memory bandwidth, which in turn helped us re-design the cache hierarchy, (b) improve thermal design power (TDP) by leveraging silicon's heat conduction properties for better cooling, and (c) increase processor core count by leveraging the area savings from removing packages. This improves processor performance by up to 3x while taking up 76% less area than its conventional counterpart. As human society becomes more reliant on computing and marches towards embedding computing in all facets of the physical world, packageless processing would help us achieve unprecedented performance and efficiency that cannot be delivered by today's technologies.

1.2 Chiplet-based Waferscale Processing

Today, the demand for parallel hardware is more than ever before. Waferscale processor systems can provide the large number of cores, and memory bandwidth and latency required by today's highly parallel workloads without the area and energy overhead incurred in conventional integration technologies. Older (1980s) and some recent works adopted waferscale integration to provide very large performance and energy efficiency gains compared to conventional systems. All these works, however, rely on monolithic waferscale integration. Monolithic waferscale chips are homogeneous and cannot integrate disparate technology such as dense memory needed for the next generation of data-intensive applications. Also, often they suffer from yield issues and must dedicate a sizeable number of resources for redundancy. This dissertation, for the first time, targets waferscale processing in a completely different way. Using the Si-IF technology, we can integrate a large number of heterogeneous known-good-chiplets on a full wafer-sized interconnect platform. This approach can help integrate TBs (instead of GBs) of memory (DRAM, Flash) alongside PFLOPs of compute (in the latest CMOS process) to target the most difficult problems in HPC and ML/AI.

A waferscale architectural design space is heavily influenced by different physical constraints such as thermal and power delivery. We show that using a waferscale GPU architecture. Our optimized waferscale GPU architecture performs up to 19x better than an optimized scale-out GPU system using traditional packaging technologies. We also explored the design methodology challenges of building a chiplet-based waferscale system by attempting to design a 14,336-core shared memory waferscale prototype system. The scale of this prototype system forced us to rethink several aspects of the design flow such as power delivery and on-chip regulation methods, reliable waferscale clock distribution, waferscale fault-tolerant network design, chiplet and waferscale system test mechanisms, and multiple physical and architectural techniques to enhance system yield.

I believe that we are just at the beginning of a waferscale revolution. TSMC in 2020 announced a chiplet based waferscale integration platform, which Tesla has adopted for their Dojo supercomputer. Several new system architectures can be built with waferscale technologies which were earlier not possible because of the limitations of conventional technologies.

1.3 System-Technology Co-Optimization

Fueled by the AI revolution and exploding growth of warehouse-scale computing, the semiconductor industry is pouring in 10s of billions of dollars in developing new specialized architectures and systems tailored towards training and inference of large models and processing as well as building efficient cloud infrastructure. One countervailing trend is that migrating designs to cutting-edge processes do not always yield good returns on investment. Moreover, several promising candidate technologies (e.g., MRAM memory, CNT interconnects) and packaging and system solutions are being explored across the globe. Commercialization of these new technologies would need further large investments across the board. To aid with these investment decisions, the industry needs to understand the performance and energy impacts of these technologies across the design space of architectures and applications. This calls for rapid design space exploration and pathfinding frameworks which would allow us to quickly evaluate the effects of these new technologies on the end-to-end application-level performance. To that end, we co-developed and open-sourced one such tool, DeepFlow, for STCO targeted towards large-scale deep learning model training. Given the characteristics of the technology node (logic and SRAM density), packaging technology, system-level constraints such as area, power, etc., and the deep learning model graph, the tool can rapidly explore the architectural search space and software parallelization strategies and outputs the best end-to-end training time.

1.4 Dissertation Outline

The research in this dissertation focuses on developing and building novel technology solutions and system-technology co-optimization frameworks that provide a revolutionary new way of enabling the much-needed system scaling.

In Part 1, the three chapters highlight the bottlenecks of conventional packages and propose a novel heterogeneous integration platform that allows elimination of conventional packages and replaces PCB with a silicon based interconnection substrate i.e., Si-IF.

- Chapter 2 makes the case for packageless processors. Using a baseline packaged processor architecture, we show that removing the package and replacing the PCB with Si-IF could enable large gains in terms of performance coming from increased memory bandwidth, thermal design power (TDP) and area benefits.
- Chapter 3 focuses on comprehensively investigating silicon based 2.5D interconnects such as silicon interposer, EMIB and Si-IF. A pathfinding methodology for 2.5D interconnect technologies is developed and used to study inter-chiplet interconnect performance and energy as a function of dimensional and technology parameters.
- Chapter 4 presents an optimization framework for multi-system/chiplet based processor ecosystem. Multiple systems can be used to target a diverse set of applications where each system is used to run a subset of applications efficiently. Recent 2.5D integration based assembly techniques allow composing systems using multiple chiplets. Finding the smallest number of chiplets to build multiple systems is a problem which we also solve using our optimization framework.

In Part 2, the two chapters focus on analyzing and building chiplet-based waferscale systems using Si-IF technology where a large number of heterogeneous known-good-chiplets are integrated on a full wafer-sized interconnect platform.

- Chapter 5 proposes a waferscale GPU architecture. It is shown that a waferscale architectural design space is heavily influenced by different physical constraints such as thermal and power delivery. Novel power delivery mechanisms and network design as well as cooling solutions are proposed to maximize the amount of compute and memory that can be reliably integrated on a wafer. Optimized waferscale-network topologies are discussed that help to balance the performance and yield of the substrate. Then, optimized thread scheduling and data placement techniques are designed for waferscale GPUs.

- Chapter 6 focuses on designing a 14,336-core shared memory waferscale prototype system. Several aspects of the design were built from the ground up due to the scale of the system: power delivery and on-chip regulation methods, reliable waferscale clock distribution, waferscale fault-tolerant network design, chiplet and waferscale system test mechanisms, and multiple physical and architectural techniques to enhance system yield. The chiplets were taped out in TSMC N40-LP node and a functional prototype was assembled in-house.

In Part 3, the two chapters focus on understanding the scalability characteristics of deep learning (DL) training applications and exploring the cross-stack impact of hardware-software-technology co-design at-scale.

- Chapter 7 studies which parallelization strategies to adopt to minimize end-to-end training time for a given DL model on available hardware. It is shown that hybrid parallelization (data parallelism[DP] and model parallelism[MP]) outperforms DP alone at different scales for different DL networks. An integer linear programming based tool is also developed to find optimal operation-to-device placement to maximize MP speedup.

- Chapter 8 explores the cross-stack impact of technology scaling, model scaling and architecture innovations from a holistic perspective, and at the same time considering real-world design constraints like area and power budget for deep learning training. Using this tool, we can perform rapid-yet-accurate design space exploration and find optimal architectures under given logic, memory, and inter-chip interconnect technology parameters.

Overall, the research in this dissertation focuses on system-technology co-optimization and provides a dramatically new dimension to system scaling. Such revolutionary new technology solutions and frameworks are absolutely needed to achieve performance, power and cost improvements in future systems as conventional technology scaling is becoming increasingly harder. The proposed chiplet-assembly based packageless processing and the scale-out waferscale architectures leveraging the novel integration technology developed in this dissertation provide the much needed performance and energy improvements for next-generation applications and are expected to create new areas of research.

CHAPTER 2

A Case for Packageless Processors

Demand for increasing performance is far outpacing the capability of traditional methods for performance scaling. Disruptive solutions are needed to advance beyond incremental improvements. Traditionally, processors reside inside packages to enable PCB-based integration. We argue that packages reduce the potential memory bandwidth of a processor by at least one order of magnitude, allowable thermal design power (TDP) by up to 70%, and area efficiency by a factor of 5 to 18. Further, silicon chips have scaled well while packages have not. We propose *packageless processors* - processors where packages have been removed and dies directly mounted on a silicon board using a novel integration technology, Silicon Interconnection Fabric (Si-IF). We show that Si-IF-based packageless processors outperform their packaged counterparts by up to 58% (16% average), 136% (103% average), and 295% (80% average) due to increased memory bandwidth, increased allowable TDP, and reduced area respectively. We also extend the concept of packageless processing to the entire processor and memory system, where the area footprint reduction was up to 76%.

2.1 Introduction

Conventional computing is at a tipping point. On one hand, applications are fast emerging that have higher performance, bandwidth, and energy efficiency demands than ever before. On the other hand, the end of Dennard scaling [14] as well as Moore's law transistor scaling diminishes the prospect of easy performance, bandwidth, or energy efficiency scaling in future. Several promising and disruptive approaches are being explored, including (but not limited to) specialization [15], approximation [16], 3D integration [17], and non-CMOS devices [18].

Current systems place processor and memory dies inside packages, which allows them to be connected to the PCB and subsequently to other dies. A striking observation is that in the last two decades while silicon chips have dimensionally scaled by 1000X, packages on printed circuit boards (PCBs) have merely managed 4X [19]. This absence of “system scaling” can severely limit performance of processor systems. This realization has motivated the push toward 3D and 2.5D integration schemes which alleviate the problem but do not address the root cause. In this work, we propose another approach - removing the package from the processor altogether.

At first glance, removing the package from the processor may seem both simple in implementation and, at best, incremental in benefits. However, neither is true. Packages significantly limit the number of supportable IOs in the processor due to the large size and pitch of the package-to-board connection relative to the size and pitch of on-chip interconnects ($\sim$10X and *not* scaling well). In addition, the packages significantly increase the interconnect distance between the processor die and other dies. Eliminating the package, therefore, has the potential to increase bandwidth by at least an order of magnitude(Section 2.2). Similarly, processor packages are much bigger than the processor itself (*5 to 18 times bigger*). Removing the processor package frees up this area to either be used in form factor reduction or improving performance (through adding more computational or memory resources in the saved area). Lastly, packages limit efficient heat extraction from the processor. Eliminating the processor package can significantly increase the allowable thermal design power (TDP) of the processor (up to 70%). Increase in allowable TDP can be exploited to increase processor performance significantly (through frequency scaling or increasing the amount of computational or memory resources). Unfortunately, simply removing the processor package hurts rather than helps as we point out in Section 2.3. We develop a new silicon interconnect fabric to replace the PCB and make package removal viable in Section 2.4. Essentially, we place and bond bare silicon dies directly on to a silicon wafer using copper pillar-based I/O pins.

This chapter makes the following contributions:

- We make a case for packageless processors. We argue that modern processor packages greatly hinder performance, bandwidth, and energy efficiency scaling. Eliminating packages can enable us to recoup the lost performance, bandwidth, and energy efficiency.

- We present Si-IF, a novel integration technology, as a potential replacement for PCB-based integration and as the enabling technology for packageless processing.

- We quantify the bandwidth, TDP, and area benefits from packageless processing. We show that up to one to two orders of magnitude, 70%, and 5-18x benefits respectively, are possible over conventional packaged processors. These benefits translate into up to 58% (16% average), 136% (103% average), and 295% (80% average) performance benefits, respectively, for our benchmarks.
- We also extend the concept of packageless processing to the entire system on the board; reduction in system-level footprint was up to 76%.

2.2 Packaging Processors and its Limitations

Traditionally, processor and memory dies are packaged and then placed on printed circuit boards (PCB) alongside other packaged components. The PCB acts as the system level interconnect and also distributes power to the various packages using the board level power distribution network (PDN). The package is the interface to connect the dies to the PCB. A schematic cross-section of a typical packaged processor on a PCB is shown in Figure 2.3. Packages serve three primary functions:

- *Packages act as a space transformer for I/O pins:* The diameter of chip IOs is relatively small (50μm-100μm) [20]. However, the bump sizes required to connect to the PCB often range between at least a few hundred microns to about a millimeter [21–23]; large bumps are needed due to PCB's high surface warpage. To enable connectivity in spite of the large difference between chip I/O diameter and the required bump size to connect to PCB, packages are needed. Packages are connected to the silicon die using C4 (controlled collapse chip connection) micro bumps, while the package laminate acts as a redistribution layer (RDL) and fans out to a BGA (ball-grid array) [24] or LGA (land-grid array) [25] based I/O with typical pitch of about $\sim 500\ \mu$m - 1 mm. Packages perform the same function even in the scenario where they do not use solder balls, but use sockets with large pins to prevent breakage from manual installation and handling.

- *Packages provide mechanical support to the dies:* Packages provide mechanical rigidity to the silicon dies, protect them from the external environment (moisture and other corrosive agents), and provide a large mechanical structure for handling. Also, the coefficient of thermal expansion (CTE) of the FR4 material used to make the PCB is $\sim$ 15-17 ppm/°C, while that of Silicon is about 2.7 ppm/°C. This large mismatch in CTE between the materials leads to large stresses. Packages provide some mechanical stress buffering, and thus help in mitigating the thermal stresses.

- *Easier testability and repairability:* Since test probe technology has not scaled well [19, 26, 27], it has become harder to probe smaller I/O pads on bare dies. The larger IOs on the packages are easier to probe using conventional probing techniques. Also, while dies come in different sizes, they go into standard packages which can then be tested using standard test equipment.

  Similarly, solder-based joints and pin-based sockets allow for in-field repairability. Solder joints can be simply heated up, melted and taken off while sockets allow plug-n-play.

Historically, the above advantages have been significant enough that most processor systems, excluding some ultra-low-power processors [28, 29], have been package-based. However, packaging processor dies leads to several significant limitations, many of which are becoming worse, even debilitating.

- *Packages reduce I/O Density*: Use of packages inherently limits the maximum number of supportable processor IOs because of the large size and pitch of the package-to-board connections (BGA balls/ LGA pins). The BGA/LGA technologies have not scaled well over the past few decades. On the other hand, the demand for IOs in high-performance processor systems is growing rapidly. Figure 2.1 shows the relative scaling of the number of processor I/O pins in the largest Xeon processor available in a given year against the density scaling (number of IOs/mm^2) of the BGA and LGA technologies. As can be seen, the gap between the demand in the number of IOs versus pin density is increasing every year. This widening *I/O gap* limits the amount of power and the number of signals that can be delivered to the processor chip; this can be a severe limitation for future processors that demand high memory

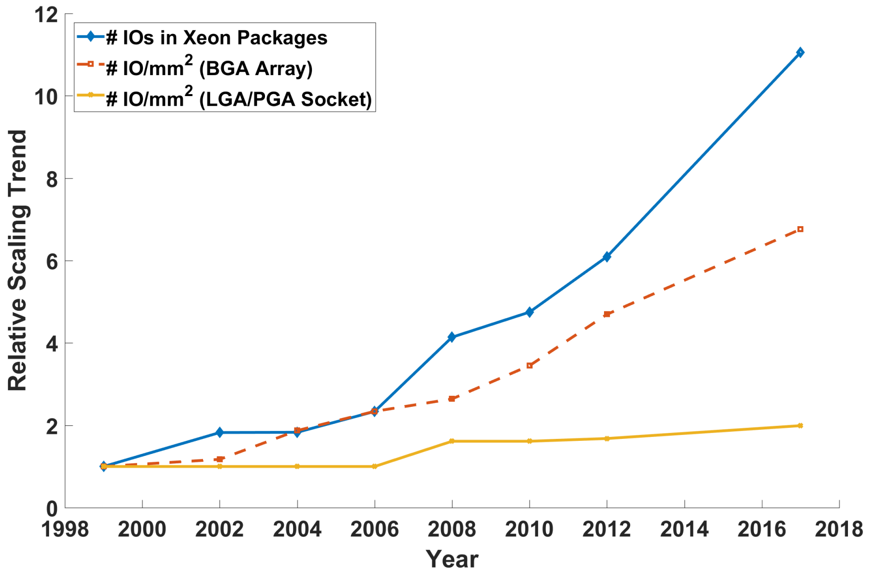
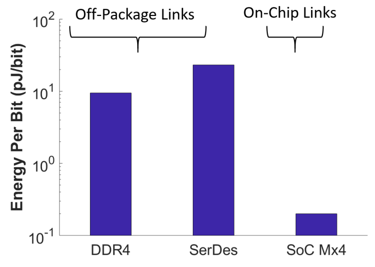


Figure 2.1: I/O demand is growing faster than the I/O pin density scaling.

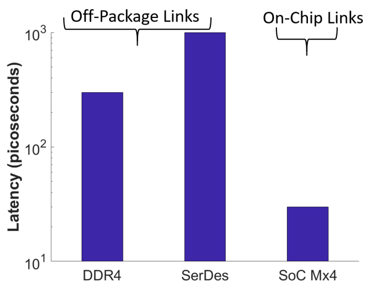
and communication bandwidth. Alternatively, processor packages need to become larger; this, however, significantly affects the form factor, complexity, and cost of packages and the length of inter-package connections. In both these cases, the overheads may be become prohibitive in near future [19, 30].

- *Packages increase interconnect length*: Increasing the size of the package (the package to die ratio is often $>$5, even up to 18 in some cases – (Table 2.1)) leads to a significant increase in the interconnect length between two dies inside separate packages. This is because the die to die connection now needs to traverse the C4 micro-bumps, package RDL, BGA balls, and PCB traces. As the interconnect links become longer, they become more noisy and lossy, which then affects link latency, bandwidth and energy. This problem is aggravated by the fact that a fraction of the interconnect now uses wire traces on PCBs, which are 10X-1000X coarser than the widest wire at the global interconnect layer in SoC chips.

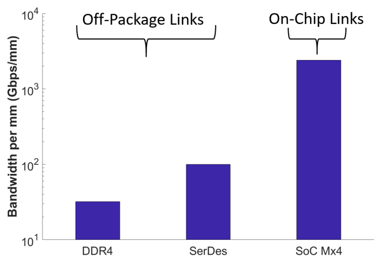
  Figure 2.2 compares the energy, latency, and aggregate bandwidth of package-to-package communication links through PCB vs global routing level interconnect wire (Mx4) in an SoC. As seen from the figure, both energy and latency are disparately high for off-package links as compared to the on-die interconnects, while bandwidth is severely limited - these



(a) Energy per bit



(b) Latency



(c) Aggregate bandwidth per mm die edge

Figure 2.2: Comparison of communication link energy, latency and bandwidth for on-chip versus off-package links.

gaps between off-package links and on-die interconnects must be bridged to enable continued performance scaling.

- *Packages trap heat*: A package traps heat generated by the processor and thus adds to the thermal resistance between the processor die and the heat sink. Figure 2.3 shows the thermal resistance model of a packaged processor system. In such systems, heat conductively flows upward from the processor die through the package lid and thermal interface materials (TIMs) to the heat sink. The typical thermal resistance values for a canonical 100-130W processor are shown in Figure 2.3. Thus, for every 10 W of dissipated power, the package lid adds about 1°C to the chip junction temperature. For high-performance processors with TDP ratings in excess of 100 W, the effect of package thermal resistance can cause major reliability issues due to high chip junction temperatures; this limits the TDP, and, therefore, performance of a processor.

  Moreover, the downward flow of heat encounters high thermal resistivity from the package laminate and the PCB. In fact, the downward heat flow path has about 7-8x higher thermal resistivity than the upward flow. This further exacerbates the above reliability problems from high package thermal resistance. Disruptive solutions that reduce the overall thermal resistance are needed to allow higher sustainable TDP, and, therefore, higher performance at reliable chip-junction operating temperature.

Table 2.1: Package-to-Die Size ratio

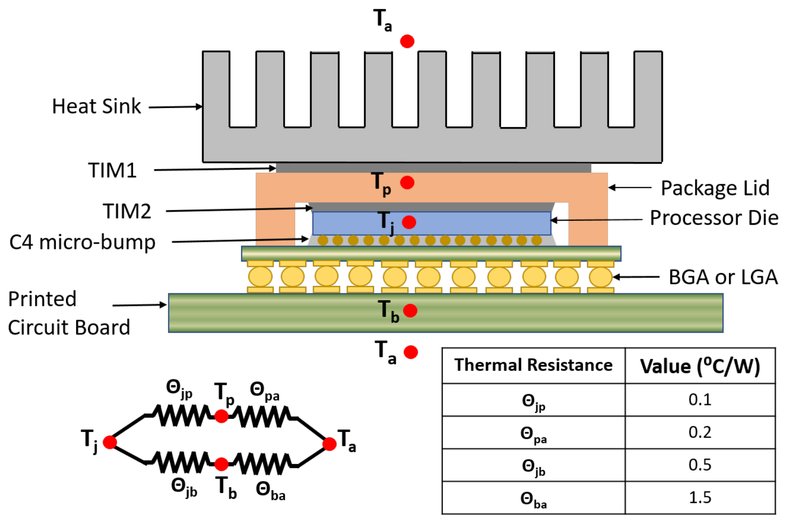
Product Name	**Package-to-Die Size Ratio**
Intel Knight's Landing [31]	7
Intel Broadwell [32]	7 - 10
Intel Atom Processor [33]	5 - 18
DRAM Package [34]	2.5 - 3.6

- *Packages increase system footprint*: As mentioned earlier, package-to-die size ratio has been increasing to accommodate the high I/O demands of today's processors. Some examples of die-to-package ratio in commercially available processors are shown in Table 2.1. Thus, the overall package footprint is much larger than that of the processor die. Also, since the interconnect width and length are relatively large on PCBs, the total interconnect area is a significant portion of the overall PCB area (see Figure 2.14a). As I/O demands increase, an increasing amount of system footprint would be taken up by packages, interconnects and passives. Disruptive solutions may be needed to reduce the area cost of these non-compute components to meet the computation density demands of future applications.

Though packages have been an integral part of computing systems for decades, they are becoming a bottleneck for system and performance scaling due to the reasons above. In this work, we rethink the value of packages for today's and emerging processors, and ask the question - should we build future processor systems without packages?

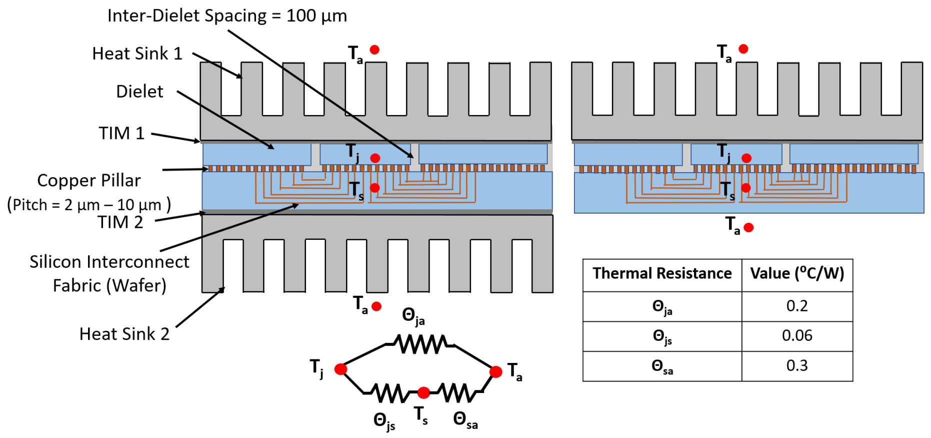
2.3 Why Not Simply Remove the processor package?

While some ultra-low-power processors with a small number of I/O pins can be directly mounted on a PCB without packaging [28, 29], it is difficult to do so for high power, high performance processor systems without prohibitive performance and reliability costs. Simply mounting bare die on PCB will dramatically reduce I/O availability proportionately to die-to-package size ratio (e.g., see Table 2.2 for some commercial processor examples) as the PCB I/O size is still limited to



Thermal Resistance	Value ($^oC/W$)
Θ_{jp}	0.1
Θ_{pa}	0.2
Θ_{jb}	0.5
Θ_{ba}	1.5

Figure 2.3: Cross-section of a packaged die with heat sink placed on a PCB is shown, alongside the thermal resistance model. T_a, T_p, T_j, T_b denotes the ambient, package lid, chip-junction, PCB temperature respectively. The thermal resistance values for a typical processor package is shown alongside. [1, 2]



Thermal Resistance	Value ($^oC/W$)
Θ_{ja}	0.2
Θ_{js}	0.06
Θ_{sa}	0.3

Figure 2.4: Cross-section of an Si-IF system, alongside the thermal resistance model. T_a, T_j, T_s denote the ambient, chip-junction and silicon substrate temperatures respectively. Heat sink can be directly attached to the top of the dielets or both at the top and bottom of the Si-IF.The thermal resistance values for a typical system on Si-IF is shown alongside [1–3]

500μm (usually much larger). Further, the large CTE mismatch between silicon die and organic PCB can become a reliability bottleneck causing thermal stress-induced I/O failures.

Table 2.2: Analysis of board level I/O availability

Product Name	**# Package IOs**	**Die Area (mm^2)**	**# Die BGA Balls**	**Enough Area for I/O?**
Knight's Landing [31]	3647	682	2728	No
Xeon E5-2670 [35]	2011	306	1224	No
Atom N280 [33]	437	26	104	No

In order to realize a packageless processor and its benefits, one would need to replace PCB-based integration with a new integration technology that offers high density interconnect and mechanical robustness.

In the next section, we will describe a novel integration technology (and the accompanying interconnect) we have developed that has the above properties and that can enable packageless processor systems.

2.4 Silicon Interconnect Fabric: An Enabling Technology for Packageless Processing

We have developed a novel system integration technology, *Silicon Interconnect Fabric (Si-IF)*, that realizes large scale die to wafer bonding technology with very fine pitch interconnection and reduced inter die spacing. The key idea behind Si-IF is to replace the organic PCB board with a silicon substrate. Essentially, we place and bond bare silicon dies directly on to a thick silicon wafer using copper pillar based I/O pins. Processor dies, memory dies, non-compute dies such as peripherals, VRM, and even passive elements such as inductors and capacitors can be bonded directly to the Si-IF. This allows us to completely get rid of the packages. A schematic cross-section of a processor die on Si-IF is shown in Figure 2.4.

Wafer-scale system manufacturing for building large high-performance computers had been proposed as far back as the 1980s [36], but yield issues doomed those projects, which attempted to make large wafer-scale monolithic chips. Here, the approach is to make small dies with good yield and connect then on a wafer with simple and mature fabrication technology.

Although at a first glance Si-IF technology seems similar to interposers, it is fundamentally different. Interposers use through-silicon-vias (TSV) and because of aspect ratio limitation of TSVs, the interposer needs to be thinned and thus it becomes fragile and size limited. In fact, interposers are typically limited to the maximum mask field size (e.g., $\sim$830 mm^2 which is the same as maximum SoC size) to avoid stitching. Though larger interposers can be built using stitching, they are much costlier and have lower yield. Also, interposers need packages for mechanical support and for space transformation to accommodate larger I/O connections to the PCB. Therefore, connections with chips outside of the interposers continues to suffer from the issues of conventional packaging. On the other hand, Si-IF is a standalone rigid interconnect substrate capable of scaling up to a full size of a wafer and doesn't require packages for mechanical support.

Next, we discuss the distinguishing characteristics of the Si-IF technology in more detail:

Fine pitch inter-die interconnect with 2 - 10 μm pitch: Solder extrusion and surface warpage limit the minimum I/O bump pitch on PCBs. Rigid (polish-able) silicon wafer and copper pillar based IOs (bonded using thermal-compression bonding (TCB) at tight pitches) in Si-IF address both these limitations.[1]

Since the interconnect wires on Si-IF are manufactured using standard back-end process, the wire pitch can scale like normal top-level metal in SoCs and well below 2 μm [4, 37]. This technology thus bridges the gap between the SoC level interconnects and system-level interconnects and allows a processor die to support the required number of I/O and power pins even without a package.

Small inter-die spacing: Using state-of-the-art pick and place tools, bare die can be placed and bonded on to the Si-IF at very close proximity ($<$100μm) [4]. Thus, interconnects between the dies can now be orders of magnitude smaller than the case where the dies are placed inside separate packages. Coupled with fine pitch interconnects, SerDes links can now be replaced with parallel

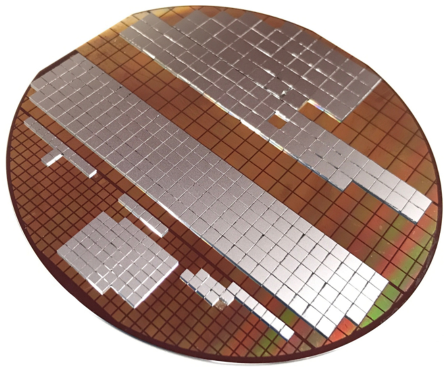
[1]The copper pillar TCB process involves using a bond interface temperature of $\sim$250-260°C for 3 seconds. Eutectic solder bonding is also done at 220-270°C for roughly the same period. Therefore, Si-IF-based integration is not expected to cause any temperature related aging of the chip.

interfaces and shorter links, thus resulting in lower latency as well as lower energy per bit. The link latency and bandwidth improvement from near placement of the dies coupled with increased I/O density from the fine pitch enables high bandwidth energy efficient communication even without a package.

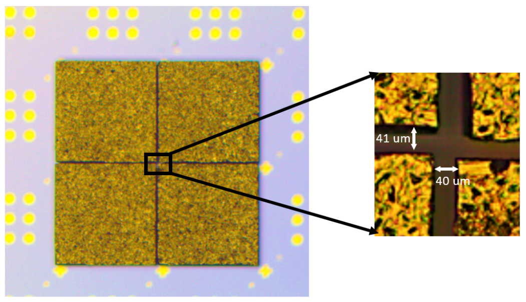
Efficient heat dissipation: Unlike PCB and package materials, silicon is a good conductor of heat. Heat sinks can be mounted on both sides of an Si-IF. Figure 2.4 shows how the overall thermal resistance of the Si-IF based system is smaller than that of a canonical packaged and PCB based system. The secondary heat sink attached to the back-side of the Si-IF has the added advantage of acting as a protective shield for the silicon substrate. In fact, the heat sinks would provide mechanical support and protection to the Si-IF instead of a conventional package. To summarize, Si-IF allows much more effective heat dissipation on packageless processors than a conventional packaged processor (more details in Section 2.5.3).

Lowered CTE mismatch: Since both the processor die and the Si-IF are silicon-based, thermal stresses are minimal. As such, the mechanical reliability issues such as bump/ball failures that arise in the conventional setting due to the CTE mismatch between the processor die and the package as well as the package and the PCB are eliminated. Unlike silicon interposers which need to be thin to support TSVs and therefore fragile and size limited [38, 39], Si-IF is thick, rigid and does not use through silicon vias. Therefore, Si-IF-based integration enables large scale processor assembly without requiring the mechanical support traditionally provided by the package.

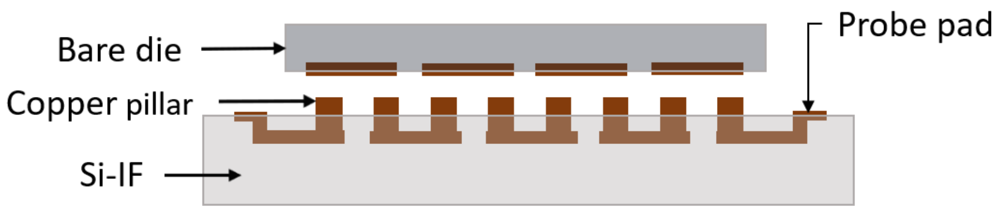
The above factors coupled with advancements in low-cost silicon processing [19, 40, 41], provide a viable pathway to realizing packageless processors. To demonstrate the feasibility of Si-IF technology for enabling packageless processors, we have built an Si-IF prototype which supports reliable fine-pitch interconnect, high I/O pin density, and close proximity inter-die spacing. Figure 2.5a shows a 4-inch wafer partially populated using 350 different dies of sizes 4 mm^2, 9 mm^2, 16 mm^2 and 25 mm^2. A micro-graph of four dies on a wafer spaced apart by only $\sim$40 μm is also shown in Figure 2.5b. Each of these dies on the wafer has copper pillar pitch of 10μm and interconnect wires of line-width of 3 μm. This enables high I/O density even without a package. To perform yield analysis of the copper pillars, we built in rows of serpentine test structures in every die as shown in Figure 2.5c. In each row, pillars n and $n+1$ were connected on the die, while pillars



(a)



(b)



(c)

Figure 2.5: (a) Wafer scale interconnect fabric partially populated with eighty 4 mm^2, one hundred and seventy one 9 mm^2, fifty eight 16 mm^2 and forty one 25 mm^2 dies bonded on a 4-inch silicon wafer. Copper pillar pitch of 10 μm is used. (b) Micrograph showing four dies bonded on to an Si-IF with $\sim$40 μm inter-die spacing. (c) Serpentine test structure with copper pillars on the Si-IF and landing bond pad on the bare dies [4].

$n+1$ and $n+2$ were connected using the Si-IF interconnect. Once the die was bonded to the Si-IF, the entire row were connected resembling a serpentine structure. End-points of the serpentines were electrically tested for continuity along a row of the pillars. Out of the 72000 pillar contacts tested, only 3 contact failures were observed. Thus >99.9% yield of the copper pillar connections is observed. This demonstrates the reliability of Si-IF as an enabling technology for packageless processors.

The specific contact resistances were measured to be within 0.7-0.9 $\Omega - \mu m^2$ [4] which is smaller than that of the solder balls (40 $\Omega - \mu m^2$) [42–44]. This is not surprising considering that copper has much higher conductivity compared to solder balls ($\sim 5e7$ $\mho/m$ vs 9.6e6 $\mho/m$). Therefore, the contact resistance of a 5 μm copper pillar is about 42 mΩ which is similar to contact resistance of 23 μm C4 solder bumps [42]. Also since, inter-die spacing can now be $\sim$100 μm, instead of the minimum spacing of 1 cm for package-based connections, trace resistance of Si-IF is expected to be much smaller in spite of thinner wires (e.g., assuming similar copper trench depth in PCBs and Si-IF, a 100 μm Si-IF trace will have 8 times lower resistance than the 25 μm, 1 cm length PCB trace). Similarly, relative permittivity of SiO_2 is 3.9, while that of FR4 material is 4.5. Comparing a PCB trace of width and spacing of 25 μm each and length of 1cm with Si-IF trace of width and spacing 2 μm and length 100 μm, the capacitance of the Si-IF trace is about 2 orders of magnitude smaller than that of PCB trace. Thus, RC delay would also be smaller. Using detailed multi-physics and SPICE simulations, we verified that the links can be switched at 2-4 GHz, while consuming <0.3 pJ/bit using very simple I/O drivers [37].

Moreover, the shear bond strength of the Cu pillars was measured to be greater than 78.4 MPa [4], while that of the BGA balls is about 40 MPa [44, 45] which confirms the superior mechanical strength of the copper pillars. Also, due to CTE mismatch of the different components of a package, the solder based bumps go through continuous temperature cycling and often suffer from fatigue related cracking, which would not be a case for Si-IF as the CTE mismatch is negligible.

More details on Si-IF manufacturing (e.g., patterning, die alignment, bonding, etc.) and characterization can be found in [4] and [37].

2.5 Quantifying Memory Bandwidth, TDP, and Area Benefits

In this section, we consider a baseline many-core processor architecture and evaluate the impact on memory bandwidth, TDP, and area if the processor's package is removed and the processor die is integrated using silicon interconnect fabric.

2.5.1 Baseline Processor

Our baseline processor is a 36-tile many-core architecture (with 22 peripheral tiles). Each tile consists of 2 cores and a shared L2 cache of size 1MB. The cores are out-of-order (OOO) with 64 KB private L1 cache. All the 72 cores share a total of 256 MB eDRAM based last-level cache (LLC). The LLC is organized as 8 slices of 32 MB 16-way cache each. Other micro-architectural parameters of the baseline processor are shown in Table 2.3. We assume that the processor is a standalone packaged processor as shown in Figure 2.6, where the DDR-based main memory is off-package. We use 8 memory channels for off-package DRAM with effective bandwidth of 9.6 GBps per channel. Thus an aggregate of 76.8 GBps of main memory bandwidth is available. The area of the processor die implemented in 22nm technology node is 608 mm^2 and estimated minimum size of the package required is 2907 mm^2. Details regarding the methodology to evaluate area, power, and performance are described in Section 2.6. To estimate the area of the package, we use the model described in Section 2.5.4. Next, we quantify the bandwidth, TDP and area benefits from a packageless implementation of this processor.

2.5.2 Memory Bandwidth

As discussed earlier in Section 2.2, packaged processors are I/O pin limited because of the pitch of the solder balls used to connect to the processor. Similarly, memory modules are I/O pin limited because of large pins used in vertically slotted DIMMs. Coupled with the fact that processor-memory connection uses wide PCB wire traces ($\sim$100 μm), DDR based communication bandwidth is usually capped at $\sim$10-15 GBps per channel. Limited interconnect wiring and pin density also constrains the maximum number of memory channels. Though higher bandwidth can be achieved using complex

Table 2.3: Configuration of the many-core baseline processor

Cores	36 Tiles, each having 2 Silvermont-like OOO at 1.1 GHz, 1 hardware thread, dual issue
Caches	64 KB L1 (Private), 1 MB L2 (Private per Tile), 256 MB eDRAM L3 (Shared)
Memory	DDR4-1600 MHz, Double-pumped at 9.6 GBps, 2D mesh interconnect
Cache Coherence	Directory-Based MESIF
Prefetching	L2, L3 prefetch-on-hit, 4 simultaneous prefetches

SerDes techniques, they are energy inefficient (~10x) and lead to additional latency [46–48]. Some high-end processors [31, 38, 49, 50] use 2.5D technologies such as interposer [51–53], EMIB [54]. etc. to integrate high bandwidth in-package DRAM memory that can achieve up to 450 GBps of bandwidth. However, the number of memory dies that can be accommodated inside a package is limited due to low yield and high manufacturing cost of larger interposers, EMIBs, etc [54, 55]. Typically, interposers are limited to maximum mask field size (~830mm^2 which is the same as maximum SoC size) to avoid stitching. The largest commercially available interposer is ~1200 mm^2 (uses stitching) which only accommodates a processor die with four 3D memory stacks [56]. As a result, the majority of the main memory that is usually placed off-package continues to suffer from limited memory bandwidth.

Since memory chips are connected to the processor chip directly (i.e., without a package and PCB traces) in the Si-IF setting as long as they can fit in the size of a silicon wafer (Table 2.4), the corresponding supportable memory bandwidth is much higher. As one estimate, the interconnect traces on Si-IF are 2-10 μm in pitch (Section 2.4) as opposed to ~100 μm on PCB, which means about 10-50x more bandwidth is available on Si-IF than on PCB. Moreover, since the link length is expected to be small in Si-IF, signalling can be done at relatively higher frequencies of 4-5 GHz with simple transceivers. The estimated bandwidth per mm edge of a die is ~50 GBps and ~250 GBps for 10 μm and 2 μm interconnect pitch respectively.

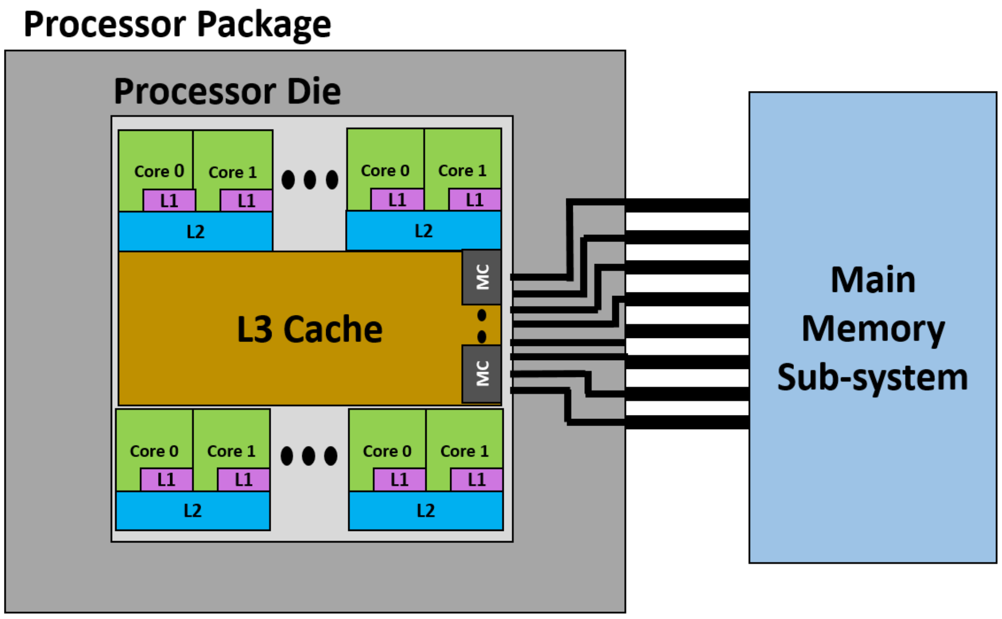


Figure 2.6: Base Processor Architecture Overview.

2.5.3 TDP

Thermal characteristics of a processor system drive many design decisions such as maximum operating frequency, peak power, etc. Since packageless processors allow more effective heat extraction (see Section 2.4), the allowable TDP for the same junction temperature constraint increases. To compare the thermal characteristics in PCB based packaged systems against Si-IF based packageless systems, we use the thermal resistance model shown in Figures 2.3 and 2.4. Simulations to estimate the thermal resistance of the heat sinks taking into account the air flow, heat spreading effects and size of the heat sink were performed using a commercial thermal modelling software ‘R-Tools’ [8]. We compare different design points such as a conventional package on large PCB vs small PCB, an interposer package on large PCB, a die mounted on large Si-IF vs small Si-IF, and a PCB replaced with Si-IF without removing the package. TDP for the baseline packaged processor is calculated as 0.75 times the processor peak power [2, 57]. We assume a heat sink of the size of processor package, ambient temperature of 25 °C, and forced airflow convection to calculate the junction temperature to be 64.2 °C in this case. We then calculate for each design point the maximum allowable TDP that produces a junction temperature no higher than 64.2 °C. Figure 2.7 shows the results.

Table 2.4: Comparison of Si-IF vs other 2.5D technologies

	Silicon Interposer [51]	**EMIB [54]**	**Si-IF**
I/O Pitch (μm)	30-100	20-40	2-10
Interconnect Wire Pitch (μm)	2-10	1-10	1-10
Maximum Size/Dies	8.5 cm^2	5-10 Dies	Up to a Full Wafer
Inter-Die Spacing (mm)	>1	>1	<0.1
System Integration Scheme	Package on PCB	Package on PCB	Bare Die on Wafer
Other Factors	Complex Assembly Process and TSV Capacitance issue	Complex Manufacturing of Organic Substrate	Bonding Passives and Legacy I/O Ports

Results show that the TDP benefit from just removing the large PCB and replacing it with an Si-IF is about 6%. Removing the package in case of Si-IF gives an additional $\sim$15% benefit. The surface area of the Si-IF also affects the amount of heat dissipation, which can increase allowable TDP by about 5-7%. In a packageless system with one heat sink and a large Si-IF, maximum TDP that can be allowed for the same junction temperature is 181 W which is 21.5% higher than the baseline case. The benefit increases to 70% (TDP of 254W) when heat sinks are installed on both sides.[2] Meanwhile, interposer-based 2.5D integration shows no benefit in terms of TDP. In fact, the use of additional interposer layer inside the package lowers the allowable TDP by a small amount due to increased thermal resistivity on the downward heat flow path. The TDP benefits of packageless processing will only increase with increasing die area since the more effective heat spreading on

[2]We expect the cost of placing a single heat sink to be comparable to the packaging cost of baseline system. Since we do not have a package, the second heat sink can be added without increasing cost over the baseline system, while providing significant TDP (and, therefore, performance) benefits in return.

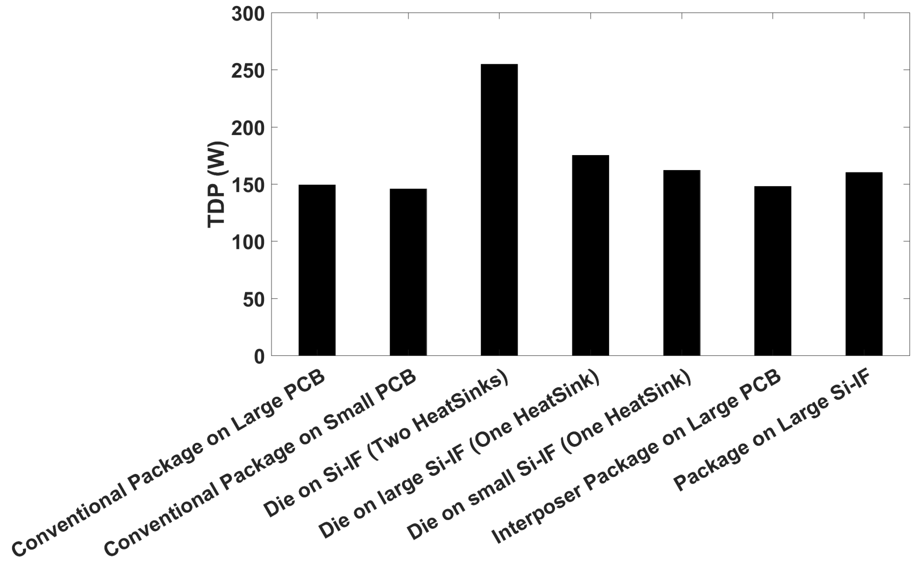


Figure 2.7: Maximum achievable TDP of the baseline processor system in various integration schemes.

larger dies makes the package resistance a bigger fraction of the overall thermal resistance for packaged systems.

2.5.4 Area

Due to the high package area to die area ratio (Table 2.1), removing the package can lead to significant area benefits. To quantify the area benefits for the baseline processor, we use the following model to estimate the minimum size of the package given the peak power of a processor, number of signal IOs (SPins), and type of I/O.

$$Area_{package} = bump\_pitch^2 \times (\frac{Peak\_Power}{Power_{perPin}} + \#SPins) + Area_{Non-I/O} \tag{2.1}$$

Non-I/O area is determined by other factors such as RDL layer and PCB routing constraints. We assume the maximum current per power/ground pin to be 250 mA [58, 59] and bump_pitch to be 900 μm [19, 22]. Using this model for the baseline processor, the minimum package area (when non-I/O area is not considered) is estimated at 2907 mm^2 which is about 5x larger than the processor die area (608mm^2). The area benefit from removing the package will be higher for processors with higher power density (Figure 2.8) since packages required such processors need to be larger so as to

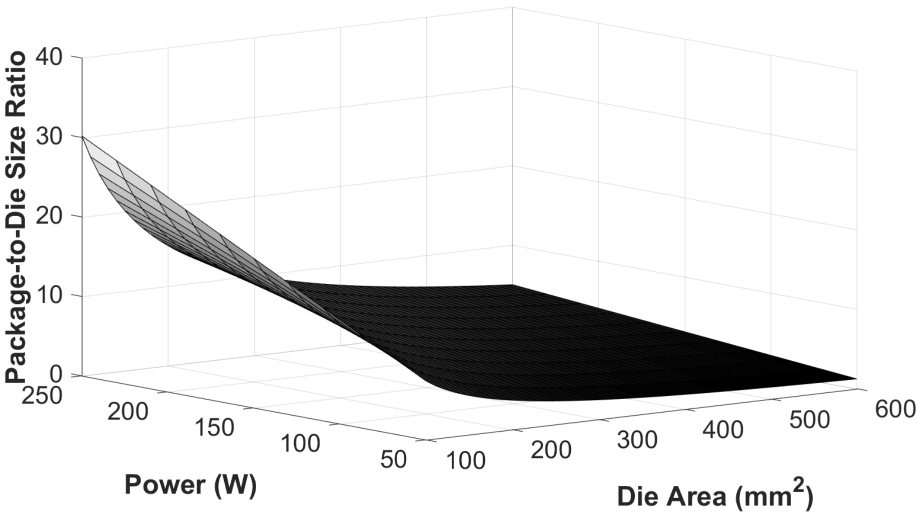


Figure 2.8: Sensitivity analysis of area benefit from removing the package.

accommodate the power pins and also to dissipate the heat efficiently.

2.6 Methodology

In this section, we describe our methodology for estimating the performance benefits from packageless processing for our baseline processor (Table 2.3).

First, we use McPat 1.0 [60] to determine the area and TDP for the baseline processor. Next, we calculate the additional bandwidth, TDP, and area available from packageless processing (Section 2.5). We then determine a processor design that exploits the additional bandwidth, TDP, and area to improve performance. Since the bandwidth benefits from packageless processing are substantial (orders of magnitude), to eliminate bandwidth slack, we increase the number of memory channels to one per peripheral tile in the baseline processor. To exploit higher allowable TDP, we consider two approaches - increasing core frequency and increasing the number of tiles in the processor, including adding an additional slice to the eDRAM L3 cache for every 4 additional tiles. Yield concerns limit the number of tiles that can fit in a single die. Therefore, we consider a multi-chip processor system where we limit the size of each die to at most 600mm^2. Each die contains an even portion of the tiles and is connected in a 2D mesh with the other dies via an inter-processor communication protocol. We use latency of 20 ns [37, 61] and bandwidth of 1 TBps [37] to model the inter-processor communication on Si-IF. We use the same technique when considering area slack.

Once we have determined a set of processor designs, we use a fast multi-core interval simulator, Sniper [62], to determine relative performance. We simulate six benchmarks from the NAS Parallel Benchmark (NPB-3.3) suite [63] and six benchmarks from PARSEC 2.1 [64]. Among the NPB benchmarks, we chose BT (Block Tri-diagonal solver) and SP (Scalar Penta-diagonal solver) as sample pseudo applications, CG (Conjugate Gradient) and UA (Unstructured Adaptive mesh) as having irregular memory access patterns, MG (Multi-Grid) as being memory intensive and EP (Embarrassingly Parallel) as being highly scalable. We used dataset size C, which has an estimated memory requirement of 800 MB. Among PARSEC benchmarks, we chose blackscholes and fluidanimate as sample data-parallel applications, canneal and dedup as having high rates data sharing, and streamcluster and freqmine as typical datamining applications. For all evaluations, the simulation was fast-forwarded to the Region-of-Interest (ROI), simulated in cache-only mode for 1 billion instructions and then simulated in detailed mode for 1 billion further instructions.

2.7 Results

In this section, we demonstrate that packageless processors offer significant performance benefits over their packageless counterparts.

2.7.1 Exploiting Higher Available Memory Bandwidth

Since Si-IF provides at least 10x more bandwidth than the PCB case alongside plentiful of I/O pins, several techniques such as using wide-I/O interface for the whole memory system and increasing the number of memory channels can be implemented. Though wide-I/O implementation is feasible in interposer-based assemblies as well, number of memory channels is limited since only a few memory devices can be placed on the interposer (due to maximum size / yield limitation - Section 2.5) as opposed to the Si-IF case, where many more memory devices can be accommodated (limited only by the size of the silicon wafer).

Our baseline processor contains 22 peripheral tiles, so we used a maximum of 22 memory channels for the packageless case in our evaluations. Figure 2.9 shows the potential improvement in performance from having one memory channel per two peripheral tiles (107.8GBps) and one memory channel per peripheral tile (215.6 GBps) over the eight memory channels in our baseline processor configuration (78.4 GBps). We also compared the performance of all three of these configurations against the maximum achievable performance for a 10 TBps memory bandwidth along the peripheral tiles – this bandwidth is achievable on Si-IF using HMC like memory which supports up to 480 GBps per device [65]. We denote this as the *infinite bandwidth* case.

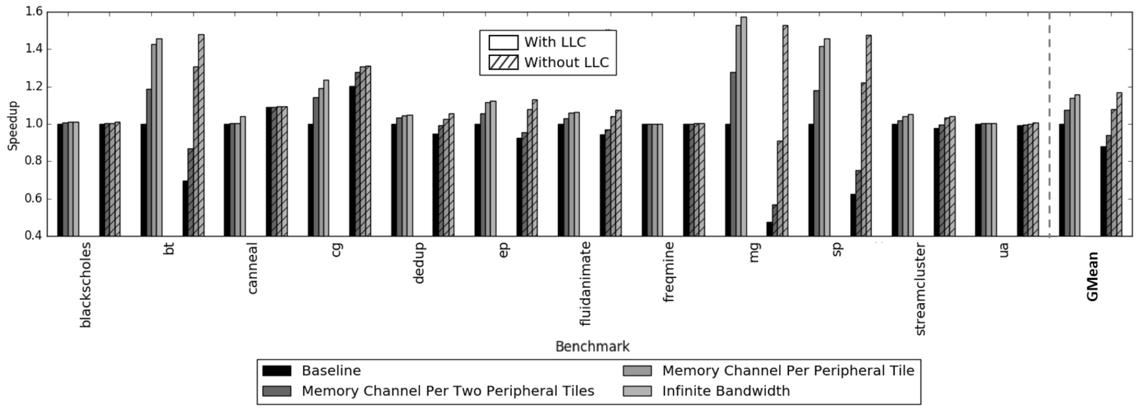


Figure 2.9: Performance benefit from increased number of memory channels with and without L3.

Increasing number of channels results in average improvement of about 15% with a large L3, while it has a much greater effect (23%, on average) in the absence of an L3. For applications such as BT, MG and SP, the improvement in performance is >42% both with 22 memory controllers as well as infinite bandwidth when L3 is present. Even without the L3, the performances of BT and SP in the 22 memory controller-case are 31% and 22 % higher respectively than the baseline case with L3. In fact, with 22 channels, but without L3, the average performance across all benchmarks is 8% higher than baseline case with L3. This is because the memory bandwidth effectively improves enough to eliminate the need for an L3. A less intuitive result is that for benchmarks such as CG and Canneal, removing an L3 results in *higher* performance. This is due to limited sharing and irregular memory access patterns in these benchmarks; an L3 increases memory latency unnecessarily in the case that data is used by one core and never shared.

Area overhead of the additional memory controllers would result in increased area of the processor chip. We estimated the area overhead per memory controller in 22nm technology to be about 1.8 mm^2 [66]. This implies that the new processor chip size (with additional memory controllers) can exceed 650 mm^2 which would worsen the die yield. This issue can be tackled using two ways. First, since Si-IF provides similar density and performance as that of global interconnects in SOCs, we can now have separate memory controller dies which contain clusters of memory controllers. The alternative approach would be to reduce the size of the LLC to accommodate the additional memory controllers. Since performance without LLC, but with additional bandwidth, is similar or higher than the baseline case with LLC in our evaluations, overall performance is expected to improve.

In summary, larger number of memory channels in packageless processors improves performance by up to 58% (average 16%) and 53% (average 14%) in case of infinite bandwidth and 22 memory channels respectively, and allows elimination of the LLC with in fact 8% higher performance with 22 memory channels than the baseline case with LLC.

2.7.2 Exploiting Higher Available TDP Budget

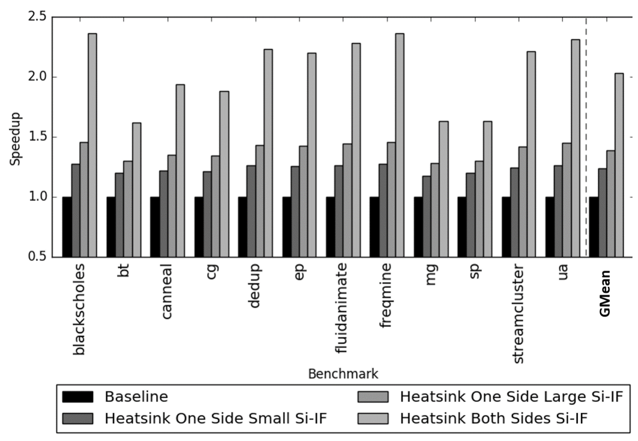
As mentioned in Section 2.5.3, additional power can now be sustained without increasing the core junction temperature. Thus, we can either add more cores to the system or increase the frequency of operation (Table 2.5). In Figure 2.10, we show the performance improvement of these two different design choices across different benchmarks. Frequency scaling alone provides consistent gains of $>15\%$ in performance across all benchmarks when only one heat sink is used. The performance boost is $>50\%$ when both the heat sinks are used. Using DVFS could result in substantially higher speedups, as strategically increasing the frequency of only certain cores would take more precise advantage of the increased TDP. Furthermore, increasing the number of tiles has potential for greater speedup for certain applications. For example, EP achieves more than 2.5x improvement in performance versus 2.2x when frequency is scaled. However, increasing the number of cores requires substantially more area - the largest processor in this experiment exceeded 1600mm^2 total area (recall that we use an multi-chip processor configuration for large area cases - each chip is still only 600mm^2 big). Additionally, some applications do not have enough exploitable TLP to fully take advantage of the increased number of cores. Freqmine and streamcluster are two examples of benchmarks which achieve substantial gains by scaling frequency, but do not gain performance from adding further tiles.

Table 2.5: Increasing Frequency or Number of Tiles to Exploit Available TDP Slack

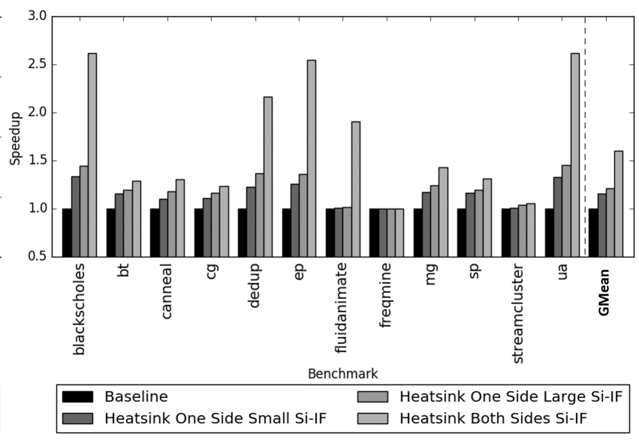
System Configuration	TDP	Max Frequency	Max # Tiles
Baseline	149 W	1.1 GHz	36 Tiles
Small Si-IF Heatsink 1-Side	168 W	1.4 GHz	48 Tiles
Large Si-IF Heatsink 1-Side	180 W	1.6 GHz	52 Tiles
Large Si-IF Heatsink 2-Side	250 W	2.6 GHz	96 Tiles

One more efficient way to take advantage of thermal slack would be to perform a two dimensional design space exploration on the chip, scaling both frequency and number of tiles until an optimal system is found. In general, frequency has a clearer and more well-defined trade-off between power and performance. In addition, through DVFS it is easier to manipulate frequency during runtime and be able to optimize the processor for a specific application. While one could dynamically change the effective number of tiles available in a processor via power gating, there is a much higher overhead for such a transition, including wakeup time, cache warmup and various OS overheads associated with context switching. However, due to bandwidth constraints and the benefits of having a larger total cache area, increasing the number of tiles provides for running massively parallel workloads much more efficiently than a smaller number of highly clocked processor.

Increasing frequency or the number of tiles would increase the power demand. Besides thermal constraints, increased power consumption also requires careful management of power distribution losses (for example by point of use step down voltage conversion just like conventional packaged systems). Packageless Si-IF with no C4 bumps or wide PCB traces can substantially help with inductive voltage drops. High power requirements also come with larger demand for power/ ground I/O pins which can be accommodated within the die area using fine pitch interconnect pillars on Si-IF.



(a) Frequency



(b) Number of Tiles

Figure 2.10: Performance benefits of utilizing TDP slack.

In summary, removing the package improves the TDP budget by up to 70% which can provide upto 136% higher performance (average 103%) upon increasing the operating frequency and up to 162% (average 60%) upon increasing the number of tiles using our benchmarks.

2.7.3 Exploiting Higher Available Area

Table 2.6: Area Slack Exploitation Parameters

System Configuration	Max Area	Processor Microarchitecture
Baseline	608mm^2	36 Tiles, Single Die
Packageless Half-Slack	1758mm^2	96 Tiles, Four Dies
Packageless No-Slack	2908mm^2	144 Tiles, Six Dies

Figure 2.11 shows performance benefits for eliminating half and all of the area slack available in a packageless processor. For our evaluations, dies are restricted to 600mm^2 - see Section 2.6 for details. One might not want to fully exploit available area for many reasons: higher power and lower yield being among the chief concerns. Much like the case of tile-based power slack elimination, some applications benefit drastically more than others from area slack reduction. For applications such as fluidanimate, nearly all of the performance benefits, i.e., ∼86% over the baseline case, are achieved via half-slack reduction, while other applications can continue to take advantage of any

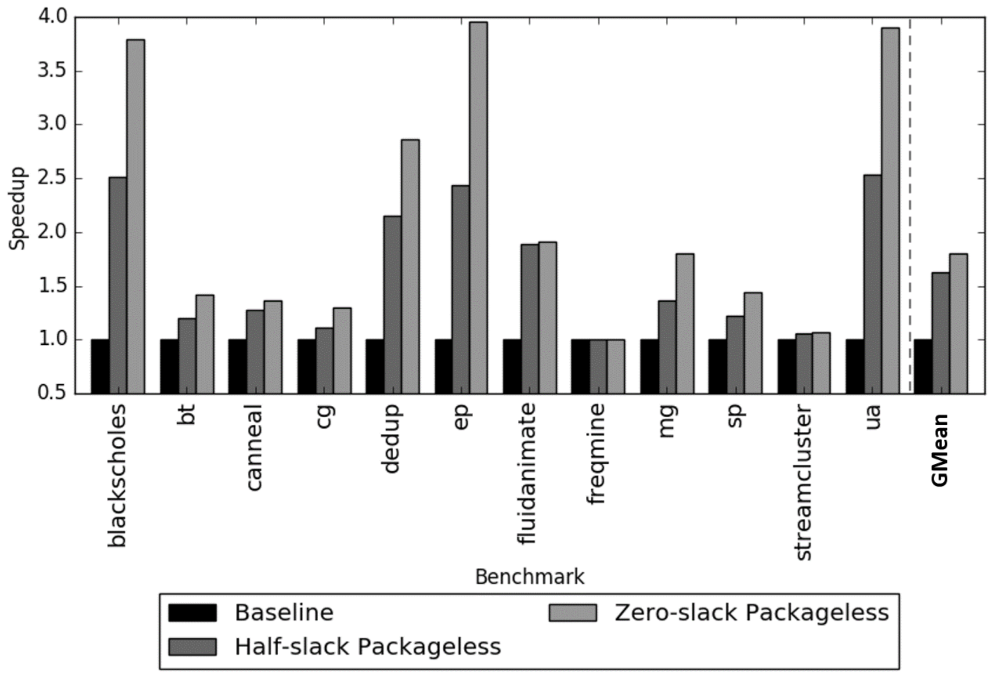


Figure 2.11: Performance increase by exploiting area slack.

extra cores available. As in Section 2.7.2, benchmarks such as blackscholes, EP and UA increase performance proportionally to number of tiles, due to high thread level parallelism (TLP). For applications which lack such easily exploitable TLP, having a large number of cores may still be useful in the case of multi-programming. The power overhead of such a large design can be mitigated using per-core DVFS or power-gating. The removal of a package allows for systems with much denser compute: for compute-intensive high performance systems which require thousands of cores, packageless processors could prove to be a critical technology.

In summary, across the benchmarks evaluated, packageless processors could achieve 80% average performance improvement, up to 295% by utilizing the extra area slack coming from removing the processor package.

Note that we are allowing the original TDP budget to be breached in these experiments; we assume that a costlier cooling solution exists to tackle the increased thermal dissipation if the intention is to use the entire area slack. Section 2.7.4 considers this tradeoff between area slack and TDP slack.

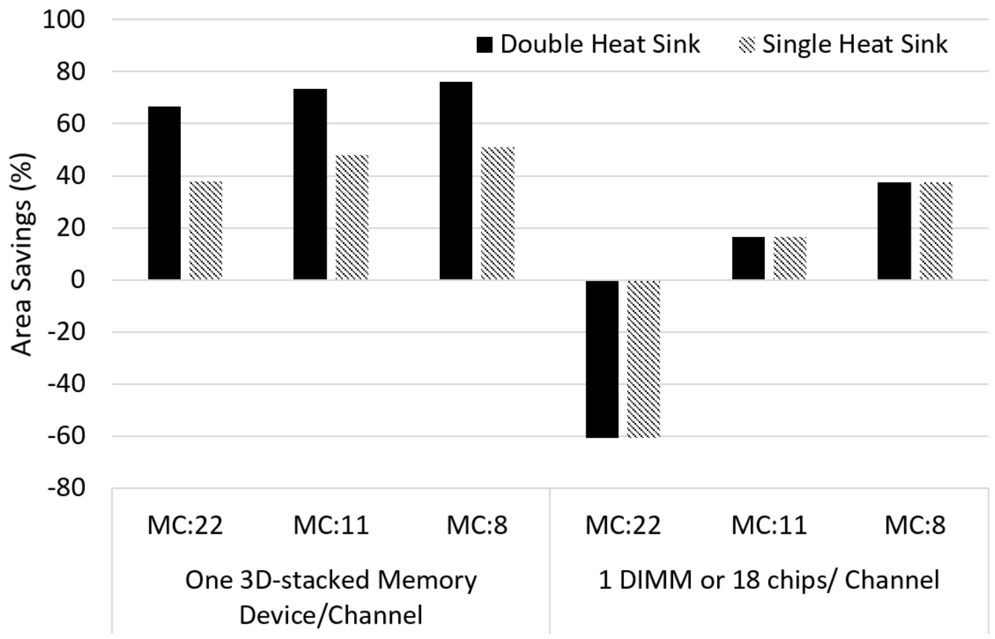


Figure 2.12: Area savings when implementing processor-memory subsystem on Si-IF. 22, 11 and 8 memory controller configuration on Si-IF are compared against baseline packaged configuration with 8 channels of off-package DRAM.

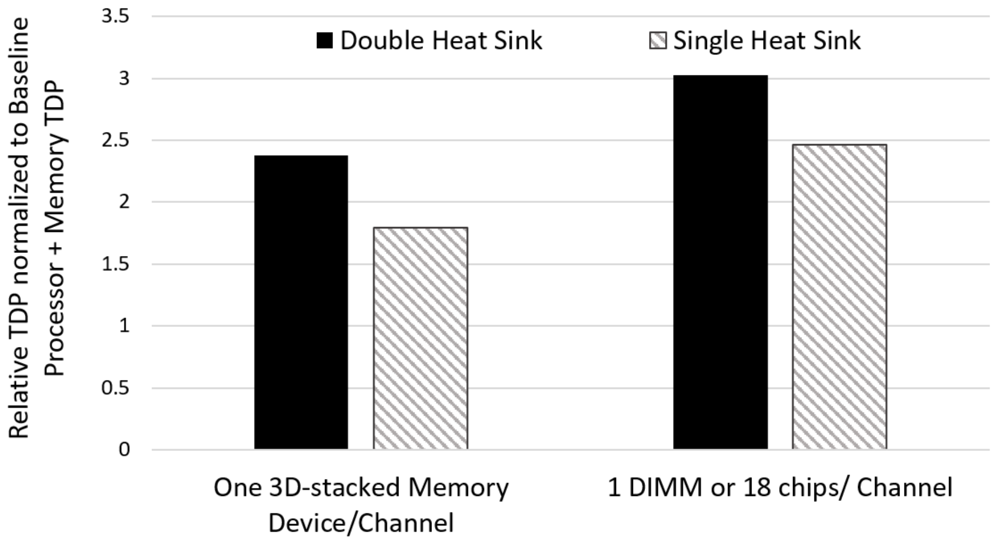


Figure 2.13: TDP of implementing the baseline processor-memory subsystem on Si-IF of size of total processor and memory package area normalized to baseline processor system TDP.

2.7.4 Area-TDP Tradeoff

Thus far, we have quantified the bandwidth, TDP and area benefits individually that packageless processors provide over conventional systems. In this section, we ask the question - how much improvement in form factor, TDP and bandwidth can be achieved when the factors are considered simultaneously?

For our evaluations, we consider two PCB-based baselines - one DIMM (with 18 chips) per channel and one 3D stacked memory device per channel. The corresponding Si-IF design points have 18 packageless DRAM chips per channel laid out in a planar configuration (Figure 2.14b) and one packageless 3D stacked memory device per channel respectively. The processor footprint is 2907 mm^2 (Section 2.5.4) in the packaged case, while it is 608 mm^2,on Si-IF. For estimating memory footprint in the PCB case, we assume that the DIMMs are slotted vertically onto the PCB (Figure 2.14a).The PCB footprint for each DIMM is estimated to be 7.92 cm^2 (we used the DIMM socket size as the footprint estimate to perform a worst case comparison of area benefits from Si-IF, ignoring large inter-socket distances typically used on a PCB). The PCB footprint for each 3D stacked memory package is considered to be 320 mm^2 [67]. For estimating the memory subsystem footprint in two packageless cases, we considered 36 mm^2 per DRAM die and 55 mm^2 per 3D stacked memory device [68].

(1) ***Form Factor Reduction in iso-TDP case:*** In the iso-TDP case, we compare the footprint of the baseline 8 memory channel configuration on PCB against the same system implemented on Si-IF. We also extend the analysis for both the 22 channel (one channel per peripheral tile) and 11 channel (one channel per two peripheral tiles) memory configurations on Si-IF and compare it against the same baseline PCB case.

We adjust the size of the heat sink so as to achieve the same maximum junction temperature as the baseline junction temperature of 64.2 °C (junction temperature of the in-package baseline processor die). In case the heat sink required to achieve the desired junction temperature is larger than the total processor and memory footprint, the area of the heat sink determines the compute footprint of Si-IF system.

Figure 2.12 shows the area savings in different scenarios. For one memory device per channel case, the dual heat sink setup leads to area savings of up to 76 % and using one heat sink provides $>36\%$ area reduction. This is because in the dual heat sink setup, the thermal resistance is lower, meaning smaller heat sinks can help achieve higher TDP.

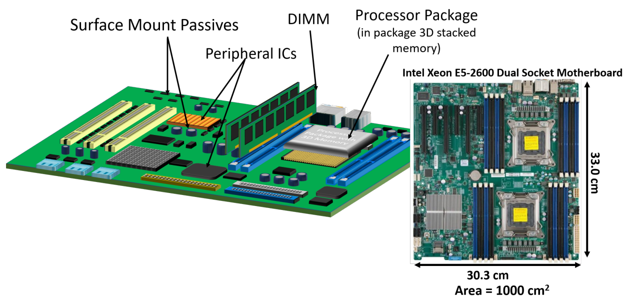
For DDRx style memory configuration with 18 dies per DIMM, when the baseline 8 memory channel configuration is laid out on the Si-IF with memory bandwidth similar to PCB case, 37% area savings can be achieved with similar performance as of the baseline packaged case. The area saving reduces to 17% in the 11 memory channel case while the performance increases by 7.5%. For 22 memory channel case (memory channel per peripheral tile with LLC in Figure 2.9), where $22 \times 18 = 396$ memory dies need to be accommodated on planar Si-IF, the footprint does increase but there is plenty of TDP slack left unused as the large heat sink "overcools" the system.

In summary, packageless processing under a TDP constraint with emerging 3D stacked memories can deliver dramatic footprint reductions (40%-76%) while increasing available memory bandwidth. In conventional DDR-style memory systems, going packageless can deliver 36% footprint reduction with same performance.

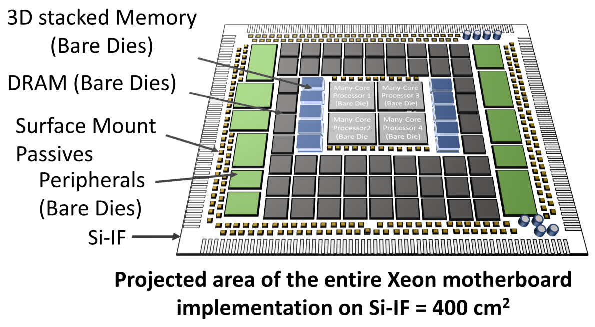
(2) ***Increased TDP Slack in iso-Area case:*** Here, we compare the TDP slack available for the packageless processor system if the total area of the Si-IF and the heat sink are equal to the total PCB footprint of the processor and the memory subsystem. Figure 2.13 shows the total packageless TDP available as compared to the total TDP of the baseline processor and memory subsystem. Since the area footprint of the DIMM is much larger than that of the 3D stacked memory packages, the equivalent iso-area Si-IF/heat sink size is larger which leads to extra TDP slack. This excess TDP slack alongside the excess area under the heatsink can be utilized by increasing the number of tiles, frequency of operation, memory capacity etc. *In summary, packageless processing with the same computing footprint can deliver 1.7X-3X extra power to burn to improve performance without violating thermal constraints.*

2.8 Discussion

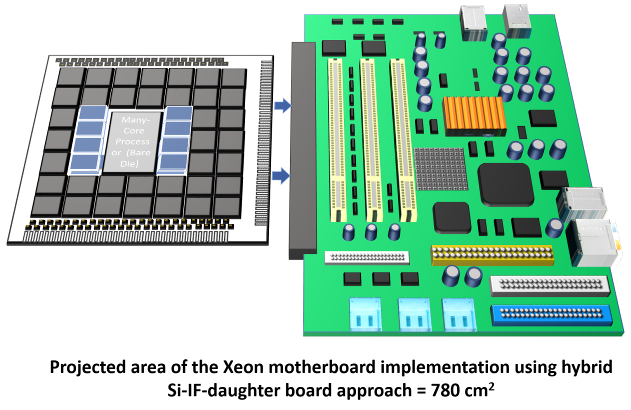
In this section, we discuss the implications of packageless processing on how the overall system could be realized, and other aspects such as repairability, testability, manufacturability, and cost.



(a) Conventional system integration on PCB


3D stacked Memory (Bare Dies)
DRAM (Bare Dies)
Surface Mount Passives
Peripherals (Bare Dies)
Si-IF
Projected area of the entire Xeon motherboard implementation on Si-IF = 400 cm²

(b) Full system integration on Si-IF


Many-Core Processor (Bare Die)
Projected area of the Xeon motherboard implementation using hybrid Si-IF-daughter board approach = 780 cm²

(c) Compute modules (full processor-memory subsystem) on Si-IF alongside PCB-based daughter board for peripherals, passives and other components

Figure 2.14: Illustration of a conventional PCB based system and different integration schemes using Si-IF.

We also discuss some architectural implications not covered in this work.

2.8.1 Overall System Architecture

A full system implementation comprises of core compute elements such as CPUs, GPUs, memory, etc., and non-compute elements such as crystal oscillators, driver ICs for system IOs, components of power delivery network, etc. So far, we have only discussed the compute elements of the system, however architecture of non-compute components is important as well.

Traditionally, surface mount non-compute components are soldered directly on the PCB (Figure 2.14a). In Si-IF, we envision two alternatives to integrate these components into the system. One is to bond the passives and other non-compute components directly onto the silicon board using solder balls and large pads on Si-IF, as shown in Figure 2.14b. We have been able to achieve bonding of passives on to the Si-IF successfully. This enables a full system integration on Si-IF. The other alternative is a hybrid approach shown in Figure 2.14c, where the compute components alongside some Si-IF compatible non-compute components can be integrated on to the Si-IF, and other remaining non-compute components can be integrated on a separate daughter board. An ancillary benefit of a daughter board approach is that the daughter board can now also host some upgradeable and spare components such as extra spare DIMMs alongside legacy connectors.

We estimated the footprint of a $\sim$1000 cm^2 Intel Xeon dual socket motherboard [69] in Si-IF setting. Considering non-compute footprint reduction by 50% when non-compute is fully implemented on Si-IF (due to denser integration of all components on Si-IF) alongside packageless implementation of memory and processor dies, a full Si-IF implementation footprint can be $<$400 cm^2 while the hybrid approach can be $<$780 cm^2.[3]

[3]Link lengths are often lower in the packageless systems case since inter-die spacing can now be reduced to $\sim 100um$. We estimated that the farthest DDR links are about $\sim 2x$ longer on standard Xeon PCBs, when vertically slotted DIMMS are used, compared to when bare dies are placed in a planar fashion on Si-IF. So, DDR-type signalling and routing will not be an issue. In very large Si-IF systems where signal integrity may be an issue, we can use intermediate buffer dies/chiplets to buffer the signals if simpler signalling is used (for lower power).

2.8.2 Test, Reliability, and Serviceability

Bare dies are difficult to probe because of the small size of the I/O pads. However, significant progress has been made in bare die testing techniques, primarily driven by need for known good die in 2.5D and 3D IC technologies [70, 71]. Some examples include temporary packages [72, 73], wafer level burn in and test [74, 75], die-level test [76], and built-in self testing mechanisms [77, 78].

A packageless processor may be more susceptible to environmental agents (radiation, moisture, etc.) than its packaged counterpart. Layer of radiation hardening material (e.g., SiC/H, Boron-10, etc.) can be CVD deposited to protect against radiation . Also, in many cases, package itself is the source of radiation which in the packageless case is omitted. Similarly, the IF-assembly can be passivated with a CVD-based coating, which protects it from moisture and salt intrusion. Furthermore, we apply a hermetic sealant around the edges of the dies to prevent environment agents to get beneath the dies and corrode the copper pillars. External heatsink(s) provide additional environmental protection when used. Finally, the chip-to-wafer bonders have necessary (e.g. for ESD) protections to avoid any charge accumulation on the chip as well as the Si-IF.

While soldered or socketed components can be replaced in conventional PCB based integration schemes, replace, rework, or upgrading components is relatively difficult for Si-IF based systems since de-bonding metal-metal joints is a complex process which requires high temperature to melt the bond joint. As such, the benefits of Si-IF must be weighed against serviceability concerns (MCMs, MDPs, 3D integration, etc., also provide improve performance and energy efficiency at the expense of serviceability).

Serviceability of Si-IF-based systems can be improved through redundancy and self-repair. While these solutions incur additional costs, the considerable cost reduction from the proposed approach should defray these costs in many applications. Also, redundancy/self-repair costs can be reduced. For example, if a specific component is prone to frequent failure, it may be soldered or socketed, instead of TC-bonded, to improve serviceability (we already have a mix of solder/socket and copper pillar TCB on some of our prototypes). While this reduces I/O count, area and TDP benefits remain. Even the I/O costs can be minimized. As one example, DRAM chips have low I/O density and are prone to faults; they can use conventional solder bumping or placed in soldered sockets, while processor chiplets can be TC bonded (using copper pillars) to support large I/O count.

2.8.3 System Level I/O Connections and Mechanical Installation

External I/O connections would be made at the edge of the Si-IF to allow the rest of the surface to be covered using the heat sink. Conventional plug connectors or solder based connections can be used for signal and power delivery to the Si-IF. Silicon is much more robust than FR4 used to build PCBs (compressive strength of 3.2-3.4 GPa vs 370-400 MPa) and can easily handle the normal insertion force of a plug connector (few MPa to a few 10s of MPa), especially with backside support (e.g., backside heat sink) - our 700 μm-thick prototype kept flat on a chuck was intact even when a compressive stress of 1.5 GPa was applied over 0.13 mm^2. Even with minimal backside support, silicon is much more robust than the PCB (Ultimate Tensile Strength (UTS) of 165-180 MPa vs 70-75 MPa).

There are several options for installation. In case of server chassis, the complete system-on-wafer can be inserted using low force insertion sockets. Alternatively, in implementations with external metal heatsink(s), the heatsink(s) can be bolted to the chassis. If heatsinks are not required on both sides, backside heatsink will be preferred to provide support. The other side can be optionally covered using a robust material , e.g., metal plate. In case of cellphones, Si-IF can be held with mechanical jaws or can be fixed using a thermally conductive glue.

2.8.4 Manufacturing Challenges and Cost

The Si-IF integration required to enable effective packageless processing relies on metal-metal thermal compression bonding of copper. After the initial TCB process of 3 sec bonding at $\sim$250°C interface temperature, batches of bonded wafers undergo thermal annealing for about 6-8 min at $\sim$150°C to enhance bond strength and reduce tail probability of bond failures [4] - potentially decreasing the throughput of the manufacturing process. Maskless lithography is used to pattern large area, fine-pitch interconnect on Si-IF which can also have throughput concerns. Further improvements in large area patterning may be needed for volume production.

Removing the package has significant cost benefits since for many processors, packaging costs are often about 30-50% of the total processor cost [79,80]. Also, the significant area reduction from packageless processing should lower costs even further. As an example, the baseline 8 memory channel, 3D memory system will have area of 1048 mm^2 (608 + 8*55) in Si-IF and 5467 mm^2 (320*8+2907) on packaged PCB. A processed silicon wafer with a 90nm global layer back-end (enough to sustain 2 μm pitch) is roughly $500 per 300 mm wafer. Moreover, the die-to-Si-IF bonding is performed using industry standard die-to-substrate bond tools with small upgrades. Assembly cost per system is therefore expected to be around $15. For packaged systems, just the cost of packages is roughly $44 (3*8 + 20) per system [81]. Similarly, since wire pitches in Si-IF are several microns wide (2-10 μm), Si-IF fabrication is performed using older technology node (90nm/180nm) processes that support these wire pitches. As such, the fabrication cost is low. High performance multi-layer PCBs often cost a few hundred dollars while having much lower compute density than that of Si-IF. Finally, since Si-IF provides large form factor benefits, performance density per volume goes up. This has the potential to decrease the overall total cost of ownership [82].

2.8.5 Other Architectural Implications and Use-case Scenarios

In addition to the architectural techniques explored in this work to exploit the benefits of packageless processors, there exist several other micro-architectural optimizations that may be used in the context of packageless processors. For example, aggressive prefetching techniques [83] can leverage the availability of ultra high bandwidth. Similarly, architectures without L3 may be promising for applications where the reduction in L3 miss penalty can offset the effect of L3 miss rate. Also, TDP and area benefits can be utilized by introducing heterogeneous computing, such as GPUs, accelerators, DSP modules, etc. Moreover, since interconnect links are shorter in Si-IF, Ldi/dt noise would be smaller. Not only does this potentially reduce the number of decoupling capacitors required on the chip (or inside the package) thereby reducing chip area (or making it available for additional features), inductive noise driven constraints on frequency and timing of power gating, DVFS [84] etc can also now be relaxed. Finally, it may be possible to build wafer-scale systems using the Si-IF integration technology - such systems, in turn, may enable large neural network accelerators, GPUs, and microdatacenters.

2.9 Summary and Conclusions

Processor packages can significantly impact the bandwidth, allowable TDP, and area taken up by a processor. We proposed packageless processors - processors where the packages are removed and PCB-based integration is replaced by a Silicon Interconnection Fabric, a novel interconnection technology that involves mounting dies directly on a silicon wafer using copper pillar-based I/O pins. We showed that packageless processors can have one to two orders of magnitude higher memory bandwidth, up to 70% higher allowable TDP, and 5X-18X lower area than conventional packaged processors. These benefits can be exploited to increase processor performance. For a set of NAS and PARSEC benchmarks, we showed performance improvements up to 58% (16% average), 136% (103% average), and 295% (80% average) resulting from improved memory bandwidth, processor TDP and processor footprint respectively. For the same performance, packageless processing reduces compute subsystem footprint by up to 76% or equivalently increases TDP by up to 2X. The benefits from packageless processing should only increase with increasing I/O and performance demands of emerging applications and processors.

CHAPTER 3

Pathfinding for Chiplet Interconnect Technologies

As conventional technology scaling becomes harder, 2.5D integration provides a viable pathway to building larger systems at lower cost. Therefore recently, there has been a proliferation of multiple 2.5D integration technologies that offer different interconnect characteristics such as wiring pitch, bump/pad pitch, inter-die distance, etc. All these factors affect the interconnect metrics of bandwidth, latency and energy-per-bit which ultimately determine system performance. There are other factors such as the choice of ESD circuitry, dicing technology and signaling voltage that also influence these interconnect metrics. In this work, we propose a novel pathfinding methodology for 2.5D interconnect technologies, which seeks to identify the trade-offs among the different factors which affect the performance metrics. We show that incessant scaling of the critical dimensions of the interconnect is not very useful. We emphasize the importance of managing ESD and dicing in improving energy efficiency of these interconnects. We also show that a heterogeneous chiplet ecosystem comes with significant I/O energy penalties. Overall, we demonstrate that a holistic approach considering features of 2.5D integration technology, chiplet technology and various other factors need to be considered and optimized simultaneously to maximize the performance and cost benefits of these integration solutions.

3.1 Introduction

On one hand, transistor scaling is becoming more difficult and costly; on the other hand, the demand for larger System-on-Chips (SoCs) is growing rapidly as a result of growing need for performance. Large monolithic implementation of SoCs in advanced technology nodes often suffer from yield issues and are costly to design and manufacture. As an alternative, instead of building large monolithic dies, the components of an SoC can be separately manufactured in to disparate chiplets and integrated on a separate interconnect substrate shown in Fig. 3.1. These chiplets would be smaller, thus resulting in better yield of manufacturing, and the chiplets can be manufactured in suitable technology nodes for additional cost and performance optimization opportunities [10, 85].

However, to enable SoC like performance and energy efficiency, the interconnects on the substrate should closely resemble those of on-chip interconnects. Unlike conventional multi-chip module (MCM) substrates [86, 87] or PCB based interconnects which have coarse interconnect bump and wiring pitch, recent advancement in 2.5D technologies (such as silicon interposer [5]) allow communication between chiplets at high bandwidth, energy efficiency and low latency. This is achieved by: (1) manufacturing the substrates using mature semiconductor back-end-of-the line (BEOL) technology, such as $65nm$ or $90nm$ process node and (2) using fine-pitch μbumps[1] (pitch of below $70\mu m$) to connect the flip-chip bare dies to the integration substrate for larger I/O density. Therefore, these technologies enable high performance multi-chiplet systems without the traditional off-chip communication bottlenecks.

Several chiplet integration technologies have already been commercialized and many others are under active development. Examples include TSMC's CoWoS [6, 88], InFO [89], Intel's EMIB [7], Samsung I-Cube [90], Amkor's CoS, CoW, HDFO technologies [91], Silicon Interconnect Fabric (Si-IF) [92, 93], etc. These technologies offer minimum μbump pitch in the range of $10\mu m$ - $65\mu m$, minimum wire pitch in the range of $0.4\mu m$ - $4\mu m$, and 2-4 layers of metal routing.

The number of links between dies and the characteristics (bandwidth, energy, latency) of these links depend on multiple factors such as μbump/copper pillar sizing, wire sizing, inter-die spacing (length of the links), number of metal layers available for routing, ESD circuitry, etc. Past research [94, 95] has focused on one or few of these factors and discussed their scaling impacts. In this work, we focus on silicon based 2.5D interconnects such as silicon interposer, EMIB and Si-IF and comprehensively investigate all the multiple factors that affect the energy-bandwidth-latency scaling of inter-die links and highlight the trade-offs that exist between them. We develop a 2.5D interconnect pathfinding framework that takes all the design parameters as inputs, along with tech-

[1]In this work we use μbump to refer to copper pillars, solder bumps or other bonding interfaces.

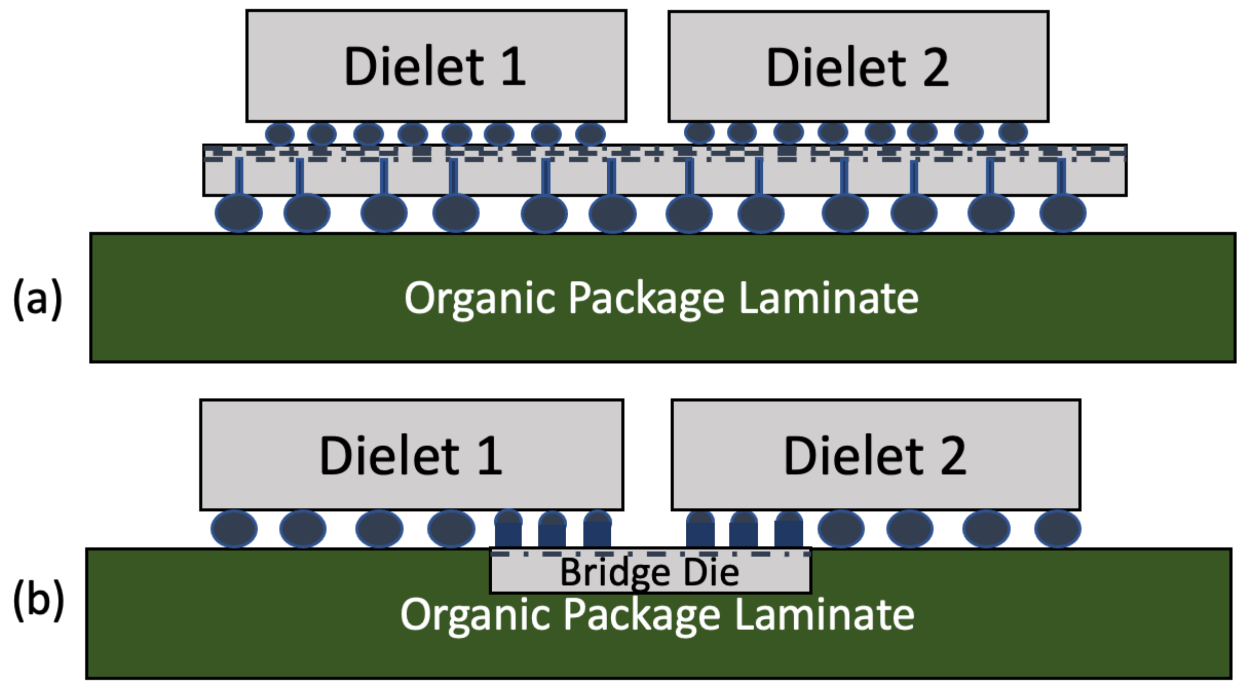


Figure 3.1: Cross-section view of two 2.5D substrates: (a) Silicon Interposer [5,6], (b) EMIB [7].

nology constraints and system design requirements (e.g. perimeter bandwidth density) and ranks all possible substrate designs in descending order of parameter set dimensions and energy-per-bit. We then analyze the link energy-per-bit and perimeter bandwidth density of these design points to understand and evaluate the trade-offs that come with scaling the critical dimensions of the multiple design parameters. Knowing this would help understand the exact parameters that need to be scaled in order to obtain substantial link energy, bandwidth and latency gains. For example, for a fixed number of routing metal layer, ESD capacitance and minimum inter-die link length, what is the optimal wire and μbump pitch beyond which scaling down only incurs additional manufacturing cost overhead while providing negligible benefits?

The rest of the chapter is organized as follows: Section 3.2 discusses the interconnect pathfinding framework flow and the different components of our interconnect model that we used in our analysis. Section 3.3 covers our detailed analysis and highlights the main takeaways from our study. Section 3.4 discusses the trade-offs between designing a 2.5D interconnect substrate for homogeneous vs. heterogeneous chiplet ecosystems. Section 3.5 concludes the chapter.

3.2 Chiplet Interconnect Pathfinding Framework

In this work, we propose a framework that evaluates trade-offs that come from scaling and varying physical link and interconnect substrate parameters. The framework helps assess return on (technology) investment in inter-chiplet interconnect and integration substrate.

3.2.1 Pathfinding Flow

Fig. 3.2 shows the chiplet interconnect pathfinding framework flow. The framework takes system design parameters (wire and μbump dimensions, number of metal routing layers, minimum link length), technology constraints (ESD capacitance, inter-layer dielectric[ILD] thickness, wire thickness, inter-die spacing, etc.), and system constraints (e.g., perimeter bandwidth density) as inputs. Based on the inputs, the 2.5D interconnect design space is enumerated. For each interconnect design (a set of parameter values), we apply the interconnect length model and wire parasitic model to compute the maximum inter-die link length and the link parasitics. Based on this, we calculate the maximum load that needs to be driven by the transmitter. We assume that homogeneous transceiver circuitry would be used for all the neighboring inter-die communication links and therefore, we appropriately size the transceiver circuitry to support the maximum inter-die capacitive load. The models and transceiver circuit are then provided as inputs to our HSPICE [96] based simulation framework to calculate the latency and energy-per-bit for each design. Once the link characteristics are enumerated for all designs that meet the system constraints, the framework ranks the designs in descending order of parameter set dimensions and energy-per-bit.

3.2.2 Interconnect Modeling

In 2.5D integration, bare chiplets are directly bonded on the interconnect substrate. Since the dies are un-packaged, inter-die spacing is small and the link lengths can be as small as $100\mu m$ and usually the maximum length of the inter-die wires is about $5mm$. Moreover, since abundant interconnect wiring resources are available in the 2.5D substrates, they are operated at a few GHz and the interfaces are usually designed as parallel interfaces instead of serialized/de-serialized

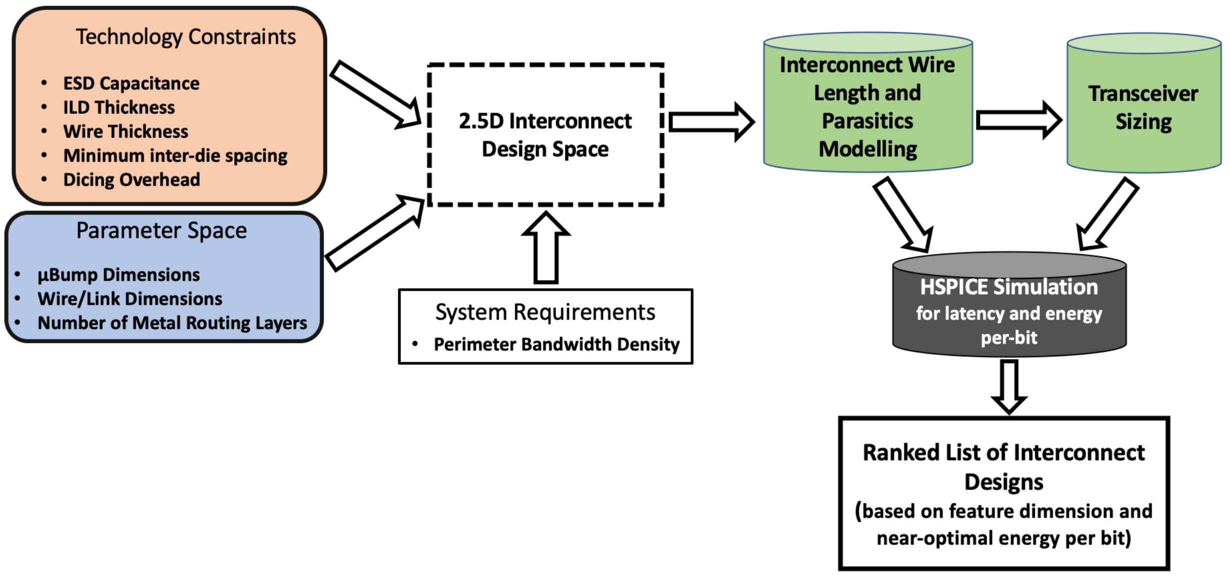

Technology Constraints
ESD Capacitance
ILD Thickness
Wire Thickness
Minimum inter-die spacing
Dicing Overhead
Parameter Space
µBump Dimensions
Wire/Link Dimensions
Number of Metal Routing Layers
2.5D Interconnect Design Space
System Requirements
Perimeter Bandwidth Density
Interconnect Wire Length and Parasitics Modelling
Transceiver Sizing
HSPICE Simulation for latency and energy per-bit
Ranked List of Interconnect Designs (based on feature dimension and near-optimal energy per bit)

Figure 3.2: Chiplet interconnect pathfinding framework

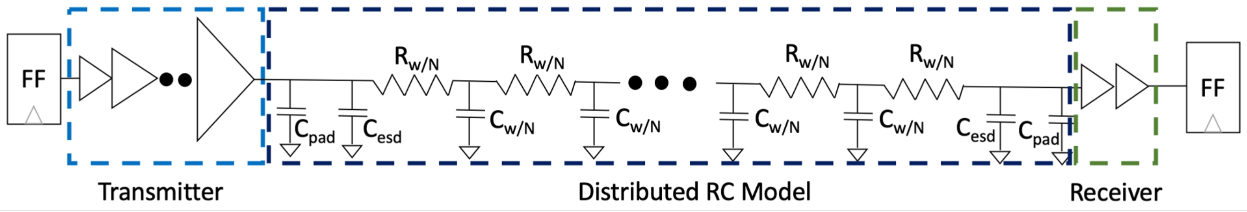


Figure 3.3: Distributed wire model for interconnect link. C_{pad} is the pad/μbump capacitance, C_{esd} is the capacitance introduced by the ESD protection circuitry, $R_{w/N}$ and $C_{w/N}$ are the wire segment resistance and capacitance respectively.

interfaces (SerDes [97]) that are used in conventional coarse-grained interconnect substrates. As a result, the transmitters and receivers can be designed using simple appropriately-sized cascaded inverters. We build link and μbump models to calculate the inter-die link parasitics and maximum length, and μbump parasitics based on the input parameter dimensions. We also appropriately size the transmitter circuitry based on the load it is driving. Next, we describe the components of our model in detail.

Modeling Wire Parasitics: In this work, we model repeater-less interconnect links that are found in today's passive integration substrates using a multi-segment Π model as shown in Fig. 3.3. We explore multiple wire and I/O pad parameters such as width, length, spacing. For each combination of these parameters, we calculate the link parasitics using the model proposed in [98] and validate the results against experimental results in [99]. We take in to account both neighboring wire coupling capacitance as well as the substrate ground capacitance.

The typical wire lengths in these interconnect technologies do not exceed a few millimeters. Therefore, the inductance effect is negligible [99] and the links behave as RC links. We further verified this effect by including inductance in a subset of our experiments.

Modeling Interconnect Link Length: Multiple columns of staggered I/O μbumps are used to support the interconnect wires that escape the periphery of the die per routing layer. In Fig. 3.4, we show an example with μbump pitch that is 4x of the wire pitch. To support the maximum possible wire density that can escape in one layer, four columns of I/O μbumps are required. As the ratio of μbump pitch to wire pitch increases, the number of I/O columns also increases. In addition to this, as the number of routing layers increase, the number of columns of μbumps increase as well. Therefore, the maximum length of the interconnect link increases as the columns grow orthogonal to the edge of the die. We calculate the worst case link length (l_{max}) using equation 3.1. It can be seen from equation 3.1 that scaling down the μbump pitch not only reduces the number of columns, it also reduces the additional link length per I/O column as shown in Fig. 3.5. As l_{max} increases, the capacitive load as well as resistance of the link increases.

$$l_{max} = l_{min} + (\frac{IO_{pitch}}{wire_{pitch}}) \times IO_{pitch} \times (2 \times N_{layers} - 1) - IO_{pitch} \tag{3.1}$$

where, l_{min} is the minimum distance between the μbumps in the neighboring dies, N_{layers} is the number of routing layers as shown in Fig. 3.4. l_{min} primarily depends on two factors: (1) inter-die spacing, and (2) distance of the first column from the edge of the die. Inter-die spacing can range from as low as $50\mu m$ (requires precise die placement and low die edge roughness) [7, 88, 93] to usually a few millimeters [100]. The first I/O column is placed at a distance from the edge of the die to accommodate dicing channel and sealring. This distance varies across foundries and processes and usually lies between $50\mu m$ - $200\mu m$. Therefore, l_{min} has to be at least $150\mu m$.

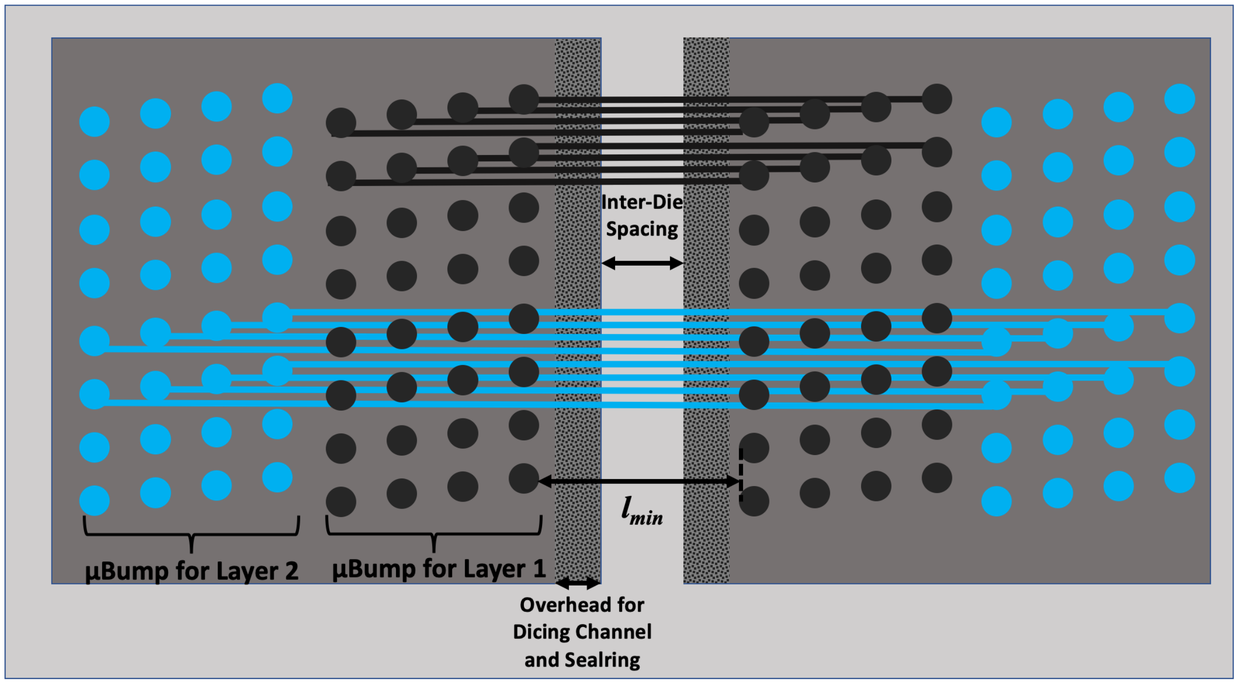


Figure 3.4: Top-down view of two dies on a 2.5D substrate with μbumps and interconnect wiring on multiple layers.

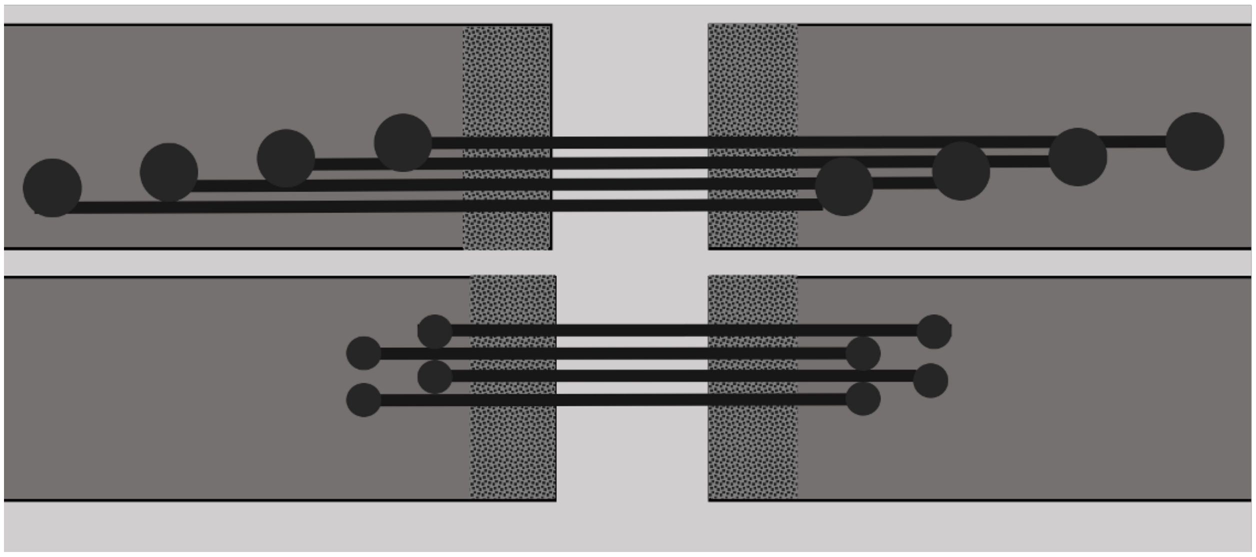
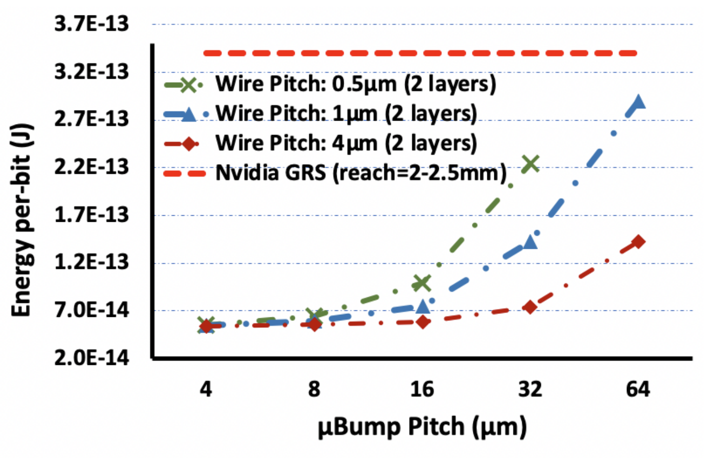
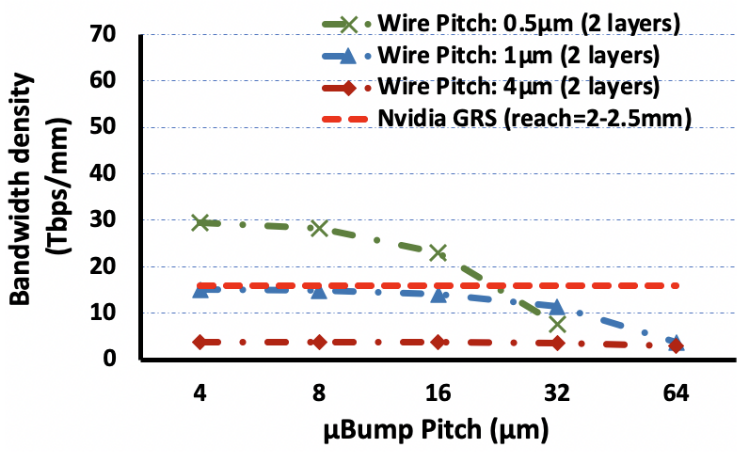


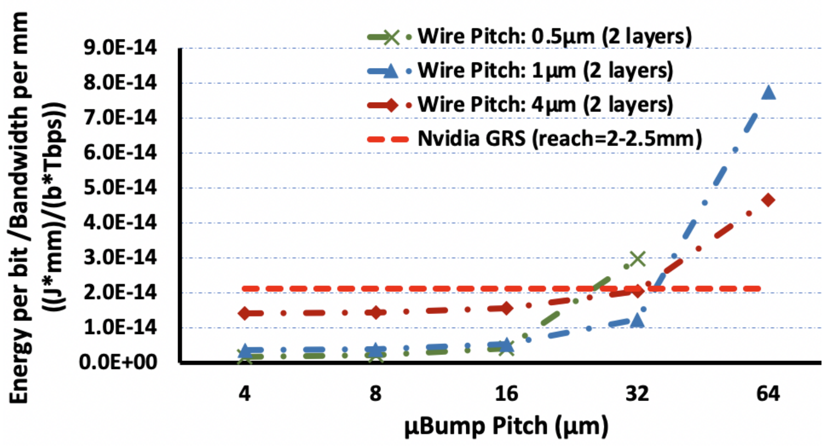
Figure 3.5: Smaller μbumps help reduce the interconnect length.



(a) Energy-per-bit

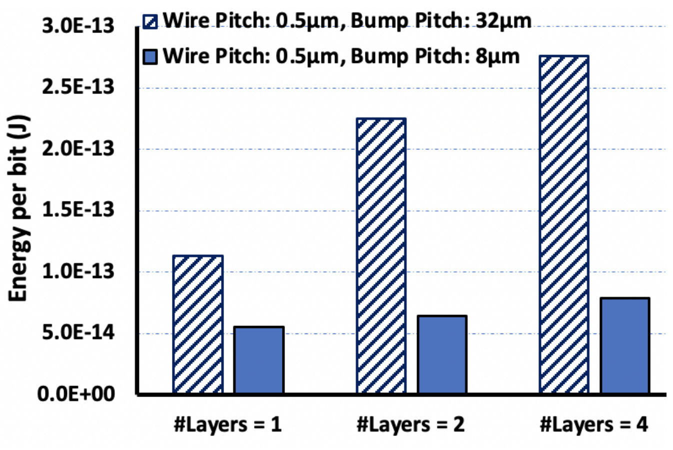


(b) Bandwidth-per-mm

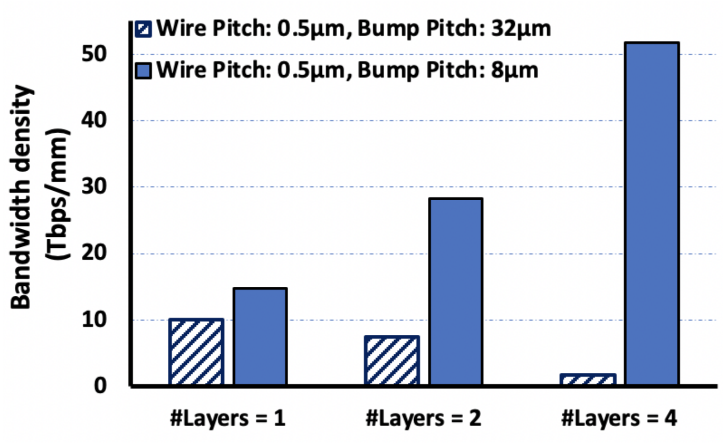


(c) Energy-per-bit per bandwidth-per-mm

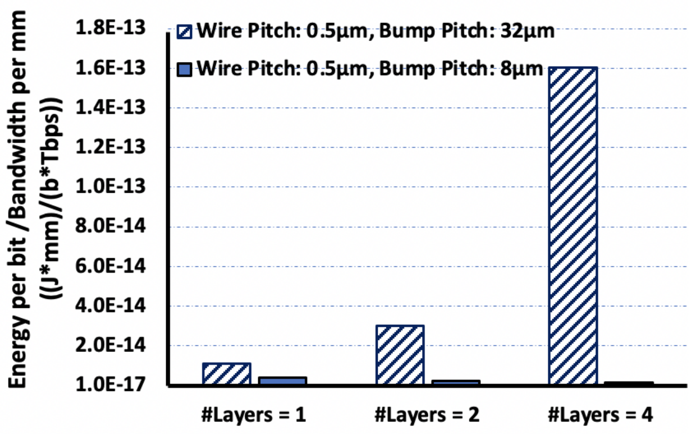
Figure 3.6: Scaling of energy-per-bit, bandwidth-per-mm and their ratio with μbump pitch and wire pitch for two metal routing layers (C_{esd}=50fF, l_{min}=150μm).



(a) Energy-per-bit



(b) Bandwidth-per-mm



(c) Energy-per-bit per bandwidth-per-mm

Figure 3.7: Scaling of energy-per-bit, bandwidth-per-mm and their ratio with metal routing layers for fixed wire pitch and two different μbump pitch values (C_{esd}=50fF, l_{min}=150μm).

Table 3.1: Parameter values used in our analysis

Parameters	**Values**
$wire_{pitch}$ (width=spacing)	{0.5, 1, 2, 4} μm
IO_{pitch} (width=spacing)	{4, 8, 16, 32, 64} μm
Wire thickness aspect ratio	1.5x (of wire width)
ILD thickness ratio	2x (of wire thickness)
Min. bump-to-bump dist. (l_{min})	{50, 150, 500, 1000, 2500} μm
ESD capacitance (C_{esd})	{0, 20, 50, 100, 200} fF
Metal routing layers (N_{layers})	1, 2, 4
Flip-Flop - $t_{clk-Q+setup}$ (45 nm)	62ps

Modeling I/O μbump Parasitics: We calculate the capacitance of a μbump (C_{pad}) using the model proposed in [101]. We augment this model by accounting for the coupling capacitance added by the surrounding μbumps and validate the results against the capacitance values presented in [48, 99].

Electrostatic Discharge (ESD) Circuitry Overhead: The individual dies that are placed on the interconnect substrate have to go through multiple post manufacturing processes like die thinning, known good die (KGD) testing, bonding etc. As such, the chiplets are prone to electrostatic discharge related incidents which can potentially damage the I/O circuitry resulting in die yield loss [102]. Therefore, the I/O pads need protection against these catastrophic ESD events, and is provided using large-sized high current carrying diodes. These diodes add significant capacitive load to the interconnect. We model this capacitive load as additional capacitors (C_{esd}) [48] on either end of the interconnect wire as shown in Fig. 3.3.

Transceiver Sizing: A cascaded inverter transmitter is used starting with the minimum sized inverter for a given technology Process Design Kit (PDK) and subsequent stages sized to drive a load equivalent to fan-out of four or less. We change the number of stages depending on the amount on load the transmitter is driving. The maximum sized inverter that we use is 128x the minimum size (five stages). The receiver is designed using two minimum sized inverters.

Table 3.1 shows the parameter exploration space for all the components of the interconnect model. We use HSPICE [96] and 45nm PDK to simulate the model and measure energy-per-bit, propagation delay/latency, rise/fall times. We calculate energy-per-bit by averaging over a pseudo-random binary sequence (PRBS). To calculate the maximum achievable bandwidth, we assume two flip-flops on either side of the interconnect link and consider the clock-to-Q delay and setup time in addition to the link latency. Moreover, we use three different technology nodes to show the trends from technology scaling.

3.3 How Should We Scale the Interconnect Substrate?

As mentioned earlier, there has been a recent trend in scaling the μbump size and pitch. Conventional μbumps are made of solder and thermo-compression bonding is used while attaching the die to the substrate. However, solder extrusion issues limit the scalability of the μbumps. Several alternative technologies such as solder-on-copper-pillar [103] and direct copper-to-copper bonding [93] has been proposed to shrink μbump size and pitch to sub-25μm and sub-10μm range, respectively. Though this allows us to pack more μbumps under the die area, these advanced processes are often more complicated and adds to the cost of bonding and assembly. Here next, we evaluate the efficacy of μbump scaling on the characteristics of the inter-die link.

Scaling μbump pitch vs wire pitch: We study the impact of scaling down the μbump pitch on energy-per-bit and perimeter bandwidth density for three different wire pitches and three different metal routing layer schemes. We keep l_{min} and C_{esd} fixed at 150 μm and $50fF$, respectively. Scaling down μbump pitch while keeping the wire pitch constant (shown in Fig. 3.5) decreases the number of staggered I/O columns needed per routing layer to support all the wiring, which, in turn, leads to a reduction in the maximum inter-die link length. As seen in Fig. 3.6a, for the same wire pitch, the energy-per-bit reduces as μbump pitch is scaled down from 64μm.

However, Figures 3.6a, 3.6b and 3.6c show that scaling down the μbump pitch indefinitely does not improve energy-per-bit or bandwidth. This is because eventually the parasitics coming from

l_{min} portion of the wire, C_{esd} and C_{pad} dominate. Due to the maximum link length dependence, the μbump pitch beyond which the benefits saturate increases with increase in wire pitch. This "saturation" happens at μbump pitch of $16\mu m$ for a wire pitch of $1\mu m$ and at $32\mu m$ for a wire pitch of 4 μm (e.g. in EMIB). Therefore, for a fixed wire pitch, incurring additional processing cost to reduce the μbump pitch beyond the saturation knee point might not be beneficial in terms of improving the link characteristics.

Takeaway: Incessant scaling of μbump pitch is not beneficial as the wire load is eventually dominated by ESD capacitance and inter-die separation.

Impact of additional metal routing layers: To support higher bandwidth density, one option is to increase the number of metal routing layers. This results in an increase in the the total amount of wiring per unit die edge. However, the number of I/O columns required to support all the wiring also increases. This, once again, leads to an increase in the worst case link length. In order to offset this effect, it is beneficial to scale down the μbump pitch as seen in Figure 3.7.

Another interesting observation is that even though increased number of metal layers help increase wiring resources, beyond a certain μbump pitch (>16μm), the added parasitics because of longer wires completely offset the gain from increased wiring and adversely affects the bandwidth/mm. This can be seen in Figures 3.7a and 3.7b. For the wire pitch of $0.5\mu m$ and μbump pitch of $32\mu m$, the bandwidth/mm with four routing layers is 5x lower than that with a single routing layer. This is especially true at low wire pitch values where the wire resistance, and coupling and area-fringe capacitance values are high. The opposite is true for smaller μbump pitch (<16μm) where increasing the number of metal layers increases the bandwidth almost linearly while having negligible ($< 1.5x$ when increasing from 1 to 4 metal layers) impact on energy-per-bit. Hence, the energy to bandwidth ratio in Fig. 3.7c decreases with increase in metal routing layers for μbump pitch of $8\mu m$.

Takeaway: Increasing the number of wiring layers *must* be accompanied by a correspondingly smaller μbump pitch to derive bandwidth benefits from available increased wiring.

Comparison with μSERDES scheme: We also compare the link bandwidth and energy-per-bit with a μSERDES scheme. We use Nvidia's on-chip Ground Referenced Signaling (GRS) scheme [104] as the baseline for comparison against parallel interfaces on 2.5D substrates. The GRS links for which the energy-per-bit and bandwidth/mm values are plotted in Figures 3.6a and 3.6b have a wire pitch of $2\mu m$, reach of 2-2.5mm and run at 16GHz. As can be seen in Figure 3.6b, parallel interface with wire pitch of $1\mu m$, can achieve comparable bandwidth as that of the serialized/de-serialized GRS link. However, the parallel interfaces on 2.5D substrates achieve that bandwidth at a much lower energy cost. For example, with two metal routing layers, a parallel interface of $1\mu m$ wire pitch and $16\mu m$ μbump pitch has almost the same reach and achieves the same perimeter bandwidth density as GRS but at 4.5x lower energy cost. This is because, as mentioned earlier, the parallel links operate at a few GHz and the simple transceiver circuitry has negligible energy overhead. We show this later in Fig. 3.12 that transceiver circuitry energy is less than 10% of the total link energy. On the other hand, serialized/de-serialized links have much larger transceiver overheads that often dominate the overall link energy. With even smaller wire pitch, the parallel interface bandwidth/mm can be higher than the GRS links. However, the μbump pitch has to be scaled down significantly (16 μm or lower) to achieve any bandwidth benefits (Fig. 3.6b) and below 32μm for any bandwidth-normalized energy gains (Fig. 3.6c).

Takeaway: High bandwidth systems which want to move away from complex, energy-hungry serial links should aim for μbump pitches smaller than 16μm, and μbump pitches below 32μm are essential for leveraging parallel link energy efficiency benefits.

Impact of ESD capacitance (C_{esd}): C_{esd} adds a significant amount of load to the interconnect link. In general, about $50fF$ capacitance is added by the ESD diodes on each side of link [48]. When the maximum link length is small, C_{esd} dominates the link energy and latency. Therefore as mentioned earlier, reducing the link length or reducing the bump pitch (i.e., below the knee points in Fig. 3.6) doesn't help in improving the overall link characteristics.

On one hand, with strict ESD control in modern advanced foundries during manufacturing, testing and bonding, the amount of ESD protection required is expected to decrease [105]. On the other hand, the 2.5D ecosystem is expected to accommodate dies from different foundry sources including

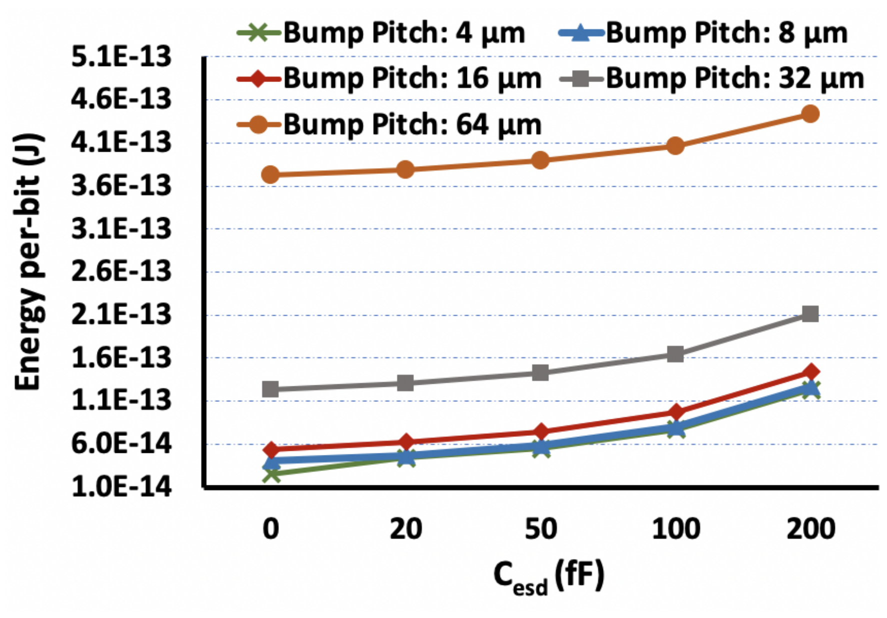


Figure 3.8: Energy-per-bit scaling with C_{esd} and μbump pitch (l_{min}=150μm, $wire_{pitch}$=1μm, N_{layers}=2).

older non-advanced foundries. As a result, some dies can in fact come with ESD circuitry with even larger amount of capacitance such as up to $200fF$ to $300fF$ [95, 105]. Here, we perform sensitivity analysis of ESD diode capacitance overhead.

As expected, in Figures 3.8 and 3.9 we see that when C_{esd} increases, even at smaller μbump sizes, the energy-per-bit and bandwidth density are considerably worse. Moreover, energy-per-bit and bandwidth degrades by a smaller fraction when moving to larger μbump sizes than compared to the case when C_{esd} is small. This is due to the smaller amount of load that is added by the additional wire length compared to the fixed overhead of C_{esd} when its value is large. Alternatively, if C_{esd} is small, μbump scaling can provide larger gains in energy efficiency and bandwidth.

Interestingly, reducing C_{esd} from 200fF to 50fF gives energy/bandwidth benefits comparable to reducing bump pitch from 32μm to 16μm. Therefore, it may be worthwhile to control ESD events in the entire manufacturing and handling process rather than moving to an aggressive (and costly) μbump size.

Takeaway: C_{esd} can be used as a lever to scale both energy-per-bit and bandwidth and can enable us to stay at larger μbump pitches.

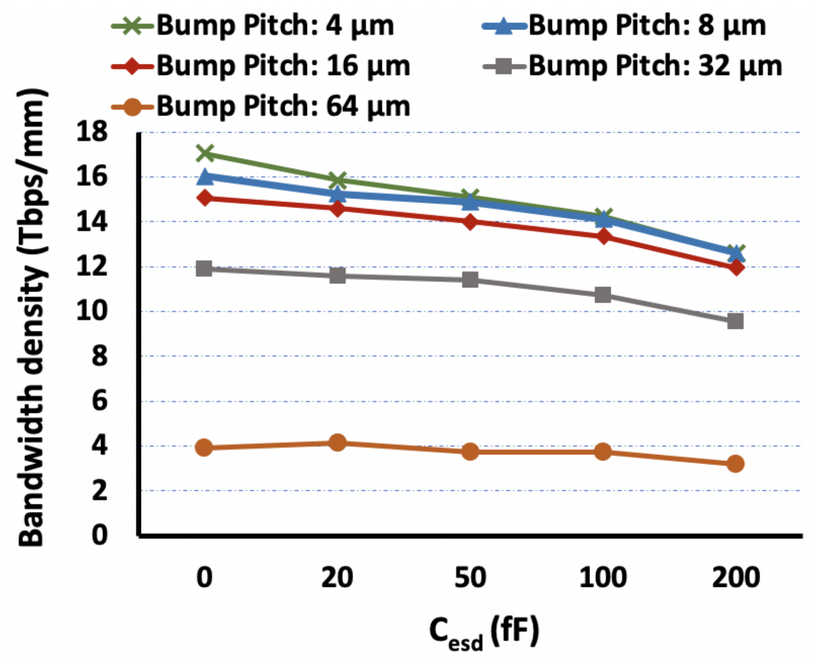


Figure 3.9: Bandwidth density scaling with C_{esd} and μbump pitch (l_{min}=150μm, $wire_{pitch}$=1μm, N_{layers}=2).

Impact of inter-die spacing and dicing overhead: Inter-die spacing and dicing related guard-bands impact the minimum distance between the μbumps on adjacent dies. Usually, mechanical dicing (using dicing saw) is used to singulate the dies on a wafer. This process often creates rough die edges and therefore the width of the die can vary by up to 50μm. Though advanced place and bond tools can achieve inter-die spacing of 50μm or less [88, 93], the dies are usually placed apart at minimum by more than 100μm to avoid die-to-die collision during bonding. Moreover, stress fracture and cracking at the edge of the die is a common occurrence with mechanical dicing [106] and therefore, seal ring and crack stops are added around the perimeter of design of the die [107]; this also affects l_{min}. On the other hand, plasma etch based dicing solutions [108] claim to reduce the die edge roughness as well as have minimal mechanical stress. Therefore, these solutions can reduce the overhead to below 10μm which can potentially reduce l_{min} to about 50μm. Next, we analyze the effect of inter-die spacing on energy-per-bit and bandwidth density of 2.5D substrates to understand if and when advanced processing for die singulation and better inter-die spacing is required.

Similar to the effect of C_{esd}, the capacitive load added by the minimum length wire (l_{min}) affects

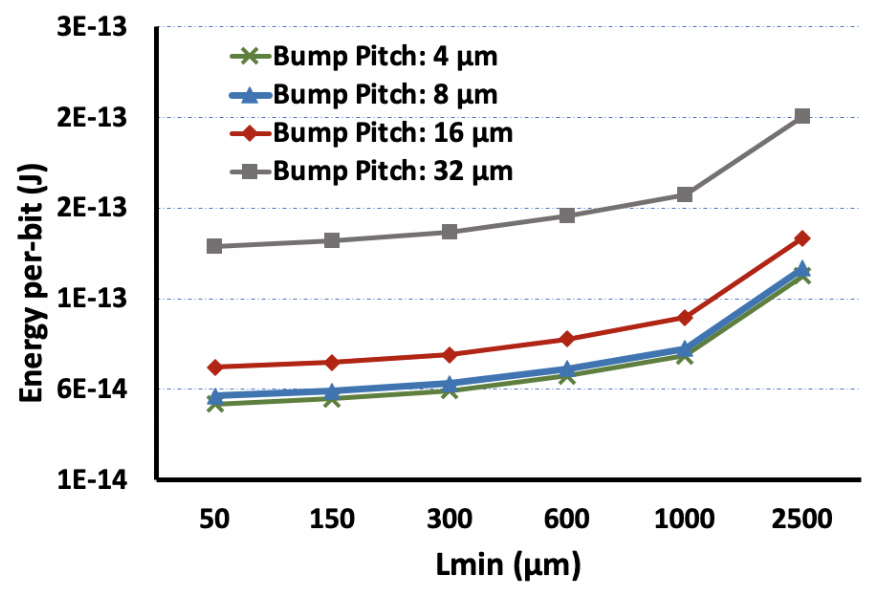


Figure 3.10: Energy-per-bit scaling with l_{min} and μbump pitch (C_{esd}=50fF, $wire_{pitch}$=1μm, N_{layers} = 2).

overall link characteristics. As shown in Figures 3.10 and 3.11, as l_{min} decreases, the link characteristics improve. However with C_{esd} of 50fF, reducing the inter-die separation to below 300μm seems to have limited use. 300μm is easily achievable using current generation dicing processes and place and bond tools, indicating that technology investment into better dicing technologies may provide limited benefit. Tighter inter-die spacing would be more useful only if ESD protection requirements can be reduced significantly.

Takeaway: Inter-die separation of 300μm which is achievable by current generation dicing and die placement processes is good enough.

3.4 Interconnects in the Chiplet Ecosystem

As mentioned earlier, 2.5D interconnects can help lower system costs by enabling us to partition larger monolithic dies and re-integrate smaller and high yielding component dies on a 2.5D substrate. On the other hand, the chiplet based eco-system promises to be a platform for heterogeneous integration where chiplets from different technologies and vendors can be assembled to build customized and cost-performance optimal systems. These two use cases have dramatically different

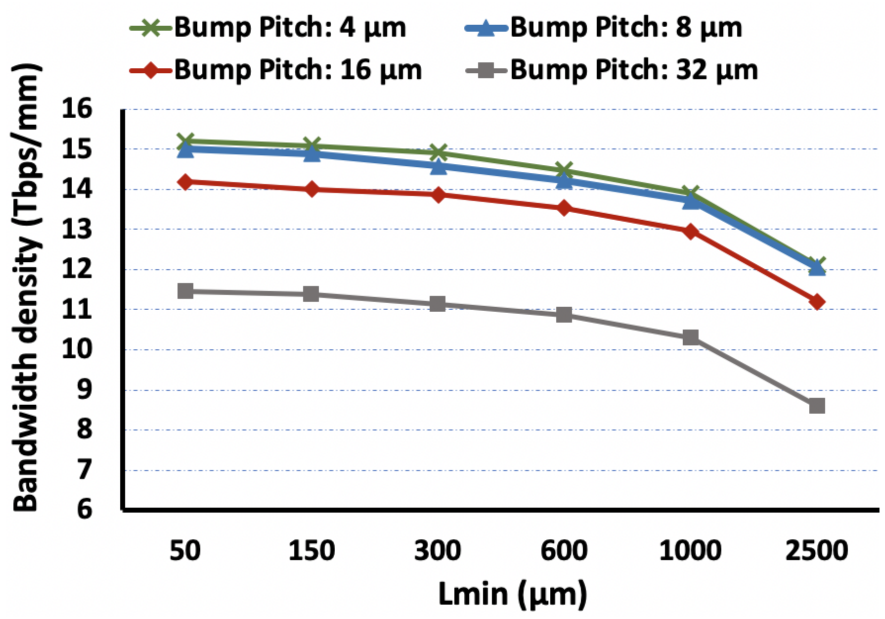


Figure 3.11: Bandwidth density scaling with l_{min} and μbump pitch (C_{esd}=50fF, $wire_{pitch}$=1μm, N_{layers}=2).

effect on the characteristics of the 2.5D interconnects as the design of the transceivers has to be done differently for the two cases.

For the case where all the chiplets come from the same technology, the transceivers on all the chiplets would be homogeneous and can be designed to operate at the core voltage offered by the technology. As the technology node scales down, the voltage of operation usually reduces. This has a quadratic impact on the energy required to switch the wire load. In Fig. 3.12, we show the energy-per-bit scaling of the interconnects for the different technology nodes of the transceiver circuitry for iso-bandwidth case (the transceivers were sized appropriately).

On the other hand, in a heterogeneous chiplet scenario, the chiplet operating at the highest voltage will determine the peak voltage of operation for the interconnect. Therefore, even though a chiplet can be manufactured in an advanced technology node, the benefits of voltage scaling won't be available. In order to operate transistors in advanced technology node at higher voltage, thick oxide devices may need to be used which could further degrade drive strength. This would require larger transistors resulting in increased transceiver energy although we expect the impact to be small (fraction of the energy spent in the transceiver itself, as shown in Fig. 3.12, is less than 10%).

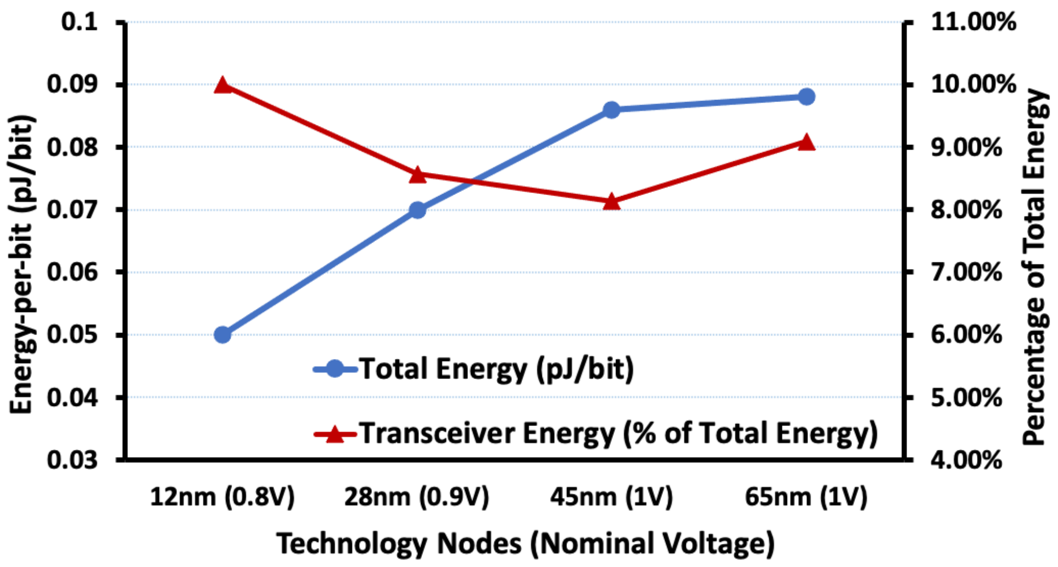


Figure 3.12: Total link energy and transceiver energy as % of total energy for for dielets (transceiver circuitry) from four different technology nodes. Link length: 1.5mm, W/S:0.5μm, C_{esd}:50fF.

The minimum operating voltage of the link will be governed by the *oldest* technology the chiplet ecosystem supports. For example, in Fig. 3.12, a link that supports 45nm-12nm heterogeneous integration will be 70% less energy efficient than 12nm homogeneous integration link.

Takeaway: Link efficiency requirements may need to limit the technologies supported by a chiplet ecosystem.

3.5 Discussion and Conclusions

Several commercial products today such as Nvidia and AMD GPUs, Xilix and Intel FPGAs, etc. use 2.5D substrates such as CoWos and EMIB to build large systems. As the demand for data-intensive and highly parallel applications grows and technology scales to fit in more compute per die, the amount of resources dedicated to 2.5D interconnect substrates is likely to rise. Therefore, it would be important to scale these interconnects in order to achieve performance and energy scaling proportional to the rest of the system. Nevertheless, it is important to keep a full-system perspective. As an example, let us consider high bandwidth memories (HBM) integrated on silicon interposers that are used in high performance systems today. For HBM2, about 10%-15% (0.4pJ out of 3.9pJ) [109] of memory access energy is used to shuttle data between the memory and compute dies. Memory energy itself would be a fraction of total compute energy ($\sim$20% for GPUs [110]) implying system-level energy benefits may be modest from link energy improvements. However, the benefits may be more significant when the non-interconnect part of memory energy is improved or for specialized communication-limited applications such as graph processing or streaming architectures.

Several efforts have been underway (e.g. DARPA CHIPS program [111], Open Compute Project [112]) to design interfaces and protocols to allow multiple chips to communicate. The implementation of these protocols require additional logic which ultimately adds to the inter-chiplet communication overhead. At just 0.1 pJ/bit of link energy or less, large amounts of bandwidth, e.g. 10TBps, can be supported with about 8W of power. However, protocol level logic and synchronization requirements can add 2-5X extra energy overhead [113]. Therefore, alongside optimization of 2.5D interconnect parameters, lightweight and energy efficient protocols need to be designed to enable overall communication energy reduction.

As conventional technology scaling becomes challenging, 2.5D integration provides a viable pathway to compose larger systems using smaller, high yielding dies. Therefore recently, there has been a proliferation of different 2.5D integration technologies. However, the success of this 2.5D approach depends upon optimizing the performance benefits and cost for different use case scenarios. In this work, we develop a pathfinding methodology for 2.5D interconnect technologies and use it to study inter-chiplet interconnect performance and energy as a function of dimensional and technology parameters. We demonstrate that a holistic approach considering features of 2.5D integration technology, chiplet technology and processing techniques. Our analysis indicates that beyond certain point, dimensional scaling (wire and bump pitch) provides marginal benefit in terms of energy-per-bit and bandwidth density; while other factors such as ESD and chip dicing technologies may provide additional levers for further interconnect scaling.

CHAPTER 4

Design Space Exploration for Chiplet Assembly Based Processors

Recent advancements in 2.5D integration technologies have made chiplet assembly a viable system design approach. Chiplet assembly is emerging as a new paradigm for heterogeneous design at lower cost, design effort and turnaround time, and enables low cost customization of hardware. However, the success of this approach depends on identifying a minimum chiplet set which delivers these benefits. We develop the first microarchitectural design space exploration framework for chiplet assembly based processors which enables us to identify the minimum set of chiplets to design and manufacture. Since chiplet assembly makes heterogeneous technology and cost-effective application-dependent customization possible, we show the benefits of using multiple systems built from multiple chiplets to service diverse workloads (up to 35% improvement in energy-delay product over a single best system), and advantages of chiplet assembly approaches over SoC methodology in terms of total cost (up to 72% improvement in cost) while satisfying the energy and performance constraints of individual applications.

4.1 Introduction

Microarchitectural design space exploration for a general purpose processor is traditionally aimed at determining the microarchitectural parameter values of *one* processor system that has the highest performance or efficiency for a set of representative applications. The goal is to arrive at the *one* system that has the highest performance or efficiency for these applications. However, applications are often targeted by using a family of processors where each processor in the family targets a subset of the applications. Consider Intel Xeon Product Family [114], for example. The product family targets non-consumer workstation [115], server [116], and embedded [117] applications using different processors in the family. Even within Xeon Processor E7 family [115], there is a large number of processors each targeting a different subset of workstation and server applications. Using multiple systems to target applications provides an effective way to address heterogeneity in workloads, objective functions (power vs performance), compute conditions (battery-powered vs wall-powered), and cost.

Unfortunately, a countervailing trend—increasing processor design, verification, manufacturing, and management costs [118] [119]—is putting immense pressure on the number of systems that can be used to target applications. As these costs rise, designing and manufacturing a large number of SoCs may become infeasible.

As a result, new design and assembly methods [120] [121] [122] [54] [123] are being developed and commercialized where different chiplets can be connected using SoC-like low-latency and high bandwidth interconnect substrates. Thus, a large processor SoC can now be disintegrated into multiple smaller component chiplets and then reintegrated into a full processor system. In fact, critical processor components can be partitioned into separate chiplets and integrated on these high performance interconnects without significant performance degradation ($<5\%$) compared to SoCs [123]. Since smaller chiplets often achieve better yield, this approach may decrease overall system cost. Moreover, one can create many systems using different combinations from a set of chiplets, allowing a designer to tailor each system towards a particular subset of applications. Therefore, reuse of chiplets across several systems can help amortize the cost of chiplet design and improve turnaround time. Furthermore, chiplets implemented using different technology nodes can be integrated into the same system - this may allow further reduction in design and manufacturing costs. Overall, chiplet assembly is emerging as a new paradigm for heterogeneous design at lower cost, design effort and turnaround time, and enables low cost customization of hardware. However, the success of this approach depends on identifying a minimum chiplet set which delivers these benefits. Algorithmically exploring the design space and then investigating cost, performance, energy trade-offs of such chiplet-based customization is the goal of our work.

In this work, we ask three questions: 1) How should one set up the microarchitectural design space exploration problem when a set of component chiplets will be used to build a set of systems to target different applications? 2) What are the microarchitectural characteristics of the different chiplets and the corresponding systems when each system targets only a subset of applications instead of the entire application set? 3) When total cost of design and manufacturing is concerned, what are the benefits of chiplet based assembly method and what chiplets and corresponding systems need to be built? We show that the design space exploration problem of identifying the best m chiplets to build k system configurations for a set of n applications is qualitatively and computationally very different from conventional design space exploration where $k = 1$. We present integer-linear programming (IntLP) based formulations for solving this selection problem. This is the first work on systematically performing a microarchitectural design space exploration when multiple systems will be used to target applications. This is also the first study on a methodology to determine the number and characteristics of chiplets required to target a set of applications. This work makes the following contributions:

- We develop the first optimization framework for solving the multiple chiplet/system selection problem.
- We show how chiplet assembly can deliver near custom system performance for *every* application with relatively small number of chiplets.
- We consider minimization of total design/manufacturing cost by simultaneously solving chiplet and technology selection. We also discuss chiplet characteristics when cost-aware optimization is performed and show the conditions under which chiplet assembly can be a major cost-saver compared to SoCs.
- We demonstrate the value of performing the chiplet design space exploration across different suites of applications (high performance to embedded applications) rather than performing the exploration per suite. Chiplets can be reused across different benchmark suites thus maximizing opportunity of design cost amortization.

The rest of the chapter is organized as follows. Section 4.2 discusses representative work on processor design space exploration and IP-reuse-based design. Section 4.3 motivates the value of multiple systems as well the need for an effective multi-system optimization strategy. Section 4.4

presents our optimization framework and cost model. Section 4.5 presents the experimental setup where we discuss our methodology, workloads and system/chiplets evaluated and the cost components. Section 4.6 discusses the results and the applicability of our framework in different scenarios such as multi-core and heterogeneous systems. Finally, section 4.7 concludes the work.

4.2 Related Work

4.2.1 Processor Design Space Exploration

A large body of prior work on design space exploration has focused on exploring the various parameters of a processor microarchitecture to maximize overall performance of a processor while minimizing its power and energy overhead. Papaworth [124] discusses optimizing the micro-architecture of the Intel Pentium Pro processors. Lee *et. al.* [125] propose a fast but accurate processor design space exploration approach that estimates the performance bottlenecks in a single core design and predicts the performance effects of tuning the bottlenecks. Frameworks such as *ArchExplorer* [126], *MULTICUBE* [127], *Magallan* [128], and *FADSE* [129] enable automatic design space exploration for multi-core architectures. Kumar *et. al.* [130] and Choudhary *et. al.* [131] try to find configurations of cores for a heterogeneous chip multi processor (CMP) where each core can be tuned for a class of applications sharing common attributes. However, in all these works, the goal is to still find the one, best performing processor, instead of a selection of processors.

The closest related work is by Li *et. al.* [132] and Kin *et. al.* [125]. They explore a multi-dimensional design space for multi-core architectures focusing on different related parameters of a processor: area, power, pipeline depth, superscalar width etc. They argue that it is challenging to accommodate different application classes (e.g., memory-bound and compute-bound) on one processor system and, therefore, one needs to find the optimized parameters for each class of applications. While these studies only focus on media processors, they motivate us to look deeper into the paradigm of multi-system processor design space exploration.

4.2.2 IP-Reuse based Design

IP (intellectual property) based design has been a dominant driving factor in the integrated system design where pre-designed modules are reused as plug-and-play system components [133] [134]. SoC design methodology often uses available IPs to decrease the total design time. These IPs come in different forms: soft, firm and hard [135]. They vary in terms of configurability, soft IPs being fully configurable, while hard-IPs come as non-reconfigurable layouts. Automatic selection of IPs from a library of IPs to match requirements has received attention [136].

An emerging method of hard IP reuse is chiplet assembly. With advancement in interconnect technologies, novel integration schemes are now possible. Interposers [137], EMIB [54], silicon-interconnect fabric [123], and wafer-level-fanout [138] technologies enable high bandwidth, low latency communication between individual chiplets mounted on these interconnects. Recent effort towards building systems have been reported in the works by Schulte *et. al.* [139] and Kannan *et. al.* [140]. Schulte *et. al.* proposed an exascale computing system using chiplet based integration. They argue that such a large system is not possible on one single silicon chip due to yield issues. Smaller high yielding known-good-chiplets can be integrated on highly efficient interconnect at reasonable costs. Also once a chiplet is designed, it can be reused in a multitude of systems. *MoCHI* [141] is another technology to achieve an integration scheme where SoCs can be split into multiple smaller cost-optimized modules and reintegrated without compromising system performance.

In both these cases, the question of which chiplets to design and which systems to construct from them to serve a set of workloads remains unaddressed. In this work, we provide a framework to choose the best *m* chiplets to manufacture and target a wide variety of applications and the optimal set of systems to build from them.

4.3 Motivation

Increasing the number of systems used to target performance and efficiency can have significant benefits. Figure 4.1 shows the benefits across 35 workloads selected from four benchmark suites - SPEC2006 [142], EEMBC [143], SPLASH-2 [144], and NAS Parallel Benchmark [63] - from

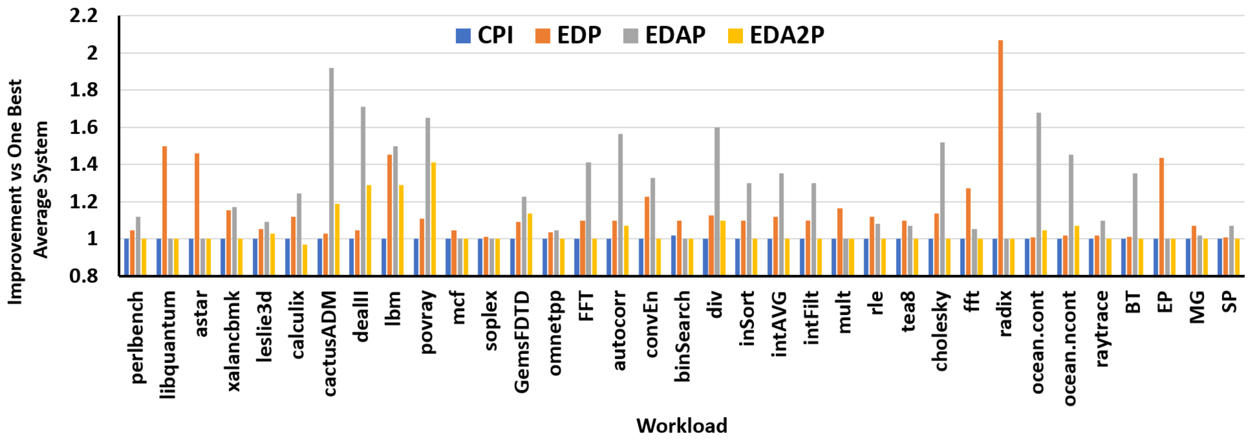


Figure 4.1: Benefit of assigning each workload its best processor vs a processor optimized across all workloads.

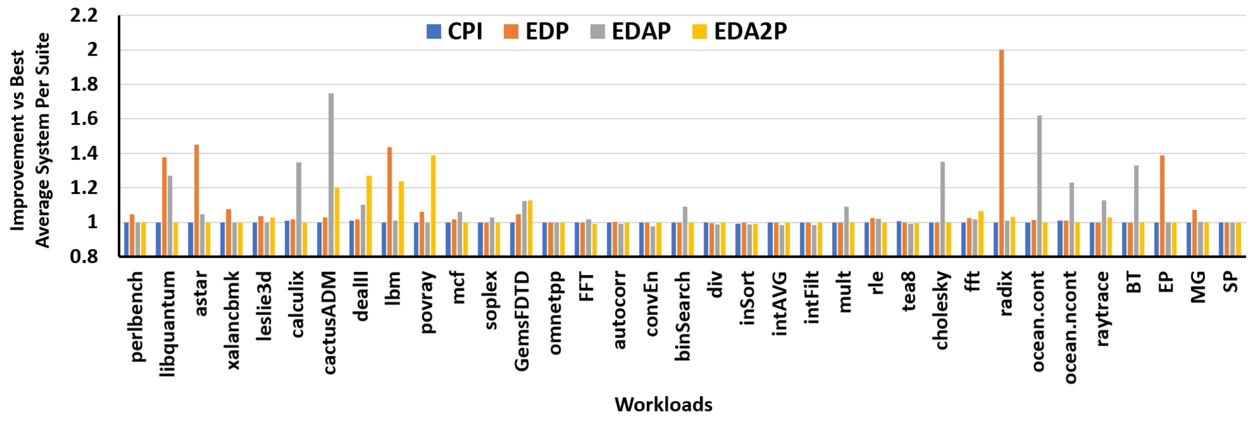


Figure 4.2: Benefit of assigning four processors optimized across all workloads vs four processors each optimized for one of the four benchmark suites.

assigning to each workload the best system available from a design space of 652 systems with private L1s, private L2s and containing both in-order and out-of-order cores. The best system chosen for each workload depends, of course, on the metric. Benefits are normalized to a single system that performed the best across *all* workloads for that metric. As expected, without constraints on power or area, CPI is minimized by selecting only one system, the largest issue width out-of-order core with the biggest caches. For other metrics, while a selection of one system per metric covers a large fraction of the workloads within 10% of optimal performance, there are significant improvements which can be achieved by picking a custom system per workload. For example, *radix* shows a 2.1x EDP improvement compared to the best average system, *cactusADM* shows a 1.9x improvement in EDAP[1] and *povray* shows a 1.6x improvement in EDA^{2}P. As such, an effective exploration of the processor design space to allow selection of the (near) best system for a given workload could be very beneficial.

Since there is typically smaller variation within a single benchmark suite than across suites, one may be inclined to subset the workloads into their respective benchmark suites and select a system for each suite. Unfortunately, with this strategy there are still certain outlier applications which are inadequately covered for some metric (Figure 4.2). This observation is consistent with prior works [131], which suggested that in the context of heterogeneous multi-cores, after selecting five cores to service average workloads, outlier workloads dominate core selection. When four systems are selected per metric using our optimization algorithm (section 4.4.1), all of the outlier workloads are covered within 10% of their optimal performance. Therefore, a systematic selection method is necessary to maximize the benefits of having multiple systems while minimizing the number of chiplets required.

Reducing the number of chiplets, processors, or processor component IPs can reduce the design, manufacturing, assembly, and management costs. Section 4.4.1 describes how we formulate the different multi-system processor optimization problems under different constraints for the number

[1]"The metrics of Energy-Delay-Area^2 Product (EDA^{2}P) and Energy-Delay-Area Product (EDAP) are examples of comprehensive metrics that consider both performance and cost, including both the operational cost (energy) and the capital cost (area). While a chip vendor may favor EDA^{2}P as area^2 provides an approximation to die cost in practice, a system vendor could prefer EDAP as other fixed system costs such as memory and I/O reduce the overall system cost dependence on chip multiprocessor cost. The new metrics are shown to reveal new design sweet spots that cannot be found using other current metrics." [60]

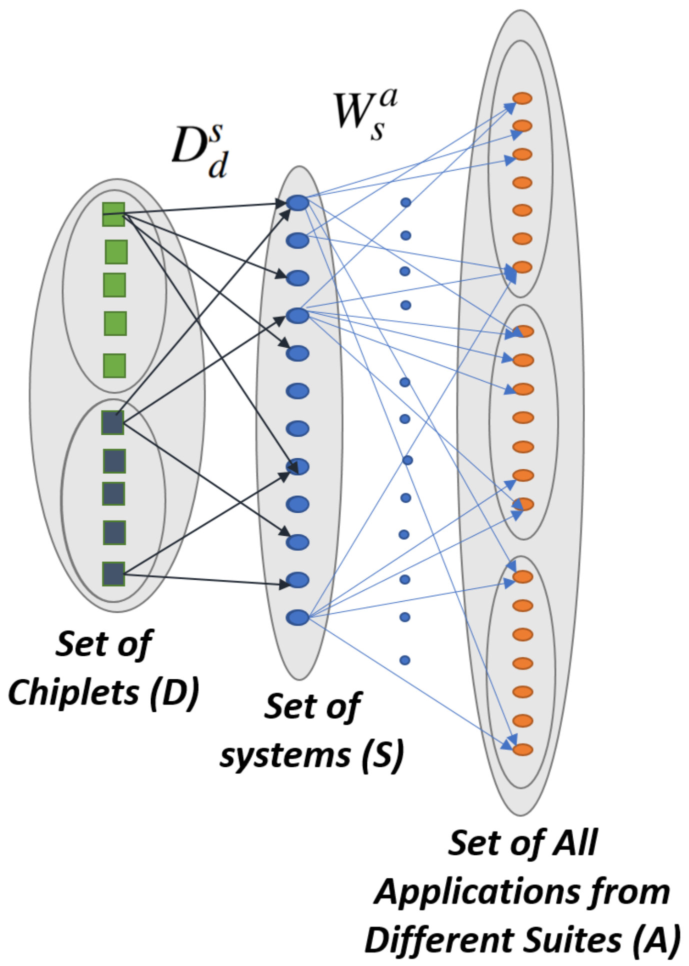


Figure 4.3: Illustration of the multi-chiplet selection problem.

of processors, processor component IPs, or chiplets.

4.4 Optimal Selection of Chiplets

Design-space exploration (DSE) of single core [124, 125], multicore [126, 129, 145] and heterogeneous multicore [130, 131] processors has received a lot of attention in the past decades. However, in all these works, the goal is to find the one, best performing processor, instead of a selection of processors. The potential gains from a multi-system approach, albeit in context of media processors, have been discussed previously [125, 132]. Our work identifies the systems, constituent chiplets and technologies to build them in order to deliver adequate per-application performance.

In this section we describe a framework to select the *m* chiplets and the *k* systems to be built with

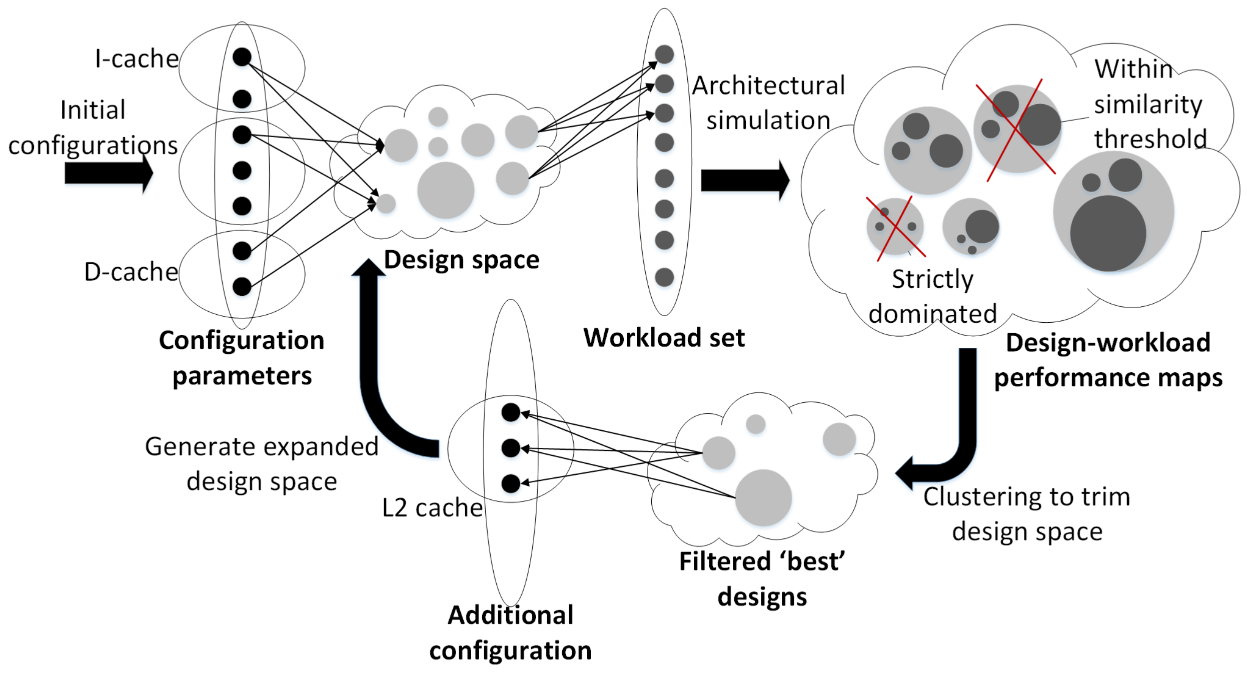


Figure 4.4: Experimental methodology for design space generation.

these chiplets with the objective of either optimizing the EDP of the applications, or minimizing total cost of the systems. In our experiments, we considered a system is composed of core+L1 and L2 chiplets; given a performance estimator, other chiplets with different functionality (L3, accelerator etc.) can also be considered in the framework. Here, we assumed that all chiplets would use standardized interface (physical layer, PHY) protocols to communicate with each other (e.g., Intel's AIB protocol [113]). These protocols can usually be extended to support any width of the interface and adds only one additional cycle of latency to the interface. The chiplets can still use the same higher level protocols (e.g., packetization schemes) as in an SoC implementation.

When $k = 1$ (systems) or $m = 2$ (chiplets), the problem becomes the conventional design space exploration problem. The nature of the problem changes for $m > 2$ or $k > 1$. For n possible systems, the search space increases from $O(n)$ to $O(2^n)$ going from single system to multi-system DSE. First, we describe an integer linear programming (IntLP) framework to optimally solve the multi-chiplet selection problem. When the number of chiplets/systems/workloads are large ($>>$ 10,000), the IntLP formulation can become computationally challenging where well known heuristic methods can be used (e.g., LP relaxations). We focus on optimal solutions as, for our problem sizes, IntLP can be solved by commercial solvers [146] in few minutes. Furthermore, formulating the problem as an IntLP gives us immense flexibility in setting up a variety of constraints and objectives within the same optimization framework as would be illustrated by the experiments in the subsequent sections. The notations used in this section are described in Table 4.1. First, in Section 4.4.1 we describe an integer linear programming (IntLP) framework to optimally solve the multi-system/chiplet selection problem. Our chiplet assembly cost model used for cost-aware optimization is described next in Section 4.4.2.

4.4.1 IntLP DSE Framework

One can visualize the problem as shown in Figure 4.3. The micro-architecture design space is used to constitute the set of chiplets and systems. For example, in our design space exploration with single core systems, a system is considered to be composed of two chiplets, core+L1 and L2. The initial set of micro-architectural parameters, such as size and associativity for L1 and L2 caches and size, type, execution units, etc. for the core, could be used to determine the set of chiplets and systems in the optimization problem. Here, the D_d^s edges indicate which chiplets are part of a system. Multiple copies of the same chiplet microarchitecture may be included in the initial set of the chiplets (D), representing different technology nodes. The set of system configurations (S) is the set of all systems that can be composed out of the chiplets. Each application can be assigned to run on any system and the cost (n-tuple containing the power, performance in cycles-per-instruction (CPI), energy-per-instruction (EPI), energy delay product (EDP) and total cost) associated with that is given by W_s^a. In absence of chiplet constraints, the multi-system selection problem is related to minimum maximal bipartite matching.

For a given set of chiplets and applications, we formulate the IntLP optimization as follows:

Table 4.1: Notations

Notation	**Meaning**
S	Initial Set of all systems under consideration
A	Set of applications
D	Set of all chiplets under consideration
V_a	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 the 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, $p \in [1, n]$
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^t	0 or 1 indicating where chiplet d manufactured in technology node t is present in the final set of chiplets
T	Set of technology nodes in exploration
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:

$$\sum_{s \in S} x_s^a = 1, \qquad a \in A \tag{4.1a}$$

$$\sum_{s \in S} x_s^a * W_s^a.N_p \leq \chi_{ap}, \; p \in [1,n]; a \in A \tag{4.1b}$$

$$\sum_{s \in S} x_s^a * c_s \leq \alpha_a, \qquad a \in A \tag{4.1c}$$

$$x_s^a \leq x_s, \qquad s \in S; a \in A \tag{4.1d}$$

$$\sum_{a \in A} x_s^a \geq x_s, \qquad s \in S \tag{4.1e}$$

$$\sum_{s \in S} x_s \leq K \tag{4.1f}$$

$$y_d^t \geq D_d^s * x_s \qquad s \in S; d \in D; t \in T \tag{4.1g}$$

$$\sum_{s \in S} D_d^s * x_s \geq y_d^t \qquad d \in D; t \in T \tag{4.1h}$$

$$\sum_{d \in D, t \in T} y_d^t \leq L \tag{4.1i}$$

The objective is to minimize the sum of a normalized metric, such as EDP, or total cost, over all application-system assignments. Each metric is normalized for each application with respect to the best possible custom system for that application. Note that other objective functions such as maximizing performance, or different objectives for different applications can also be used.

Constraint (4.1a) assigns exactly one system to every workload. Constraint (4.1b) is used to ensure that for metric p, the system k chosen for an application a is better than χ_{ap} threshold. We primarily use it to enforce a minimum normalized CPI or EDP constraint in our optimizations. Constraint (4.1c) enforces application-specific area (or power) constraints (for example for low-cost or thermally constrained systems). Constraint (4.1d) ensures that a system selected to service an application is present in the final system set. Constraint (4.1e) guarantees that every system in the final set services at least one application. Constraint (4.1f) is an optional constraint which upper bounds the total number of systems selected to k. Constraints (4.1g) and (4.1h) ensure that a chiplet is chosen if and only if it is part of a chosen system. Constraint (4.1i) is optional and puts a cap on the maximum number of chiplets allowed.

4.4.2 Chiplet Assembly Cost Model

Cost brings in an interesting dimension to the multi-chiplet selection problem. Migrating to an advanced technology node provides benefits in terms of area and power but increases design and manufacturing costs. Cost of a chiplet can be broken down into two major components: non-recurring engineering (NRE) cost and volume-dependent recurring cost. NRE costs include architecture, RTL design, IP validation, physical design, prototype, validation, and mask manufacturing cost. In our cost model, we consider the cost difference between logic and SRAM as well. Due to its regular structure, memory design is typically less expensive than logic of the same size. Moreover, SRAMs typically use fewer metal layers resulting in lower manufacturing costs as well.

We assume the recurring cost to be the cost of wafer fabrication. Yield and process complexity are the primary factors which determine the fabrication cost. Advanced technology nodes require larger number of process steps on more expensive equipment, increasing manufacturing costs.

We use the yield learning model given in [147] to model the yield improvement over time so as to account for process maturity.

$$Y(t) = Y_{asymp} \times (1 - exp(-ct)) \tag{4.2}$$

where $Y(t)$ is the yield at time t after launch of the technology node. We considered 50% yield at the time of process node introduction and that the yield reaches 95% when t=2 years. Y_{asymp} is the asymptotic yield determined by the die area (A_{die}), defect density (D_0), clustering factor (α) and is given by the well known negative binomial model

$$Y_{asymp} = \big(1 + \frac{A_{die}D_0}{\alpha}\big)^{-\alpha} \tag{4.3}$$

Recurring cost of manufacturing (C^{mfg}) taking into consideration volume of dies required (V_d), wafer cost (C^w), yield (Y) and number of dies per wafer (N_{die}) is given by

$$C^{mfg} = \frac{V_d \times C^w}{N_{die} \times Yield} \tag{4.4}$$

We use the following objective function when total cost minimization is considered for chiplet based assemblies:

$$\begin{aligned} &\sum_{a \in A} (V_a \times (\sum_{s \in S} x_s^a (C^{int} + \sum_{d \in D} D_d^s * C_d^{mfg}))) \\ &+ (\sum_{d \in D} \sum_{t \in T} y_d^t * NRE_d^t) + \sum_{d \in D} (\vee_{t \in T} y_d^t * NRE_d^{nt}) \end{aligned} \tag{4.5}$$

where V_a is the volume of systems required for application a. C^{int} is the chiplet assembly/integration cost, C_d^{mfg} is the cost of manufacturing of chiplet d and is estimated using Eq. 4.4. NRE_d^t is the technology dependent NRE cost which includes physical design, IP validation, prototype, mask set cost, etc. NRE_d^{nt} is the technology agnostic NRE cost which includes architecture, RTL development and verification etc. When a particular microarchitecture is used to build chiplets in two different technology nodes, NRE_d^{nt} is amortized; however, NRE_d^t expense is required for every technology node the chiplet is built.

The result of the IntLP solution is an optimal set of chiplets, the systems constituted out of the chiplets and the application-system mapping.

4.5 Experimental Setup

4.5.1 Methodology

The methodology for our design space exploration is an iterative process consisting of three main steps (Figure 4.4). First, we identified a set of interesting initial system configurations to explore. This set of configurations (provided in Table 4.3) must be diverse enough so that changing any parameter has a measurable impact on either power, area or performance of the system running an application. In addition, these configurations should vary enough such that their range tightly covers the range of application behavior.

Next, we performed a full factorial exploration of these initial system configurations. To evaluate performance of designs, we used Sniper [62], a fast trace-driven multi-core interval simulator. Sniper is significantly faster than gem5: this is important because the sheer number of simulations required for a full factorial exploration is large. We used the ROB-centric model in Sniper, which more accurately models in-order cores and issue contention. We studied trade-offs related to multi-core systems too. For the performance and energy calculation of multi-core systems considered in section 4.6.2 and heterogeneous multi-cores in section 4.6.8, we made a simplifying assumption that two single-threaded workloads running on separate cores of a dual-core system don't interfere and that their aggregate performance is the summation of the IPCs when each workload is run individually on a single-core. Past works [130] have made such assumptions. However, we also performed an experiment (in section 4.6.9) with homogeneous multi-core system running multi-threaded benchmarks with a shared-L2 cache.

L2 cache access usually takes $\sim$ 10 cycles. For the performance simulations, we assumed that the L2 caches have a 12 cycle access. Changing the latency by one cycle (due to interface protocol circuitry) resulted in a negligible performance difference (worst case $<3\%$) as most of the memory accesses from the cores are served by the L1 cache which lies in the same chiplet as of the core. Therefore, we used the same performance numbers for SoC and chiplet in order to avoid duplicate simulation runs.

For area and power modelling, we modeled 22nm, 32nm and 45nm processes using McPat 1.0 [60], which is deeply integrated with Sniper. To calculate average power and energy for our primary results, we excluded DRAM energy. 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 similar to [148] 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 added 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 used the optimization algorithm described in Section 4.4.1 to select the best chiplets

Table 4.2: Workloads Evaluated

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

and systems to cover the given workloads.

Runtime: The total runtime of an IntLP instance lies between 1-4 min depending on the contraints while a Sniper simulation run for a particular system-workload combination takes about 20-30 minutes.

4.5.2 Workloads Evaluated

A diverse set of workloads were chosen by subsetting 4 benchmark suites - SPEC2006 [142], EEMBC [143], SPLASH-2 [144], and NAS Parallel Benchmark (NPB) [63]. We chose 14 applications from SPEC2006 based on [149] in-order to demonstrate the diversity of the suite without capturing redundant results between benchmarks. SPEC2006 applications were simulated using 100M-warmup, 30M-detail Pinballs [150] 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. A complete list of chosen benchmarks is shown in Table 4.2.

Table 4.3: System parameters and their values

Issue-width	1, 2, 4	**ROB (OOO)**	32, 64 entries
I-Cache	8KB DM, 16 KB 2-way, 32KB 4-way, 64KB 4-way	**L2 Cache**	256KB, 1-way, 512KB, 2-way, 1MB, 4-way, 2MB, 8-way, all 12 cycle access
D-Cache	8KB DM, 16KB 2-way, 32KB 4-way, 64KB 4-way dual ported	**Memory Channel**	533 MHz, doubly-pumped, RDRAM
Execution Units	3, 6	**ITLB-DTLB**	64-28 entries
IQ (OOO)	48, 96	**Ld/St Queue**	32 entries

4.5.3 Systems and Chiplets Evaluated

We relied on previous work [130] to make our initial design space selection with a few differences in order to more accurately model the systems in Sniper which is based on a Nehalem-like architecture [151]. For instance, Sniper does not model the number of functional units, but instead multi-purpose instruction ports, which can be used to service a subset of instructions. We provide configurations for both a traditional Nehalem architecture and one with double the number of instruction ports. Additionally, due to a lack of accurate functional modelling in Sniper, we did not consider different physical register file sizes.

For this work, we considered that a single-core processor system is comprised of two chiplets: core with L1 cache (core+L1 chiplet) and L2 chiplet. We considered the interconnect characteristics to match those of the state-of-the-art chiplet assembly approach presented in [123]. 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 performed simulations on all core (includes L1 caches) configurations, with a constant 1MB L2 cache. We fully explored these configurations and then trimmed the design space to a smaller set using the clustering approach described earlier. We then simulated 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 used these results to generate power, energy and area results for chiplets implemented in 22nm, 32nm and 45nm 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 considered systems built out of heterogeneous technology chiplets for our evaluations. In total, we considered a total of 5,868 system configurations in our homogeneous-core (single and multi-core) studies and 10,404 system configurations for the heterogeneous multi-core case.

4.5.4 Cost Components and Volume

The cost model used in our formulation has been presented in Section 4.4.2. The various cost components used in the cost model for the evaluations are shown in Table 4.4 [152] [153] [154]. Note that for SRAM, the design cost was considered to be the cost of a memory compiler license and a few engineers. SRAM chiplets would also usually use only 5-6 BEOL layers as compared to 10-12 metal layers for logic (both use all the FEOL layers). Therefore, we scaled the mask and recurring wafer cost accordingly.

We considered the chiplets to be integrated on 2.5D substrates such as silicon interposers [120], silicon interconnect fabric [123] or EMIB [54]. On these substrates, the chiplets can have an inter-chiplet spacing of <1 mm (even <100 μm). Also because of fine pitch interconnections, the interfaces are usually wide and signalling is done at frequencies of 2-4 GHz, therefore the I/Os driving these substrate wires can be simple multi-stage buffers (with small ESD circuitry) [54, 123] which are area and energy efficient. Therefore, considering $\sim$2000 IOs between a core and a cache chiplet, we added an overhead of 0.5 mm^2 (conservative estimate) to the chiplet areas.

The cost of integration and bonding i.e, C^{int}, was assumed to be similar to the data in [155]. We also performed sensitivity analysis with 2x and 4x the assembly cost, because in reality, the cost of 2.5D have been very high and is in fact used for a niche class of high performance processors only. We used the ITRS data [156] for D_0 and α while calculating the yield. To estimate the per chip vendor volume per year, we used the global estimates of phone and tablet sales (180M units per vendor) [157] for the EEMBC suite, aggregate PC and laptop sales (90M units per vendor) [158] for SPLASH and SPEC suites combined and global x86 server sales (10M units per vendor) for the NPB suite [159]. We divided the total volume in each suite by the number of workloads in

Table 4.4: Normalized Cost components at different Technology Nodes

Cost Type		Technology Node		
		22 nm	32 nm	45 nm
NRE Design Cost [152]	Logic (per mm^2)	849	502	332
	SRAM	965	695	483
NRE Mask Set Cost [153]	Logic	695	541	309
	SRAM	486	378	216
Recurring Wafer Cost [154]	Logic	2.43	2.01	1.41
	SRAM	1.70	1.41	1
Chiplet Assembly Cost [155]		0.25		

each suite to uniformly distribute volume demand among individual workloads. The cost analysis is done for different sizes of the system. System size of Nx (in Figure 4.6) is the one where N copies of core+L1 chiplet and L2 chiplet constitute a system. For such multi-core systems, the NRE design cost is considered for only one copy of the core. As an example, for a system with N ARM based cores connected using AXI interconnect, the same core IP is replicated multiple times and connected to the scalable AXI interconnect IP. Therefore, the design cost incurred would be roughly similar to that of a single core system with the interconnect. For performance and energy simulations, we consider that N instances of the same workload are executed on the N-core system. The cost optimizations are also performed for different production start years to understand the impact of process maturity on cost. As processes mature, yield increases and this brings down the manufacturing cost per good chiplet.

Note that for some of the experiments where the goal is to capture both the energy and area cost, we used EDAP and EDA^2P as examples of composite metrics. This also shows that the framework is flexible and can be used in a variety of contexts.

4.6 Results

Here we discuss the characteristics of the chiplets, their usage and sharing across workloads and benchmark suites, cost implications etc. Note that we allow chiplets to be selected from different technology nodes in our optimizations. Also, the results shown here are for a small but interesting set of optimization objectives and constraints; however, the IntLP framework is flexible enough such that one can study many different objective and constraint combinations.

4.6.1 Minimizing EDP with CPI Constraint

First, we consider EDP minimization as the objective with constraints on maximum CPI and number of unique chiplets. A CPI threshold *t* ensures that any selected system-workload pair have CPI less than *t* times the minimum CPI for that workload which can be obtained on the best performing system. As shown in Figure 4.5, having more chiplets available allows significant reduction in average EDP. Initially the benefit from adding more chiplets is large, as the first few systems selected out of the chiplets target broad workload classes such as memory-bound or compute-bound applications. As more chiplets are added, new systems singularly target outlier workloads, resulting in incremental average improvement. When the CPI threshold is very strict, the benefit quickly saturates since only a subset of systems are available for selection. However, for more relaxed CPI thresholds, 7-8 chiplets are required to attain near optimal EDP. Since migrating to a newer technology node results in reduced EDP, all the chiplets selected in this cost-agnostic optimization were exclusively in 22nm technology. Using multiple chiplets can have greater than 35% reduction in EDP over a single average best system (m=2).

Chiplet Micro-Architecture: CPI threshold of 1.1 requires a minimum of 4 chiplets to be feasible. When only two unique chiplets are allowed (i.e., an average best system) for CPI thresholds of 1.2, 1.5 and 2.0, the same medium out-of-order core+L1 chiplet gets selected. However, the size of the selected L2 cache varies from 2MB to 1MB to 512KB, respectively, for the three threshold values. When 3 chiplets are allowed, an additional cache gets added in all three cases (the smaller 512KB for 1.2 and 1.5 CPI thresholds and the bigger 2MB for the 2.0 case). This shows that cache variety has a major impact on EDP.

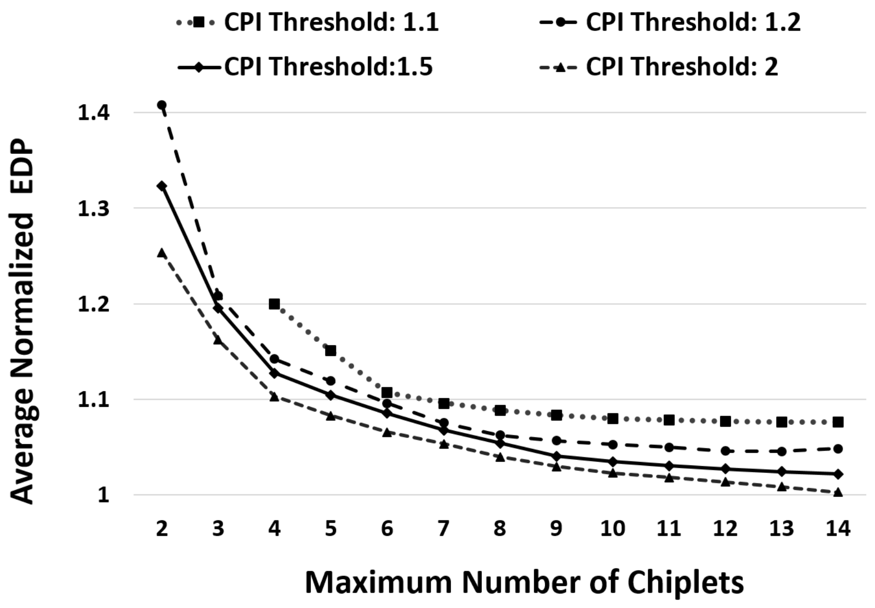


Figure 4.5: Pareto curve showing average normalized EDP vs maximum number of chiplets allowed in the optimization. We show the pareto curves for different CPI thresholds.

When four unique chiplets are allowed for CPI thresholds of 1.2 and 1.5, a smaller core chiplet with small L1 D-cache is added. Only when CPI is relaxed to 2.0, one in-order core chiplet gets selected. With four chiplets, CPI threshold of 1.1 now becomes feasible and requires 1MB and 2MB L2 caches alongside two core+L1 chiplets, in which the core components are same but the L1 caches are large (64KB) and small (16KB), respectively. Initially when chiplets are added, having a variety of L2 cache sizes minimizes EDP. With more chiplet variety, having large and small cores is more essential.

4.6.2 SoC vs. Chiplet Assembly Cost Analysis

When systems are implemented via chiplet assembly rather than IP integration on an SoC, use of smaller chiplets results in higher yield and thus reduces cost. We perform design space exploration with total cost minimization objective with EDP threshold constraint. *Chiplet based assembly provides substantial cost benefit over SoC approach* as shown in Figure 4.6. When the system size is small, SoC yield is good. This, along with the added cost of chiplet integration, results in a meager 10-14% cost benefit over a single-core SoC system when the integration cost is considered to be the same as in [155]. However, in multi-core systems as the system size increases, the SoC yield decreases according to Eq. 4.3. Because of the negative binomial nature of the yield curve, beyond a certain die size, the yield decreases rapidly. As a result, in 4.6 we observe that going from a single core to a dual core system, SoC integration results in per-core cost improvement, however, increased yield issues result in increased cost as the system is scaled to a 4-core and 8-core system. Using chiplet assembly, copies of the same core+L1 and L2 chiplets are utilized and the yield of these smaller dies remains the same. Moreover, the NRE cost of developing these chiplets gets amortized as the system size grows. Though the interconnect substrate and chiplet assembly cost grows with the size of the system, it is a smaller fraction of the overall cost. Therefore, this results in reduced total cost per core+L1 and L2 chiplets using chiplet based assembly despite a slight increase in system integration cost. We confirmed this trend by running our optimization across several EDP thresholds.

Apart from yield benefits, fewer chiplets are needed to assemble a large number of systems. The fact that relatively few unique chiplets can be mixed-and-matched to serve a wide array of applications bodes well for an era of customizable systems using chiplet-assembly approach. In today's era of multi-core processors, chiplet assembly can provide significant cost benefits, which will only improve as processors become larger and core count continues to increase.

Sensitivity to assembly cost: As mentioned earlier, there hasn't been a large improvement in 2.5D integration cost and the overall cost remains quite high today. As seen in Figure 4.6, when the integration cost is higher (4x assembly cost), the gap between SoC and chiplet assembly becomes smaller. In fact, when the system size is small, chiplet assembly can be costlier than SoC. As the technology matures, the cost of SoC drops but integration cost doesn't drop much and therefore, larger dual-core (2x) systems have smaller (or diminished) gap in cost between SoC and chiplet assembly. However, when the system is even larger (4x or 8x), the worsened yield of larger SoC dies leads to rapid increase in SoC costs while even with 4x assembly costs, chiplet assembly based systems come out ahead by as much 23%.

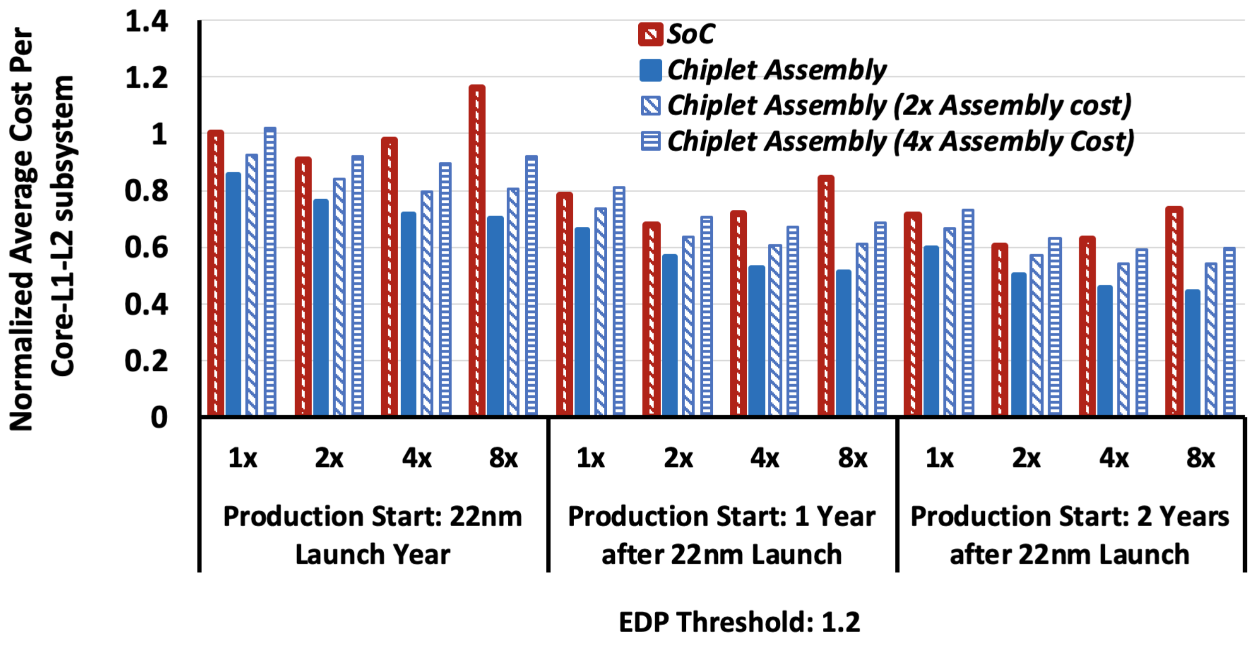


Figure 4.6: Curve showing cost benefits of chiplet based assembly over SoC for different sizes of the system and technology maturity level.

4.6.3 More Applications Share Chiplets than Share Systems

Systems can share chiplets and, hence, m chiplets allow us to construct $>> m$ systems. Systems serving different applications can share the same chiplets. We observe an example of chiplet sharing with the L2 caches. EEMBC workloads have smaller working sets, so they tend to use smaller L2 caches. However, a few workloads from SPEC and SPLASH-2 which also have smaller working sets can benefit from sharing these smaller L2 caches. We observe that when a large number of chiplets are available, most chiplets are shared across benchmarks suites. The degree of reuse is much higher in this case than that when only system/SoC is considered because once the number of available systems increases, quite a few systems get selected that only cater to specific workloads from one workload suite.

4.6.4 EDP-Cost Trade-off

Harsher EDP constraints (low threshold value) require more systems to minimize costs, essentially trimming the design space for each workload so that fewer systems are available for selection. This

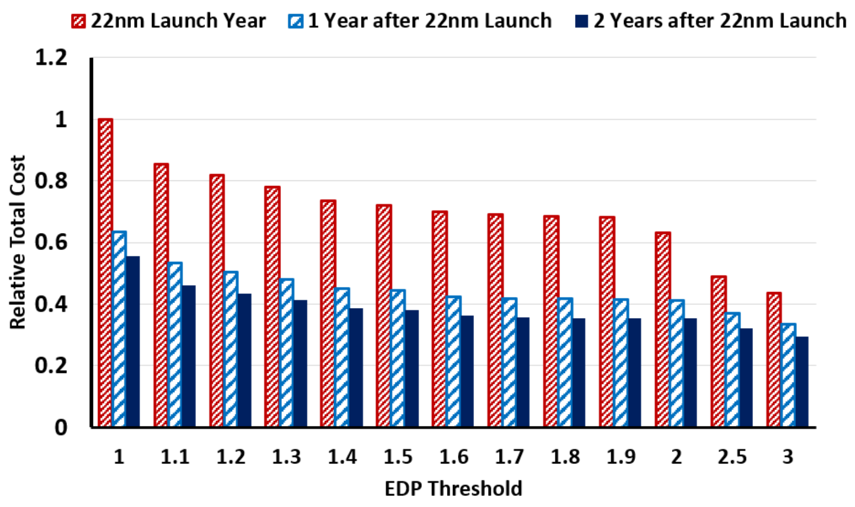


Figure 4.7: Curve showing trade-off for EDP and total cost at different technology maturity level.

leads to excessive customization with little sharing of systems across workloads. Hence, the number of distinct chiplets required to build these systems increases, which results in higher NRE costs (shown in Figure 4.7). As the EDP threshold constraint is relaxed, the optimization begins to select smaller cores and smaller L2 chiplets. Overall, fewer chiplets and systems are selected, increasing sharing across workloads. Increased sharing helps amortize NRE costs, resulting in reduced total costs. In fact, when EDP threshold is low, the cost difference is much more dramatic (>34% for threshold of 1.1) between production start during 22nm launch year and subsequent years than higher thresholds (>16% for threshold of 2.5).

For a particular total cost budget, as technology matures, one can achieve better overall EDP. This is because more chiplets, and hence more systems tailored towards particular set of applications can be built for the same total cost when technology matures.

4.6.5 Impact of Technology Maturity

Technology selection of chiplets is a key factor which determines the overall cost of designing and fabricating multiple chiplets. Migrating to an advanced technology node reduces power and therefore EDP. Additionally, area per transistor decreases, leading to reduced chiplet size. However, lower wafer yield and higher NRE and manufacturing costs may outstrip the benefits of accommodating more chiplets per wafer in an advanced technology node. These costs are much higher during the early stages of a technology node. Figure 4.6 shows how total cost varies with production start year. Our evaluations with 22nm, 32nm and 45nm technology show that although it is attractive to migrate to the 22nm node because of the EDP and area benefits, a few chiplets are still chosen from 32nm technology during the early stages of 22nm node. However, it is interesting to notice that more core+L1 chiplets are chosen from 32nm technology, than L2 chiplets. This is because the SRAM L2 chiplets are heavily shared across multiple workload suites and thus the high volume amortizes the NRE cost of the SRAM dies.

When the 22nm technology matures (yield increases) and the fabrication cost decreases, building chiplets in 22nm becomes attractive. Thus, all the chiplets are chosen from 22nm technology when the production start is 1 or more years after 22nm launch. In Figure 4.6, depending on production start, up to 72% cost reduction can be achieved by chiplet assembly over SoC methodology while satisfying the EDP constraints per workload. Chiplet assembly's continuing cost benefits makes it the ideal choice for building systems irrespective of production start year. Moreover, the cost benefits of chiplet assembly, especially through technology heterogeneity, are likely to be even higher as technology scales below 22nm where manufacturing NRE costs are much higher (e.g., due to extensive use of multiple patterning lithography).

4.6.6 Minimizing Number of Chiplets vs. Cost

Here we show that optimization with the goal of minimizing the number of chiplets doesn't necessarily result in minimum total cost. As shown in Figure 4.8, when EDP threshold is relaxed, chiplet minimization results in smaller number of chiplets than when cost is minimized. However, the total cost of design and manufacturing remains much higher than when cost minimization is the objective. This is because chiplet minimization chooses fewer but larger chiplets (lower yield and costly) to satisfy the EDP constraints of all the workloads while cost minimization chooses more number of smaller (less costlier) chiplets to build multiple systems, each tailored towards different types of workloads. For example, when EDP threshold is 2.0, cost minimization results in one in-order, one small OOO, 256KB and 512KB L2 caches, while chiplet minimization returns

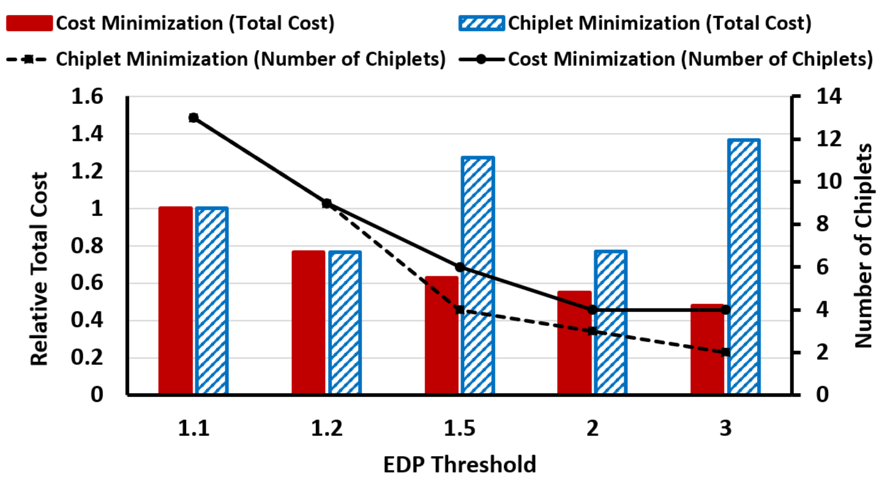


Figure 4.8: Curve showing the difference between minimizing the number of chiplets versus total cost. The experiments were done for system size of 1x and production during 22nm launch year.

two large OOO core and 512KB L2 cache.

4.6.7 Efficacy of Proposed Optimization Framework

The IntLP framework solves the multi-system, multi-chiplet selection problem optimally. Here, we briefly compare our optimisation to two other naive selection algorithms (as shown in Figure 4.9). All selection algorithms are evaluated with a CPI constraint of 1.5, normalized to the best performance of each workload across all systems. 'Best Average System' selects the system with the best average for a single metric which meets the CPI constraint across all workloads. 'Greedy 8' selects first the best system which meets the CPI constraint across all workloads, then iteratively selects the next system which improves the metric, only using the new system to cover workloads for which it meets the CPI constraint. Our IntLP-based optimization framework can be up to 65% (EDA^{2}P) better than the naive heuristics.

4.6.8 Optimization for Homogeneous Multi-cores

Our methods of finding optimal set of chiplets can be applied directly to homogeneous multi-cores by considering, for example, shared L2 as a chiplet, multiple core+L1 chiplets in a system and

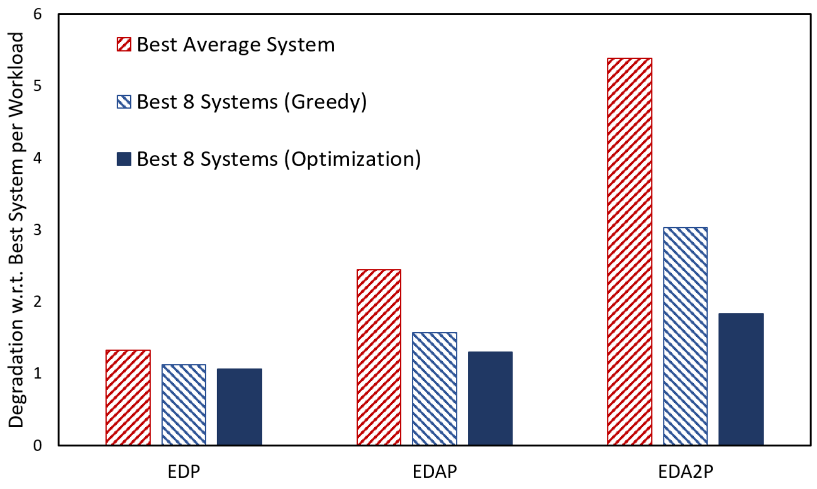


Figure 4.9: Comparison between different system selection approaches for each metric when compared to the baseline case of best custom system per workload.

multi-threaded benchmarks as workloads. For an evaluation with the SPLASH-2 suite, we find that 5 systems are still required to minimize metrics such as EDAP and EDA^{2}P for CPI-constrained systems. As observed in the previous experiments (without shared L2 cache), the set of final systems included a mix of system sizes, from small inorder cores to medium sized OoO cores to big OoO cores with a large cache. However, the progression towards a smooth gradient in system sizes that we observed in those system sets is not present in multi-core system selection. Instead, we observed that an optimal system set contains a system with large OoO cores with a large shared cache and a set of systems with much smaller L2 shared cache size but a variety of smaller core sizes. This is because the larger system caters to the workloads with large working sets which otherwise would observe degraded performance if they are run on processors with smaller caches as interference between threads increases. We also observe that for such system, the positive effect of L1 cache sizes on system performance is diminished because of coherence overhead and false sharing.

Because of the disproportionate impact of L2 size on system performance, when selecting homogeneous shared-L2 multi-core systems based on chiplet sharing, we see a very similar pattern of micro-architectures as when disregarding chiplets (i.e., SoC implementation). This suggests that the optimal system configurations for these homogeneous multi-core systems are not as flexible as single-core systems. That is, the differences in relative performance across optimal systems for these workloads is large enough such that substituting a closely related chiplet for cost reasons will substantially degrade performance. While the sensitivity of workload performance to L2 size specifically may be an intrinsic property of SPLASH-2, we conclude that shared chiplets (L2 in this case) such as those present in a multi-core system may have an especially large impact on the exact choice of chiplet micro-architectures.

4.6.9 Optimization of Heterogeneous CMP Systems

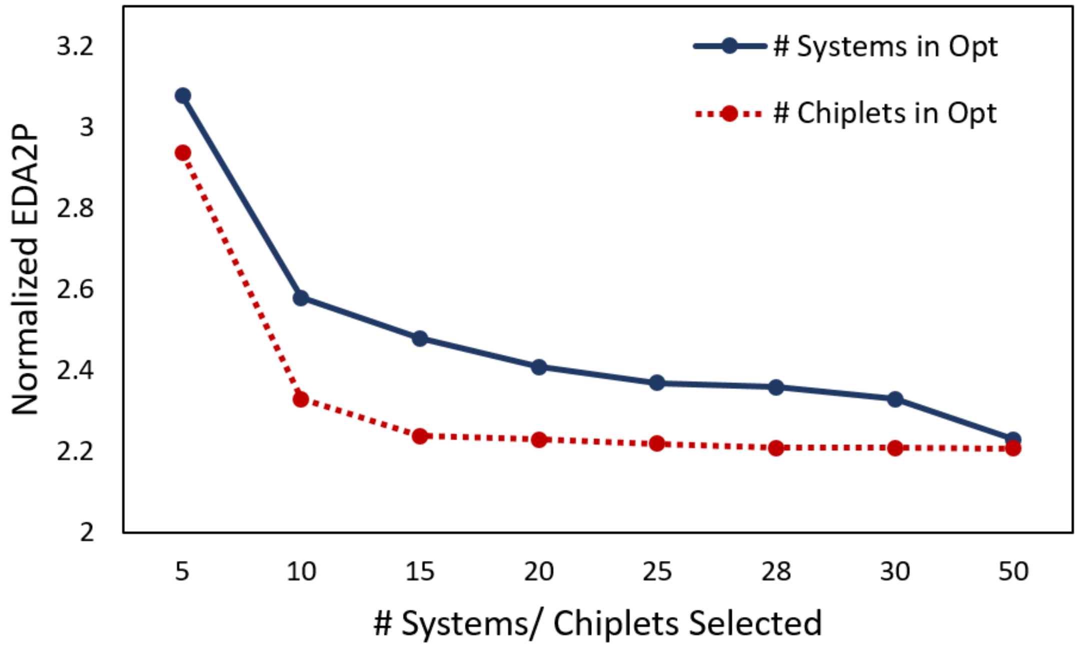


Figure 4.10: Benefit in EDA^{2}P (normalized CPI threshold = 1.2) for Heterogeneous CMP is shown for the cases where number of systems (SoCs) is taken into consideration versus when number of chiplets is considered in the optimization.

Modern processor systems are increasingly heterogeneous, incorporating a variety of core types, memories and accelerators. Our system/chiplet selection framework is also easily adapted to heterogeneous settings by essentially enumerating candidate heterogeneous systems and application sets within the framework. Of course, we realize that computational scalability may become a concern here but we believe our clustering-based design-space pruning coupled with a smart initial design of experiments can provide a solution. Also to speedup the performance runs, if one has parameterizable RTL, FPGA emulation can be used which could potentially lead to 100x or more speedup compared to software simulation. As an example, we studied heterogeneous dual-core (i.e., big-Little architecture) systems where each core runs a SPEC2006 benchmark application. Therefore in this study, a total of 276 application pairs were considered. Figure 4.10 shows the chiplet and system (SoC) exploration results. We make two observations. First, due to the dramatically increased diversity in workloads (pair of applications) to be run on a single dual-core system, the number of unique systems (composed out of chiplets) needed to achieve good performance is almost 10X higher compared to the single core case. Second, the number of chiplets required to achieve the peak performance is significantly less than the number of distinct SoCs required to achieve the same metric. For example, more than 50 dual-core SoCs are required to achieve the best EDP with a CPI constraint of 1.2x. However, similar EDP requires less than 23 chiplets because a majority of chiplets are shared across the systems. This shows that chiplet based assembly could be leveraged to generate a large number of these systems by manufacturing only a few types of chiplets. Thus, finding the optimal set of chiplets is even more important in this context.

4.6.10 Optimization with Heterogeneous Constraints

A likely common case for multi-system/chiplet optimization is when constraints and/or objectives for different application domains are very different. Our optimization framework can be constrained in almost arbitrary ways or setup different optimization objectives for different workloads or workload suites. As an example, embedded workloads may be severely cost-limited while servers are performance-limited. Figure 4.11 considers one such case and compares the results with all applications uniformly constrained by a server performance constraint. One can see the dramatic reduction in overall cost-based metrics (EDAP, EDA^{2}P) as a result. EEMBC workloads are now mapped to much smaller cores with worse CPI but better EDP while SPLASH-2 and NPB continue to use bulkier systems.

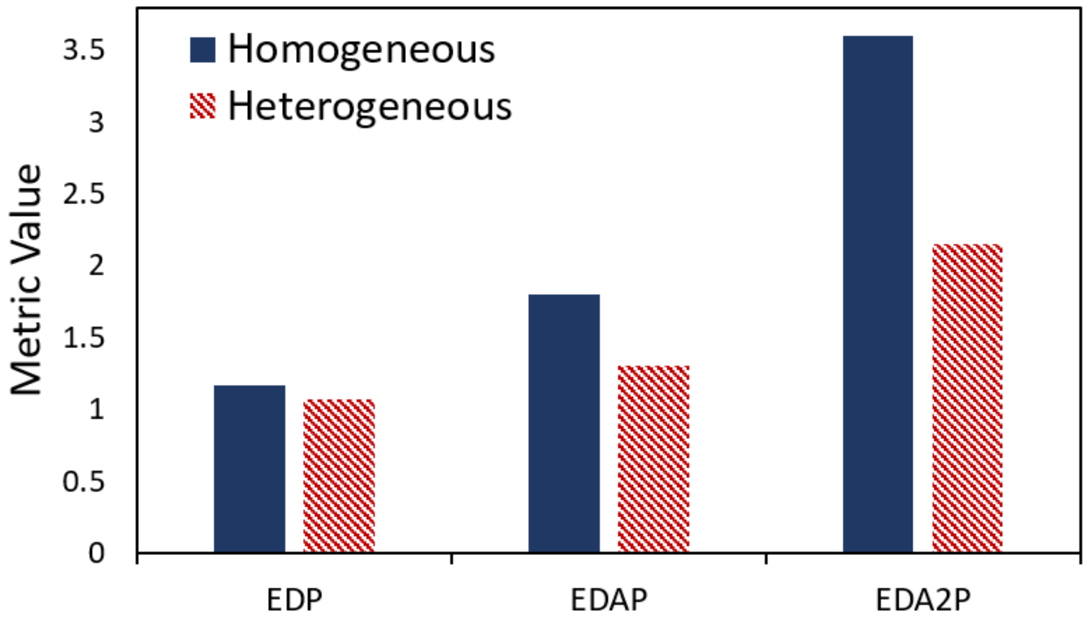


Figure 4.11: Average normalized metric values are shown for homogeneous and heterogeneous constraints in area and CPI for different metrics for 5 systems. For heterogeneous constraints, normalized CPI thresholds are 1.2 for SPLASH-2 and NPB, 1.6 for SPEC and 2.4 for EEMBC. Area thresholds are 15 mm^2 for EEMBC, 25 mm^2 for SPEC and no area constraint for SPLASH and NPB. Homogeneous constraints impose no area threshold and normalized CPI threshold 1.2 for all suites.

4.7 Conclusions

Traditionally, processor design space exploration has been focused on identifying one system that maximized performance or efficiency. However, applications are often targeted using a family of processors, where each processor targets a subset of applications. As building multiple SoCs become costlier, chiplet integration technologies enable cost-effective system customization. In this work, we have developed the first multi-chiplet processor design space exploration framework. Our results clearly show benefits of using multiple chiplet assembly based processors to service diverse workloads (35% improvement in EDP over a single best average system), and advantages of cognizance of chiplets during optimization (over 2x reduction in chiplets required for a heterogeneous CMP example). We also identify the different factors that affect the cost of building multiple systems using chiplets as well as the characteristics of the systems/chiplets chosen under different use contexts. Using the framework we show that emerging chiplet assembly approaches are very promising when total cost of design and manufacturing is considered (up to 72% benefit in cost over SoC) while satisfying energy and performance requirements of every workload. This cost benefit strongly depends on system size and technology maturity.

CHAPTER 5

Architecting Waferscale Processors - A GPU Case Study

Increasing communication overheads are already threatening computer system scaling. One approach to dramatically reduce communication overheads is waferscale processing. However, waferscale processors [160–162] have been historically deemed impractical due to yield issues [36, 160] inherent to conventional integration technology. Emerging integration technologies such as Silicon-Interconnection Fabric (Si-IF) [4, 37, 163], where pre-manufactured dies are directly bonded on to a silicon wafer, may enable one to build a waferscale system without the corresponding yield issues. As such, waferscalar architectures need to be revisited. In this work, we study if it is feasible and useful to build today's architectures at waferscale. Using a waferscale GPU as a case study, we show that while a 300 mm wafer can house about 100 GPU modules (GPM), only a much scaled down GPU architecture with about 40 GPMs can be built when physical concerns are considered. We also study the performance and energy implications of waferscale architectures. We show that waferscale GPUs can provide significant performance and energy efficiency advantages (up to 18.9x speedup and 143x EDP benefit compared against equivalent MCM-GPU based implementation on PCB) without any change in the programming model. We also develop thread scheduling and data placement policies for waferscale GPU architectures. Our policies outperform state-of-art scheduling and data placement policies by up to 2.88x (average 1.4x) and 1.62x (average 1.11x) for 24 GPM and 40 GPM cases respectively. Finally, we build the first Si-IF prototype with interconnected dies. We observe 100% of the inter-die interconnects to be successfully connected in our prototype. Coupled with the high yield reported previously for bonding of dies on Si-IF, this demonstrates the technological readiness for building a waferscale GPU architecture.

5.1 Introduction

With the emergence of new applications [164–168], new business models [169, 170]), and new data processing techniques (e.g., deep learning), the need for parallel hardware has never been stronger [171–173]. Unfortunately, there is an equally strong countervailing trend. Parallel hardware necessitates low overhead communication between different computing nodes. However, the overhead of communication has been increasing at an alarming pace. The area taken by IO circuitry to support chip-to-chip communication already exceeds 25% on some of today's processors [174]. Power overhead of such IO exceeds 30% on some processors. These overheads are expected to be worse in future as communication energy, latency, and bandwidth scale much worse than computation [19].

A large body of recent work targets the communication bottleneck. Focus areas include energy-proportional communication fabrics [175–177], in-memory and near-memory computational models [178–180], communication avoiding algorithms [181], and novel packaging techniques such as 2.5D and 3D integration [51, 182]. We explore a different approach - *waferscale processors*. Conventionally, many copies of a single processor die are manufactured simultaneously on a single wafer - largest size of the die determined by yield. Post-manufacturing, the wafer is diced into individual processor dies which are then packaged and integrated into a parallel system using IO links that connect these packaged processors on a printed circuit board (PCB). In a waferscale processor, on the other hand, the wafer is the processor, i.e., either a monolithic processor is designed to be as large as an entire wafer or a set of processors are designed that continue to reside on the wafer and the processor die are connected on the wafer itself using a low cost, on-wafer interconnect.

A waferscale processor can have considerable advantages over conventional systems in terms of area, performance and energy efficiency. Figure 5.1 shows the total area footprint of the compute dies in multiple scenarios: each die is placed in a discrete package, 4 units (each unit consists of a processor die and two 3D-stacked DRAM dies) inside an multi-chip module (MCM) package and, waferscale integration. The area overhead of packages is usually quite high and for high performance systems, the package to die ratio can be more than 10:1 [38]. While MCM packages help decrease the package footprint per die, packageless waferscale integration would provide substantial benefits

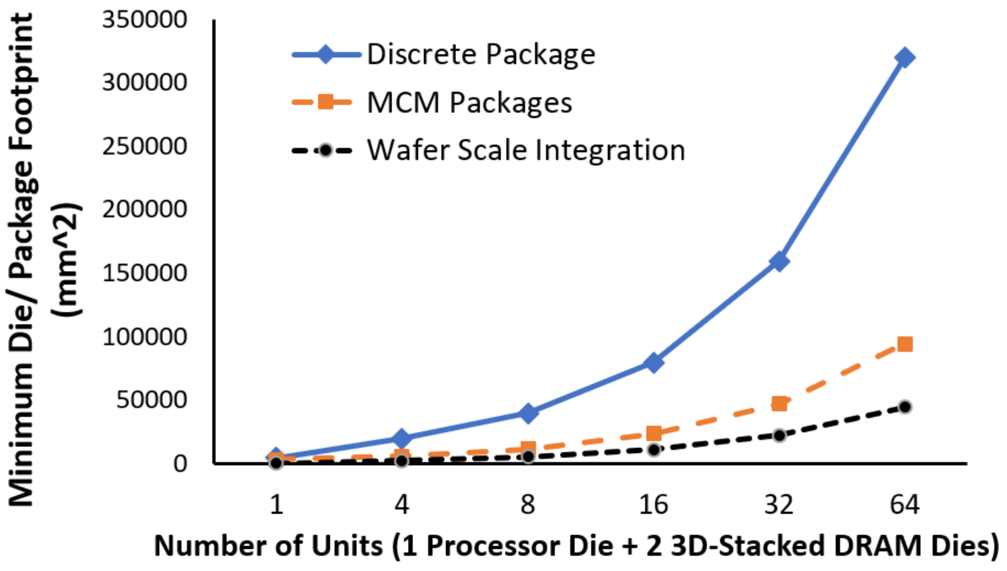
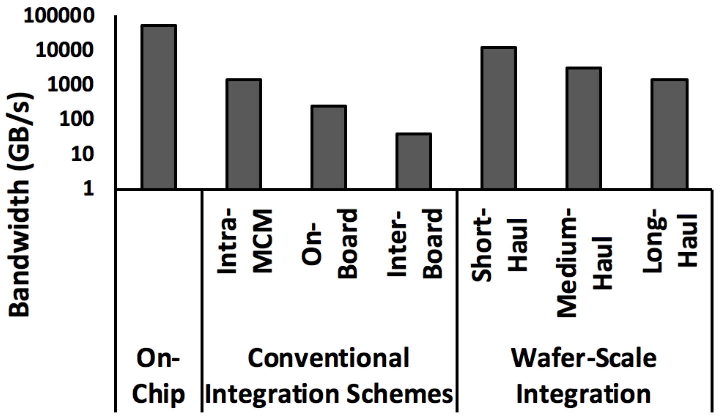


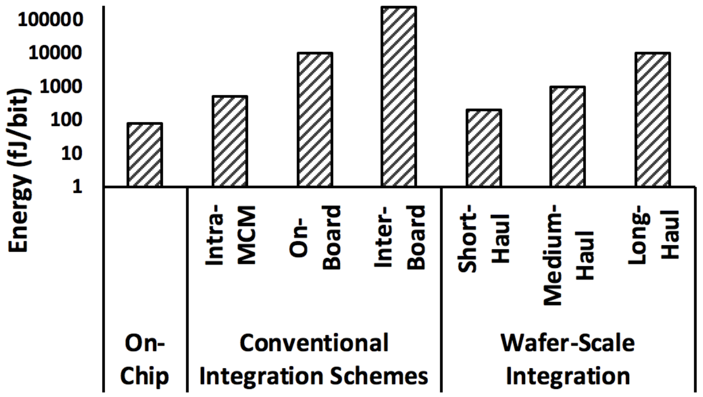
Figure 5.1: Minimum die/package footprint for different integration schemes are shown for increasing number of processor dies per system.

in terms of compute per area. Figure 5.2 compares the available communication bandwidth, latency, and energy per bit for waferscale integration versus conventional integration schemes and on-chip interconnects. In waferscale integration, the interconnects would have similar pitch as that of on-chip interconnects. In conventional integration schemes, while intra-die connections inside an MCM package can have fine wire pitch, the PCB traces as well as between-PCB links are I/O limited. This constrains the total bandwidth. Therefore, waferscale integration enables much larger bandwidth than what conventional integration schemes can provide. Since the waferscale links are smaller and high density, simple parallel communication protocol can be used where a massive number of links run at a relatively lower frequency [37]. This helps eliminate SerDes circuitry and thus reduces energy and latency of communication. There are also significant advantages in terms of test and packaging costs [79].

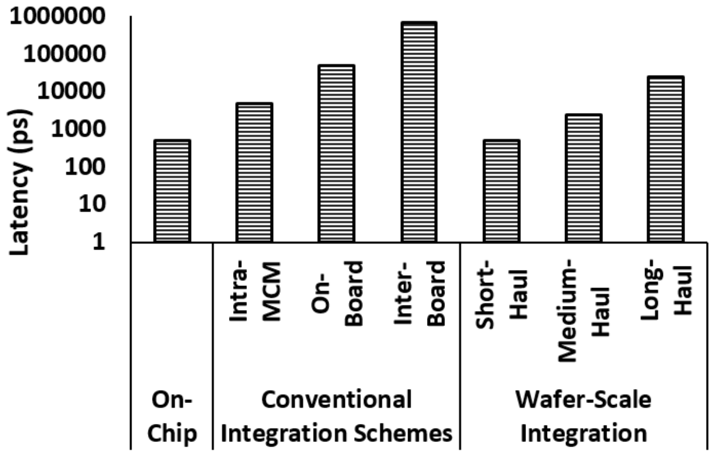
Unsurprisingly, waferscale processors were studied heavily in the 80s. There were also several commercial attempts at building waferscale processors [36, 160]. Unfortunately, in spite of the promise, such processors could not find success in the mainstream due to yield concerns. In general, the larger the size of the processor, the lower the yield - yield at waferscale in those days was debilitating [36].



(a) Bandwidth



(b) Energy Per Bit



(c) Latency

Figure 5.2: Comparison of communication link bandwidth, energy per bit and latency for on-chip links and different types of links in conventional integration schemes and waferscale integration scheme.

We argue that considerable advances in manufacturing and packaging technology have been made since then and that it may be time to revisit the feasibility of waferscale processors. In particular, it is now possible to reliably bond pre-manufactured dies directly on to the wafer [4, 183, 184]. So, it may be possible to build a high yield waferscale processor by bonding small, high yielding dies on to the wafer and connecting them using a low-cost wafer-level interconnect (Silicon Interconnect Fabric, $Si-IF$). Coupled with the fact that potential benefits from waferscale processing may be much larger now considering the high (and increasing) communication overheads today, we would like to better understand the benefits and challenges of building a waferscale processor today.

Previous work in [163] has proposed packageless processors based on the Si-IF based substrate and shown significant performance improvement coming from bandwidth, thermal and area benefits of removing packages. However, that work only focused on a conventionally-sized single-die processor system ($\sim 600\ mm^2$, 150W). This work, on the other hand, focuses on the architecture of a GPU system that is as large as an entire 300*mm* wafer, i.e., 70,000mm^2 of available area.

This chapter makes the following contributions:

- This is the first work that studies if it is feasible and useful to build a waferscale GPU system. We show that while a 300 mm wafer employing an emerging integration technology can house about 100 GPU modules (GPM), only a much scaled down GPU architecture with about 40 GPMs can be built when physical concerns are considered.

- We perform an architectural exploration for waferscale GPUs under different physical constraints. We find that waferscale GPUs are area-constrained due to power delivery network overheads, *not* thermally-constrained. We show that a 24 GPM architecture is possible on a 300mm wafer for a junction temperature constraint of 105°C. A 41 GPM architecture is enabled when 4-module voltage stacks are allowed with each GPM running at lowered voltage and frequency. We also find that waferscale GPU architectures can be supported with ring, mesh, or 1D/2D torus topologies - more connected topologies such as crossbars are not feasible to build due to wiring limitations on such large processors.

- We show that waferscale GPU architectures have considerable performance and energy efficiency benefits for many GPU applications compared to equivalent interconnected discrete

GPUs or even interconnected MCM-GPUs. E.g., *color* [185] has 10.9x and 17.8x speedup for 24-GPM and 40 GPM waferscale GPUs over equivalent interconnected MCM-GPU-based systems. Average performance and energy efficiency benefits of a 40-GPM system across all our workloads are 5.14x and 22.5x respectively.

- We study the impact of thread block scheduling and data placement on wafer-scale GPU architectures. Our techniques for thread group scheduling and data partitioning coupled with our placement strategy can provide up to 2.88x performance benefit (average 1.4x) over state-of-art [186] in large GPU scheduling. Average benefit in terms of EDP is 49% and 20% for 24 and 40 GPM systems respectively.

- Finally, we present the first Si-IF prototype with interconnected dies. The 100% successful interconnection between dies we observed for our 100mm wafer Si-IF prototype with 10 interconnected $4mm^2$ dies, coupled with high yield reported previously for bonding of dies on Si-IF, demonstrates the technological readiness for building waferscale GPU architecture.

5.2 Background and Technology Readiness

Several recent integration technologies have been aimed at building larger systems. In particular, 2.5D integration technologies such as TSMC CoWoS (interposer based solution) [187] and Intel's EMIB [175] allow building larger systems by integrating multiple high yielding dies on a high bandwidth, low latency interconnect substrate. However, these technologies have size limitations. Interposers use thinned silicon and are, therefore, fragile. As such, the size of interposer-based systems is usually limited to the size of the reticle. Beyond reticle size, interposers are built by stitching multiple reticles, which is a costly and complex process and has low yield [188, 189]. As a result, the largest commercial interposer today [190] is about 1230 mm^2 in size and accommodates only one GPU and 4 memory stacks. Even Intel's EMIB technology can integrate only about 5-10 dies on the interconnect substrate [175].

Another promising integration technology that can enable larger systems is *Silicon Interconnect Fabric, (Si-IF)* [4, 37, 163]. Si-IF replaces the organic printed circuit board (PCB) with a silicon

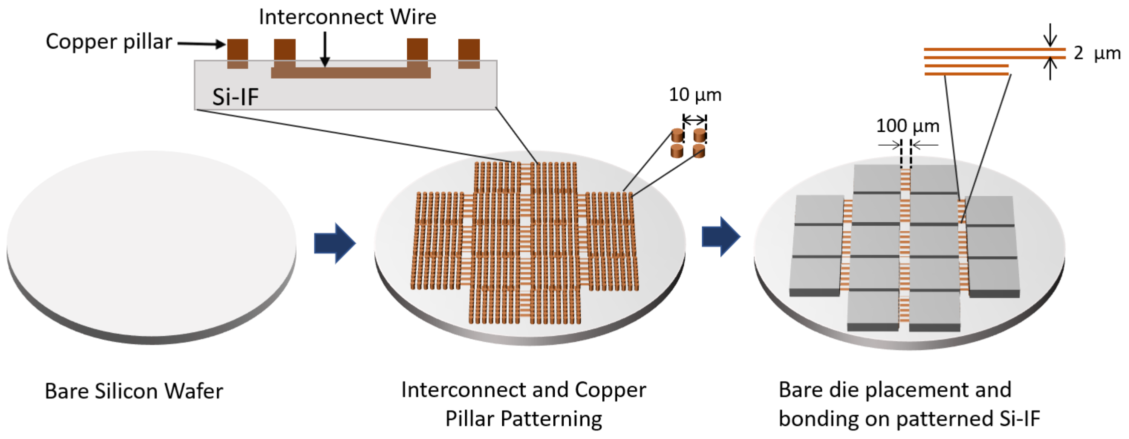

Interconnect Wire
Copper pillar
Si-IF
10 µm
100 µm
2 µm
Bare Silicon Wafer
Interconnect and Copper Pillar Patterning
Bare die placement and bonding on patterned Si-IF

Figure 5.3: The system assembly process flow is shown. Interconnect layers and copper pillars are made by processing the bare silicon wafer. Bare dies are then bonded on the wafer using TCB.

substrate and allows placing and bonding bare silicon dies directly on to the thick silicon wafer using copper pillar based I/O pins. The smaller, high yield dies are interconnected on the passive interconnect substrate (wafer) using mature fabrication technologies to ensure high yield. Different system components such as processor, memory dies alongside non-compute dies such as peripherals, voltage regulator modules (VRM), and even passives (inductors and capacitors) can be directly bonded on the Si-IF without the requirement of packages for individual components [4, 19, 163]. Figure 5.3 shows the process flow of a system assembly on Si-IF. Direct bonding of silicon dies on a silicon wafer using copper pillar coupled with short channels between the dies due to absence of packages allows Si-IF to achieve significantly more performant and energy efficient communication (Figure 5.3). [1]

Si-IF is an obvious candidate for waferscale integration since silicon dies (and other components) are directly integrated on to a silicon wafer. However, to enable waferscale processing, Si-IF must provide high system yield. Below we argue that Si-IF will provide the high system yield needed for waferscale integration.

There are three components to the yield of a Si-IF-based waferscale system - yield of the die,

[1]Other bonding technologies such as solder-based micro bumps can also be used. Tradeoff of using solder-based microbumps with Si-IF instead of using copper pillars include coarser pitch (25 μ vs 5 μm, higher electrical resistivity (11-13 $\mu\Omega$-cm vs 1.7 $\mu\Omega$-cm), higher likelihood of intermetallics and fatigue related failures, and easier debondability (at 220-230°C vs 1000°C).

yield of the copper pillar bonds, and yield of the Si-IF substrate. High yield (>99%) can be ensured for the dies by using known-good-die (KGD) testing techniques [71, 191] to pre-select the dies to be used for assembly on the Si-IF. The yield of the copper pillar bonds is also expected to be close to 99%. Note that the primary mode of copper pillar based I/O failures is opens. Copper pillars are not prone to extrusion unlike solder based connections, so shorts are not possible. Also, since both the substrate and the dies are made out of silicon, there is no co-efficienct of thermal expansion mismatch between them to cause large stresses on the copper pillar bonds due to large temperature fluctuations. Moreover, the misalignment in the place and bond tool used for the prototypes is <1 μm while the pillar spacing is 5 μm. Therefore, any I/O failures due to shorts or misalignment are unlikely. In fact, previous work indeed observed copper pillar yield higher than 99% [4, 163]. Furthermore, since fine pitch copper pillars ($<10\mu m$) allow at least 25x more I/Os than today's integration schemes with solder based connections ($>50\mu m$ pitch), redundancy can be employed to improve system yield i.e., multiple pillars per logical I/O. Moreover, network level resiliency techniques [192, 193] can be employed to route data around faulty dies and interconnects on the wafer to enhance system yield.

Finally, the yield of the Si-IF substrate will be high ($> 90\%$) since it is a passive wafer with only thick interconnect wires (2um width, 4um pitch) and no active devices[2]. We calculate the expected yield of an Si-IF substrate (shown in Table 5.1) for different number of metal layers and metal layer utilization with $2\mu m$ wire width and spacing using industry standard yield modeling equations 5.1 [140, 156] and 5.2 [147]. [3] The calculated yield values for the substrate for small number of metal layers is high. [4]

$$Yield = (1 + \frac{D_0 \times F_{crit} \times Area}{\alpha})^{-\alpha} \tag{5.1}$$

[2]Vast majority of the yield loss in semiconductor manufacturing happens in transistor layers (front-end-of-the-line) and first few metal layers with small pitch ($\leq$200nm).

[3]D_0 is the defect density per mm^2, α is the defect clustering factor and we use the ITRS value of 2200 and 2 [156] respectively for the calculations. r_c is the critical defect size, p is the interconnect pitch, F_{crit} is the fraction of critical area prone to faults.

[4]Since the inter-die interconnect would only run between the tightly spaced dies, the amount of interconnect area is expected to be less than 10% of the entire wafer area.

$$F_{crit}^{open} = \int_0^\infty (2r - p/2) * \frac{r_c^2}{r^3} dr = F_{crit}^{short} \tag{5.2}$$

Table 5.1: Yield of Si-IF for different number of metal layers

Si-IF Metal Layer Utilization (%)	**Number of Layers**		
	1	**2**	**4**
1	99.6	99.19	98.39
10	96.05	92.26	85.11
20	92.29	85.18	72.56

Previous Si-IF prototypes had not established connectivity across multiple dies. Therefore, there was no measurement of the yield of the Si-IF substrate or the system yield for an Si-IF-based system with interconnected dies. To assess viability of inter-die interconnect on Si-IF, we built a prototype where we bonded connectivity testing dielets on a 100mm waferscale Si-IF. Copper pillars are connected in a serpentine fashion within and across dielets as shown in Figure 5.4. Figure 5.5 shows the micrograph of the prototype. Each row on a dielet with dimensions 2mm$\times$2mm has 200 copper pillars in a row (40,000 pillars in total) which are connected using the serpentine structure. We connect an array of 5x2 dielets, each 4 mm^2. We tested the electrical connectivity across the dies to check whether Si-IF can indeed provide connectivity across many dielets at high yields.

Our electrical tests show that 100% of the interconnects in this prototype were connected illustrating very high yield of the copper pillars as well as inter-die interconnect on Si-IF. Post-bonding thermal cycling tests were done from -40°C to 125°C to test the impact of temperature change on copper pillar bonds and the results demonstrated that all the copper pillars and interconnects withstood the thermal cycles without any noticeable degradation in bond contact resistance. The high yield we observed for this prototype, coupled with high yield reported previously for bonding of dies on Si-IF, demonstrates the technological readiness for building waferscale systems. In subsequent sections, we use a GPU case study to a) explore the space of feasible waferscale architectures, b) understand and maximize the performance benefits from waferscale architectures, and c) develop data placement and thread scheduling strategies for waferscale GPUs.

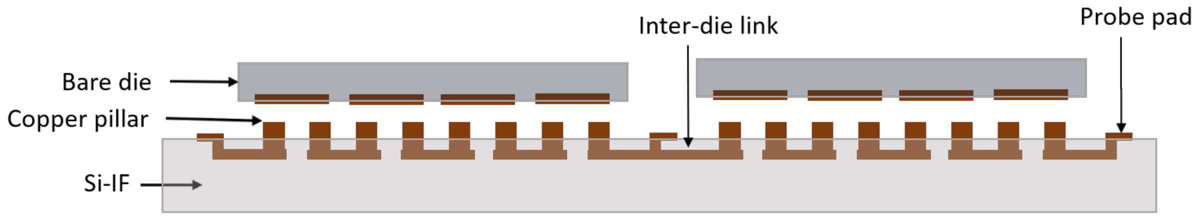


Figure 5.4: Schematic of the prototype with interconnection between the serpentine structure of two different dies is shown.

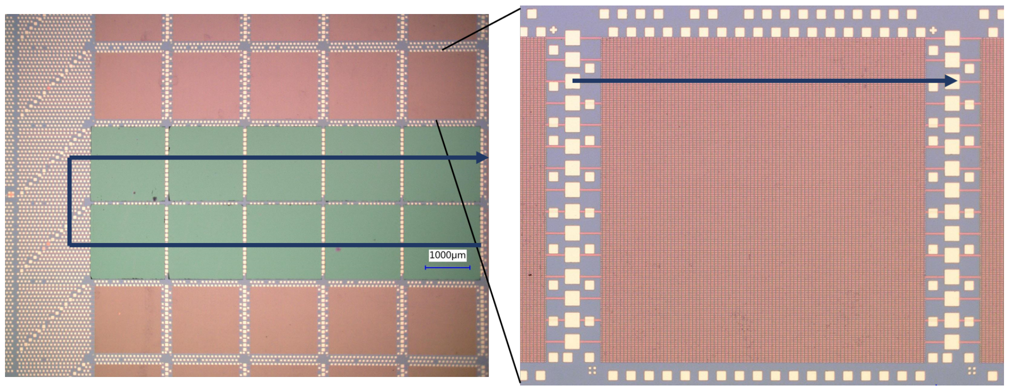


Figure 5.5: Micrograph of the prototype with inter-die connectivity is shown. Ten 4 mm^2 dies are bonded and tested for continuity of a signal across the dies. Each die has rows of serpentine structure as shown in the schematic. A zoomed-in picture of the Si-IF is also shown with 40,000 copper pillars.

5.3 A Case for waferscale GPU Architecture

GPU applications tend to have large amounts of parallelism [194, 195]. As such, GPU hardware parallelism keeps increasing [50, 186], limited only by cooling and yield. In fact, a large class of applications from the domain of physics simulations, linear algebra and machine learning routinely run across 100s of GPUs [5] - such applications will greatly benefit from increasing the effective size of a GPU since multi-GPU approaches have programmability [199] and communication overheads [186]. Proposals such as MCM-GPU [186] do attempt to increase the effective size of the GPU, but are limited by the size limit of conventional integration technologies.

Table 5.2: GPU Topologies

	ScaleOut SCM-GPU	ScaleOut 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 1.0 pJ/bit
Package Topology	Mesh	Mesh	Overall System Package
Package Interconnect	256 GB/s 96 ns 10 pJ/bit	256 GB/s 96 ns 10 pJ/bit	None

[5]E.g.NWChem, PSDNS, MILC, NAMD, QMCPAK, Chroma, GAMESS, MELD, AMBER etc. routinely run across the 3072 GPUs on Blue Waters [196–198]; Nvidia also provides HPC containers for warehouse scale GPU applications

We evaluated three constructions of a highly parallel GPU system, described in Table 5.2. We consider the smallest hardware unit in these constructions to be a GPM (GPU Module), roughly equivalent to a large sized GPU available today combined with a 3D-DRAM die. Each GPM has a TDP of 200W and area of $500mm^2$ for the GPU die, plus 70W and $200mm^2$ for two 3D-stacked DRAM dies. Note that the inter-GPM communication energy for waferscale case is higher than on-package inter-GPM communication energy in an MCM-GPU. This is because the GPM dies in the floorplan we considered (see Section 5.4.4) are separated by DRAM and voltage regulator modules (VRM) on the wafer and so the inter GPM distance is about $\sim$20 mm versus 2-5 mm in MCM package (where VRM are usually off-package or on the package periphery and not in between the dies)

The first construction we consider is ScaleOut SCM-GPU (single-chip module GPU), where each GPM is contained in its own package. GPMs are placed in a 2D mesh on a traditional PCB, connecting via an inter-package link with bandwidth, latency and energy characteristics similar to QPI. The second is ScaleOut MCM-GPU, an extension of MCM-GPU where MCM-GPU units are placed in a 2D mesh on a traditional PCB connected with a QPI-like link. The last architecture we evaluate is a hypothetical waferscale GPU (i.e., we do not consider thermal or power delivery constraints) - a single wafer containing a 2D mesh of GPMs connected via Si-IF [37]. The GPMs constitute a single logical GPU from the perspective of the programmer.

Figures 5.6, 5.7 show the potential advantages of a waferscale GPU over the ScaleOut SCM-GPU or ScaleOut MCM-GPU approach for two benchmarks, SRAD and Backprop, both from the Rodinia benchmark suite [200]. These two applications were chosen to be representative of medical imaging and machine learning, both fields expected to substantially benefit from waferscale processing. Our simulations are performed by using gem5-GPU [201] to generate memory traces and activity profiles, which we feed into our own trace-based GPU simulator. We expand upon our experimental methodology in Section 7.4.

For Backprop, we observe a 47.54x speedup for 64 GPM waferscale GPU over a single GPM system. Speedup is 20.8x and 21.13x over the highest performing ScaleOut SCM-GPU and ScaleOut MCM-GPU configurations respectively. The speedups are eventually limited by the memory transfer latency. Note that these speedups are achieved without requiring changes to the programming model

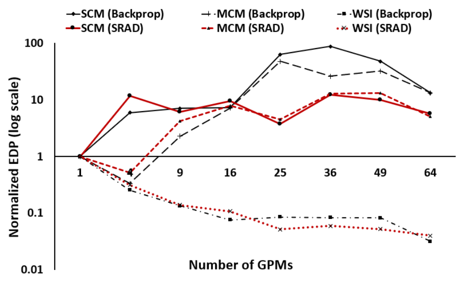


Figure 5.6: Normalized EDP for Backprop and SRAD.

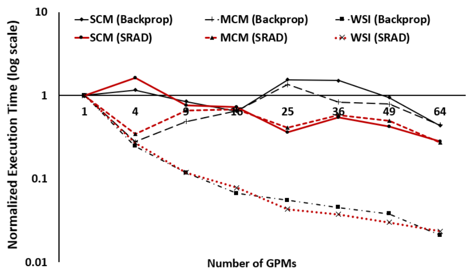


Figure 5.7: Normalized Execution Time for Backprop and SRAD.

unlike speedups achieved by other ScaleOut system integration schemes.

The benefits of waferscale GPU over ScaleOut SCM-GPU and ScaleOut MCM-GPU are more apparent when considering EDP. 64 GPM Waferscale GPU has a 31.54x reduction in EDP compared to a single GPM and trends downwards, whereas both ScaleOut MCM-GPU and ScaleOut SCM-GPM systems increase in EDP past 9 GPMs.

For SRAD, we observe a 42.56x speedup for 64 GPM waferscale GPU over a single GPM system. This is compared to ScaleOut SCM-GPU and ScaleOut MCM-GPU which saturate at 3.57x and 3.65x speedup, respectively. Additionally, by avoiding costly inter-package communication, the 64 GPM waferscale GPU manages a 24.88x reduction in EDP, a sharp contrast to ScaleOut MCM-GPU and ScaleOut SCM-GPU, where additional GPMs actually increase EDP.

The above results show that GPU architectures are a good fit for building at waferscale as performance and energy efficiency scaling of GPU applications is much stronger on a waferscale GPU than equivalent interconnected discrete GPUs or even interconnected MCM-GPUs. In the next section, we explore the space of feasible waferscale GPU architectures.

5.4 Architecting a Waferscale GPU

Architecting a waferscale GPU is a unique problem due to the physical constraints of a waferscale processor. A waferscale GPU architecture will need to operate at kilowatts of power; the corresponding architecture must be feasible in presence of the associated thermal and power delivery concerns. Similarly, a waferscale GPU will need enormous interconnection resources (due to the need for connectivity among the GPMs). A waferscale GPU architecture must support network topologies that are realizable at the waferscale level.

In this section, we attempt to identify feasible waferscale GPU architectures in presence of thermal, power delivery, and connectivity constraints. Our analysis considers a GPM module consisting of a 500 mm^2 GPU die and 200 mm^2 DRAM dies with TDP of 200W and 70W respectively[6].

5.4.1 Waferscale GPU Architecture under Thermal Constraints

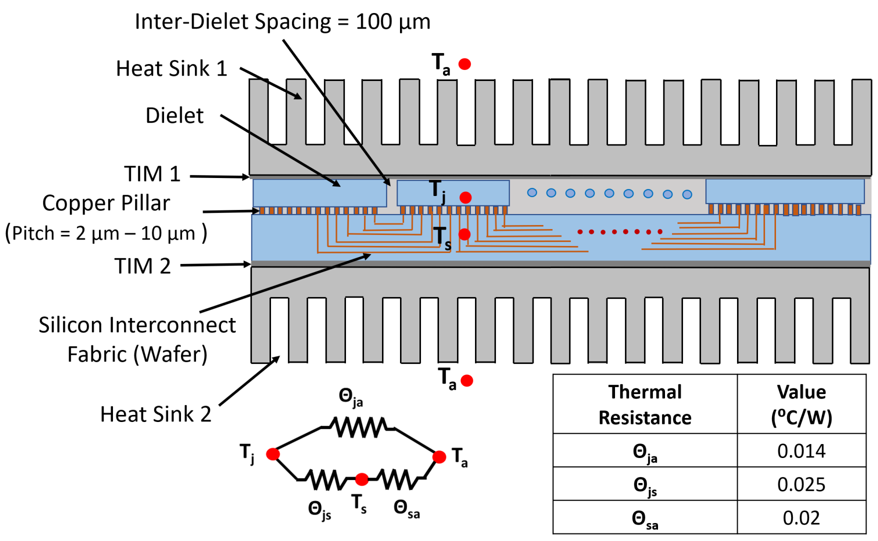
In this subsection, we ask the question: Given a target maximum junction temperature and forced air cooling, what is the maximum number of GPMs that can be accommodated on the 300 mm wafer?

To determine the maximum allowable TDP, we assume that system would be cooled using one or two square heat sinks covering the round 300 mm wafer with forced air convection cooling. Figure 5.8 shows the schematic of two heat sinks attached to the wafer, one directly on top of the dies, and the other on the back side of the wafer. The secondary heat sink not only provides mechanical support for the wafer, it also helps increase the heat extraction efficiency. The thermal resistance model is shown in Figure 5.8. The thermal modeling and analysis is performed using a commercial CFD-based thermal modelling tool from R-tools [8] - CFD is known to provide more accurate results than simple spice based models used in tools such as HotSpot [202].

We evaluated both the cases, one heat sink and two heat sinks for three different junction temperatures (T_j). Conventionally, 85°C [173, 203] and 105°C [204] are used as reliable T_j. Since, we were not able to simulate a very up-to-date heat sink solution, for fair comparison, we performed thermal simulation of a recently published multi-GPM system (MCM-GPU) described in [186] using our framework. It resulted in a junction temperature of 121°C considering a heat-sink of the size of the package 77mm$\times$77mm with ambient temperature of 25°C. As a result we also analyze for T_j=120°.

We consider that 20000 mm^2 out of the 70000 mm^2 would be used for external connections and

[6]3D stacked memory is not a necessity for waferscale. Using a planar memory dies would decrease the capacity and bandwidth per unit area.



Thermal Resistance	Value (°C/W)
Θ_{ja}	0.014
Θ_{js}	0.025
Θ_{sa}	0.02

Figure 5.8: Schematic cross-section of a waferscale system on Si-IF, alongside the thermal resistance model. Ta, Tj , Ts denote the ambient, chip-junction and silicon substrate temperatures respectively. Two heat sinks one directly attached to the dies and another backside heatsink covering the Si-IF substrate is shown. The thermal resistance values for a 300 mm^2 waferscale system with heatsinks are also shown [8].

other interfacing dies[7]. Therefore, 50000 mm^2 would be available on the wafer for placing the GPMs and point-of-load voltage regulator modules. For the maximum TDP estimation, we consider that multiple heat sources (GPMs and DRAMs) are generating heat on a surface of size 50000 mm^2.

In Table 5.3, we show the sustainable TDP for the various scenarios described above. We also present the total number of GPMs within that thermal budget with and without voltage regulator modules (VRMs). When no VRM is considered, the only heat sources are the GPM modules. In case VRMs are placed on the wafer, there would be additional heat loss due to VRM inefficiency. Here, we assume on-Si-IF VRM to have an efficiency of about 85% [205]. Therefore, effectively a VRM would lead to an additional power dissipation of 48W per GPM.

Our analysis shows that while 50000 mm^2 of area is available on the wafer for computational purposes ($\sim$ 71 GPMs), thermal limitations constrain the maximum number of GPMs that can be placed on the Si-IF to be much lower. Considering the dual heat sink solution, up to 34 GPMs can

[7]A waferscale GPU can interface to the external world using multiple (e.g., PCIe) ports connected to one or more root complexes.

Table 5.3: No of supportable GPMs for different junction temperatures

Target Junction Temperature (°C)	Dual Heat Sink			Single Heat Sink		
	Power (W)	Num GPMs w/o VRM	Num GPMs with VRM	Power (W)	Num GPMs w/o VRM	Num GPMs with VRM
120	9300	34	29	6900	25	21
105	7600	28	24	5400	20	17
85	5850	21	18	4350	16	14

be supported if no power loss from VRM is assumed, else the number reduces to 29 GPMs.

5.4.2 Waferscale GPU Architecture considering Power Delivery

A waferscale GPU system is constrained by the heat sink technology to a total TDP of up to about 9.3 kW. Considering the rated TDP to be 0.75 times [2, 57] the peak power of the system, the power distribution network (PDN) must be able to provide up to 12.5 kW of power (compared to 1-2kW for a modern server board [82, 206, 207]) with reasonable efficiency even at peak power.

We explore external power supply alternatives of 48V, 12V, 3.3V and 1.2V to the wafer with a point of load (POL) power conversion using efficient buck converters for every GPM i.e, there would be one VRM per GPM. In general, the higher the input voltage, the larger the PDN circuitry overhead, but also the fewer the number of layers required to supply power (also lower resistive loss), as shown in Tables 5.4 and 5.5.

We use the power distribution mesh sizing models given in [208] to determine the minimum area and thus the *number of layers required for power distribution*. As shown in Table 5.4, we find that the number of metal layers required for 1V and 3.3V supply to the wafer is very high, even for a very large I^2R loss. Moreover, more than 4 metal layers for power delivery is undesirable due to cost and manufacturability reasons. Therefore, the only viable options are 12V or 48V external power input.

To compare the two external power input options (12V and 48V), we recognize that the size

Table 5.4: Number of Layers Required vs Supply Voltage to the Wafer. 10μm thick metal is available in most technologies which support RF

Input Voltage (V)	**I^2R Loss (W)**	**No of Layers**		
		Thickness = 10μm	**Thickness = 6μm**	**Thickness = 2μm**
1	500	42	68	202
3.3	200	10	16	44
	500	6	8	18
12	100	2	4	10
	200	2	2	4
48	50	2	2	2
	100	2	2	2

of VRMs for DC-DC conversion and regulation can be very large due to the required inductance and capacitance. Area efficiency of state-of-art 48V-to-1V converters at reasonably high power conversion efficiency (>90%) is in the 1W/10mm^2 - 1W/5mm^2 (including the VRM inductor) [205, 209] range when implemented using PCB based integration. We conservatively assume 1W/6mm^2 area overhead of VRM for a 48V-to-1V power conversion.

This means that to support a GPM with TDP 200W alongside 70W 3D stacked local DRAM (i.e., peak power of 360W), the area of the VRM for the 48V-to-1V option would be approximately 2160 mm^2 (360×6)! We also calculated the area overhead of surface mount decoupling capacitors required to compensate for current load variation of about 50A with a frequency of 1 MHz [210] to be $\sim$300 mm^2. Therefore, for a 48V-to-1V conversion strategy, the total number of GPUs that can be accommodated in the usable 50,000mm^2 area on the wafer would be only about 15. Also, in this case, the total peak power draw from the external source would be $\sim$ 6.5 kW i.e., a TDP of 4.9 kW assuming 85% VRM efficiency and accounting for I^{2}R losses. Thus, *the system is area constrained and not TDP constrained (since the maximum allowable TDP is about 9.3 kW)*. This is a salient result since it suggests that much higher performance and energy efficiency may be possible from

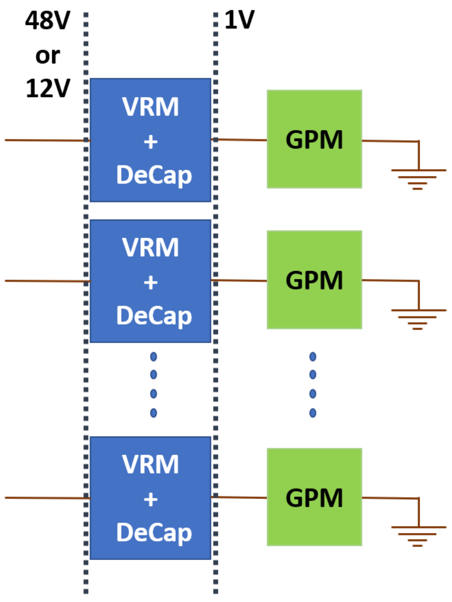
Table 5.5: VRM & Decap Overhead Per GPM

Input	**VRM+DeCap Area Per GPM (mm^2)**			**Number of GPMs**		
Voltage (V)	**No Stack**	**2-Stack**	**4-Stack**	**No Stack**	**2-Stack**	**4-Stack**
1	300	-	-	50	-	-
3.3	1020	610	-	29	38	-
12	1380	790	495	24	33	41
48	2460	1330	765	15	24	34

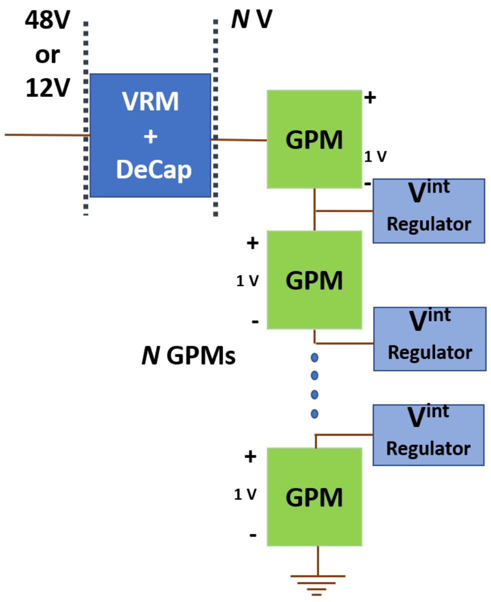
waferscale computing simply through improving the area efficiency of voltage conversion.

For the 12V option, the size of the VRM is expected to be smaller ($\sim$ 1W/3mm^2). Therefore, about 24 GPMs could now be accommodated which would amount to a total of approximately 10.3 kW of maximum power (7.8 kW TDP). However, we still cannot accommodate the maximum number of 29 GPMs, as dictated by the thermal limit at 120°C T_j. Therefore, as was the case for 48V supply, the system is area constrained, not thermally constrained.

An alternate power distribution strategy is voltage stacking of multiple GPUs [211–213], as shown in Figure 5.9. If N GPMs are stacked, the supply voltage to the stack should be N times the supplied voltage required for one GPM and the same current flows through the stacked GPMs. As shown in the previous work [213], this approach is viable in the GPM context since neighboring GPMs are expected to have similar activity and power draw at any time interval (good data placement and scheduling policy can also help). Now, instead of using N voltage regulators (one per GPM), one VRM with $1/N$ conversion ratio would be shared across N GPMs. As the conversion ratio decreases, the size of the VRM modules can be decreased for the same efficiency. Though a perfectly balanced stack would ensure stable intermediate node voltage, we anticipate use of lightweight voltage regulators (such as push-pull regulators [214] which can reduce middle-rail noise while minimizing static power dissipation) to ensure guaranteed stability of the intermediate nodes. More details regarding such intermediate voltage circuitry has been shown in [213]. Since these intermediate voltage node regulators would only be responsible for stabilizing small current demands and not voltage conversion, compact switched capacitor (SC) based regulators or linear drop-out (LDO)



(a) One VRM per PDN



(b) Stacked voltage supply

Figure 5.9: PDN schemes.

regulators can be used. Our conservative estimates (based on experience and prior work) suggest that these intermediate regulators would have an area footprint of around 200 mm^2. Due to this reduced per GPM footprint, we find that 34 GPMs can now be accommodated with 4 GPMs per stack and 48V power supply to the wafer, while 41 GPMs can be accommodated with 12 V supply and 4 GPMs in a stack. These results show that voltage stacking is a promising technique to enable scalable waferscale GPU architectures. Table 5.6 shows the different proposed PDN design choices and corresponding number of supportable GPMs.

While up to 41 GPMs can be supported using voltage stacking, recall that thermal limits only allow us to pack 29 GPMs (with VRMs considered) running at nominal operating conditions. We, therefore explore the opportunity to further maximize the number of GPMs by decreasing the supply voltage and operating frequency of each GPM. To accommodate 41 GPMs, we find the operating voltage and frequency of these GPMs such that the total power is within the maximum thermal power budget. We only considered scaling the GPM voltage while maintaining the same DRAM voltage. The scaled operating voltage and frequency for GPMs is presented in Table 5.7. Notice that to support the decreased voltage per GPM, the down conversion ratio of the VRM needs to be

Table 5.6: Proposed PDN solutions

Target Junc. Temp. (°C)	Dual Heat Sink			Single Heat Sink		
	Thermal limit (W)	Supply Voltage (V)/ # GPMs per Stack	Maximum Number of GPMs at Nominal	Thermal limit (W)	Supply Voltage (V)/ # GPMs per Stack	Maximum Number of GPMs at Nominal
120	9300	48/4 or 12/2	29	6900	48/2 or 12/1	21
105	7600	48/2 or 12/1	24	5400	48/2 or 12/1	17
85	5850	48/2 or 12/1	18	4350	48/1	14

Table 5.7: Operating Voltage and Frequency for the 41 GPMs with 12 V supply and 4-GPMs per stack

Target Junc. Temp. (°C)	Dual Heat Sink			Single Heat Sink		
	GPM Power (W)	Operating Voltage (mV)	Operating Frequency (MHz)	GPM Power (W)	Operating Voltage (mV)	Operating Frequency (MHz)
120	125.75	877	469.6	71.75	752	364.2
105	92	805	408.2	44.75	664	291.4
85	51.5	689	311.7	24.5	570	216.2

enhanced. Earlier, we conservatively estimated the size of the 12V/4V VRM to be that of 12V/1V conversion ratio, this increase in down conversion ratio (up to 12V/2.4V) can be easily handled by the VRM of this size.

5.4.3 Allowable Network Architectures for a Waferscale GPU

The above analysis does not consider interconnection between the GPM modules. For a GPM die size of 500 mm^2 (90 mm perimeter), wire pitch of 4 μm and effective signalling rate of 2.2 GHz per wire [37] (ground-signal-ground with 4.4GHz signal speed), the total bandwidth available per layer is ∼6TBps. Increasing the number of layers would result in increased inter-GPM and DRAM bandwidth, however, it would lower yield[8]. As shown in [186], increasing the local DRAM bandwidth to the GPMs beyond 1.5TBps results in only a very small performance improvement, but lowering the bandwidth results in significant performance loss. With this DRAM bandwidth in mind (1.5TB/s), we analyze a few realizable inter-GPM network topologies for different signal metal layer count on Si-IF (see Table 5.8). Here, we only consider yield loss due to shorts and opens of the signalling wires on different metal layers[9]. One should note that increasing the DRAM bandwidth has much smaller effect on the Si-IF interconnect yield than increasing the inter-GPM bandwidth as the GPM to local DRAM spacing is 100-500 μm spacing while the inter-GPM distance for a 5×5 GPM array would be about 16 mm. We assume KGD and that the copper pillar redundancy scheme would take care of the yield loss due to bonding failure.

Table 5.8: Inter-GPM Network Topologies

Num of Layers	Topology	Memory Bandwidth (TBps)	Inter-GPM Bandwidth (TBps)	Yield (%)	Diameter	Average Hop Length	Bisection Bandwidth (TBps)
1	Ring	3	1.5	95.9	15	7.5	3
	Mesh	3	0.75	95.9	10	4	3.75
	Connected 1D Torus	3	0.5	94.1	8	3	3.75
2	Ring	6	3	91.9	15	7.5	6
	Ring	3	4.5	88.6	15	7.5	9
	Mesh	6	1.5	91.9	10	4	7.5
	Mesh	3	2.25	88.6	10	4	11.25
	Connected 1D Torus	3	1.5	84.3	8	∼3	11.25
	2D Torus	3	1.125	79.6	5	∼2.6	11.25
3	2D Torus	6	1.5	77.0	5	∼2.6	15
	2D Torus	3	1.875	73.4	5	∼2.6	18.75

Three topologies: ring, mesh and connected 1D Torus can be realized using one layer. Note that 2D Torus cannot be realized using one metal layer without major design and signalling efforts as some links would have to be routed around the GPM array.

Moving to a two layer solution, we can see that a ring network would be over provisioned with inter-GPM as well as DRAM bandwidth. Increasing the signal layer count to three layers enables more balanced 2D torus network, however at an expense of yield which can now be as low as 73.4%.

[8]Increasing the number of layers increases process complexity as well as the amount of critical area susceptible to particle defects [147, 215]

[9]Yield is calculated using the industry-standard negative binomial yield model [140, 147, 156] described in section 7.2. We use inverse cubic defect density distribution [215], defect density value as per ITRS [156] and metal pitch of $4\mu m$.

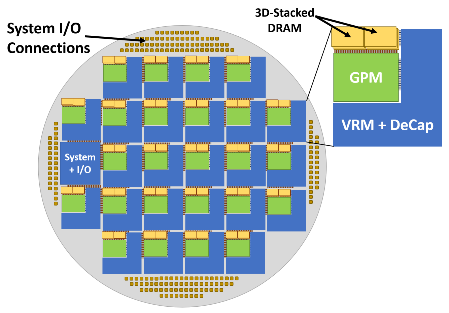


Figure 5.10: Waferscale GPU with 25 GPM units (1 redundant unit) comprising of two 3D-stacked DRAM per unit, VRM unit and decoupling capacitors.

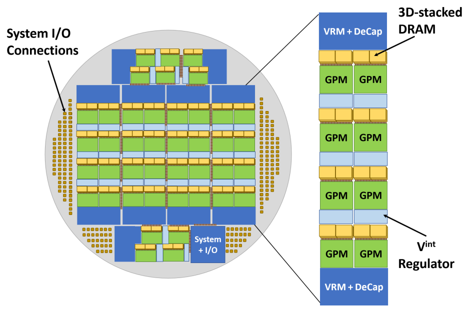


Figure 5.11: Waferscale GPU with 42 GPM units (2 redundant units) comprising of two 3D-stacked DRAM per unit, VRM unit and decoupling capacitors.

To summarize, yield concerns constrain the total number of metal layers on a GPU and this, in turn, limits the allowable network topology configurations on a waferscale GPU. We are now ready to select the viable waferscale GPU architectures that maximize performance and energy while satisfying all the physical constraints.

5.4.4 Overall System Architecture

We consider two configurations at the target junction temperature of 105°C; one with 24 GPMs running at nominal voltage of 1V and 575 MHz, second with 40 GPMs running at reduced voltage of 805mV and 469 MHz. For the former, we consider power supply at 12V and no stacking, while for the latter case, we consider 12V power supply with 4 GPMs in a stack. We show the floorplans for both these options in Figures 5.10 and 5.11. We show the floorplans with 25 GPMs and 42 GPMs considering redundancies of 1 GPM and 2 GPMs respectively.

Our inter-GPM network is a mesh network that uses two metal layer. The local DRAM bandwidth as well as inter-GPM bandwidth for each GPM is considered to be 1.5 TBps. Though 6 TBps of DRAM bandwidth per GPM can be supported with one layer, with 6pJ/bit memory access energy, the total DRAM power would be quite high (200 W) [216]. Hence, we considered DRAM bandwidth of 1.5TBps as it would result in about 72W of DRAM power. This is almost equal to the aggregate TDP of 2 3D-stacked DRAM modules [217] that we have assumed per GPM.

For the first case, when no stacking is used, every GPM has a set of 2 local 3D stacked DRAM chips, a VRM and decoupling capacitors, forming a tile of dimensions 42mm$\times$49.5mm. The goal is to floorplan a total of 25 such regular tiles. A 5×5 array floorplan is difficult to realize since the size of the largest square that can be inscribed in a round 300mm wafer is only about 45000 mm^2 ($\sim$ 21 tiles). Hence, one possible floorplan to lay out these 25 tiles is as shown in Figure 5.10. This floorplan closely resembles a mesh architecture without the corner tiles. We also considered the area for $System+I/O$ which would house many system level blocks for external interfaces (CPU-GPU, drivers), oscillators etc.

Similarly for the floorplan with voltage stacking, we place 42 dies with one VRM + DeCap every voltage stack of 4 GPMs and three intermediate node voltage (V^{int}) regulators. We were able to place 32 GPMs in an array while 10 other GPMs and $System+I/O$ are placed on the top and bottom side of the array. A mesh network connects the GPMs. Considering KGD GPM, DRAM and VRM die with 99% average yield per I/O and 4 pillars per I/O, the estimated bond yield for the 25 and 42 GPM systems are about 98% and 96.6% respectively. The Si-IF substrate yields which depend on the interconnect length are 92.3% and 95% respectively, this is because the inter-GPM wires are smaller in the 40-GPM floorplan. Therefore, overall yield is expected to be about 90.5% and 91.8% for the two cases respectively. Note that while the floorplans have 25 and 42 GPMs respectively, maximum number of operating GPMs would be limited to 25 and 40 due to thermal budgets. The extra GPMs can be used as spare GPMs to improve system yield in case of one/two GPMs become faulty.

Note also that even larger GPU systems could be built by tiling multiple wafer-scale GPUs. Given we have 940 mm of wafer edge (300 mm wafer diameter) and 20,000 mm^2 of area left for external connectors, about 20 PCIe socket connectors could be accommodated at the periphery assuming half the periphery is used to deliver power. Using PCIe 5.x which supports 128 GBps per x16 link, a total of 2.5 TBps of off-wafer bandwidth can be supported.

System Integration: One system integration strategy (Figure 5.12) involves using a primary heat sink and an optional secondary heat sink to cover the wafer assembly (which is passivated to protect against humidity, etc). In case the secondary heat sink is not used, a backside rigid metal plate would be used to encase the system. In case of installation into a chassis (servers/ desktops etc.),

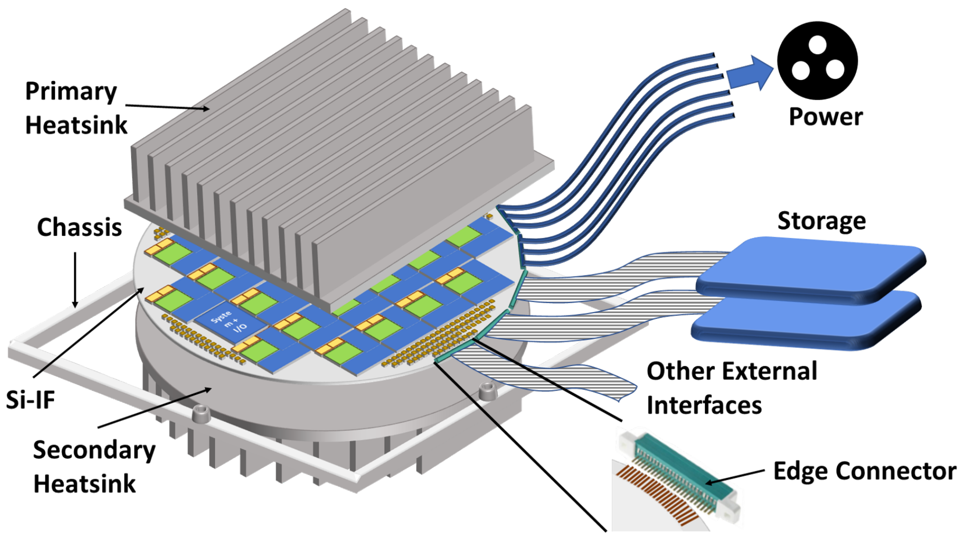


Figure 5.12: An Si-IF system assembly is shown with the primary and backside secondary heat sinks. The whole system is bolted to a chassis. The host CPU could either be connected externally or reside on the wafer itself.

the complete system can be inserted using normal plug connectors or low force insertion sockets [10]. Alternatively, the external metal heat sinks can be used to bolt to the chassis. We estimate that a row in a standard 19in wide / 36in deep cabinet can house two 300mm (12in) waferscale processors, including heatsinks. A 42U cabinet can house up to 6 rows (i.e., 12 WS-GPUs).

5.5 Thread Block Scheduling and Data Placement

Performance and EDP of a waferscale GPU architecture would also depend upon how compute and data is distributed across the waferscale system. Conventionally, thread blocks (TB) in a GPU during kernel execution are dispatched by a centralized controller to the compute units (CUs) in a round-robin order based on CU availability. However, such a fine-grained scheduling policy could place TBs of a kernel across multiple GPMs. Often, consecutive TBs benefit from data locality, and therefore such a policy could destroy the performance and energy benefits of waferscale integration

[10]Since silicon is a much more robust material than FR4 material used to build PCBs (compressive strength of 3.2-3.4 GPa vs 370-400 MPa) and with backside support (using heat sink or plate), a 700 μm to 1 mm thick wafer can easily bear the normal insertion force of plug connectors (few 10s of MPa); connections can also be wire-bonded to the system I/O pads, if needed.

as a large number of memory accesses would now need to be made across the inter-GPM network. Therefore, we use distributed scheduling instead of centralized scheduling.

In this distributed policy, a group of contiguous TBs of a kernel are assigned to each GPU so that spatial data locality between TBs can be utilized. Such a policy was used for the MCM-GPU presented in [186]. Consecutive groups of TBs were placed on the GPM array starting from a corner GPM and moving row first. Data placement is first-touch, i.e., when the first memory access to a particular page is done, the page is moved to the local DRAM of the GPM from which the memory reference was made. However, strong spatial data locality may still exist between non-neighboring TBs which cannot be exploited by this online policy. In a waferscale GPU with a large number of GPMs, because communication between non-neighboring GPMs can result in high multi-hop latency, this can lead to poor performance. Therefore, we need policies that allow TBs which share a large amount of data to be placed on neighboring GPMs so as to minimize data access latency as well as total network bandwidth utilization.

To solve this problem, we developed an offline partitioning and placement framework where the goal is to find the schedule and GPM allocation for TBs and DRAM pages that minimizes remote memory accesses. Our framework is fully automated and takes in a TB - DRAM page (TB-DP) access graph as input and outputs TB to GPM mapping as well as data placement (shown in Figure 5.13).

Nodes in the TB-DP access graph represents either a TB or a DRAM page, and an edge between a TB and a DRAM page signifies that the particular TB accesses the DRAM page. The edge weight corresponds to the total number of accesses. Given this graph, the aim is to partition the graph in to k partitions such that the total weight of the edges crossing the partition boundaries is minimized. We solve this partitioning problem using an iterative form of Fiduccia-Matthessey (FM) partitioning algorithm [218] where in each iteration of the algorithm, we extract one partition with N/k nodes. In our implementation, we allow the size ratio to drift by up to $\pm 2\%$ to minimize partition cut further. This generates a TB schedule as well as the corresponding data placement for an application which minimizes data accesses across the partitions. Partitioning, however doesn't solve the problem of minimizing overall load on the inter-GPM network and if a few but very remote (many-hops) accesses still remain, latency issues can affect the overall performance and

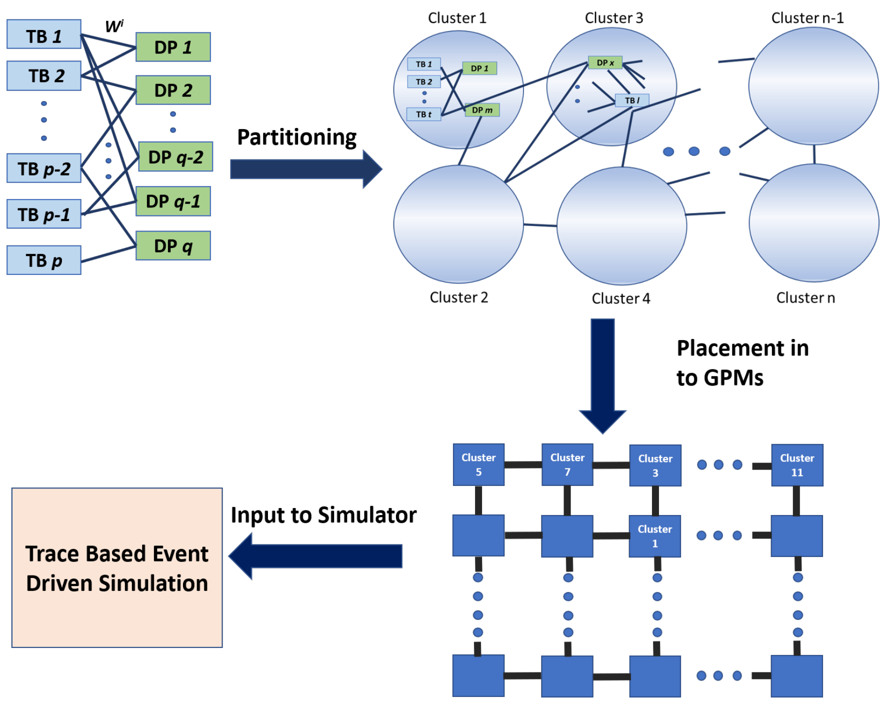


Figure 5.13: TB and DRAM page partitioning and physical GPM mapping flow.

energy efficiency. Therefore, given the network topology and number of GPMs, the next step is to allocate the TB-DP clusters to these GPMs, this is the *cluster placement problem*.

For the placement problem, we consider minimization of a *remote access cost* metric which is the summation of product of number of accesses and distance between the source and destination of the access. For example, let's consider a grid of 5x5 GPMs and 5 accesses made between GPMs at locations (1,1) and (3,5). The minimum Manhattan hop distance between the locations is 6 hops. Therefore the cost we consider is 30. The total cost is indicative of the total bandwidth utilization of the inter-GPM network and minimizing number of hops essentially minimizes latency of accesses. We use simulated annealing based placement to map the clusters to the appropriate GPMs in the GPM array. In Figure 5.14, we compare this cost for network topologies with baseline runtime dynamic scheduling and first touch page placement (discussed below) for the 40 GPM system. Our offline policy reduces the cost of accesses by up to 57%.

Note that, the partitioning and placement policy has been driven by spatial access patterns. A policy based on spatio-temporal access patterns would be able to provide better optimizations but we leave it for future work. Moreover, in this partitioning scheme, we didn't consider load

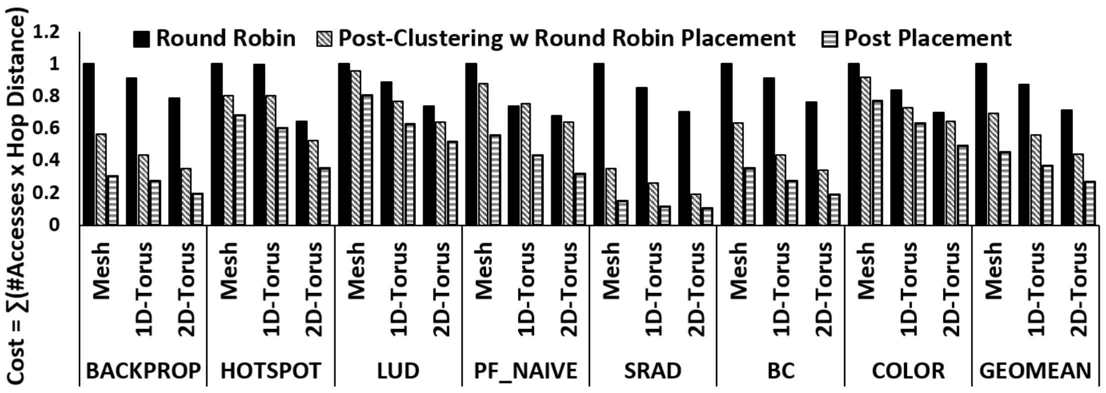


Figure 5.14: Improvement in the access cost metric from offline partitioning and GPM placement is shown. Baseline is data locality aware distributed scheduling and first touch data placement.

balancing explicitly. The partitioning algorithm divides the TB-DP graph into k nearly equal partitions, therefore this scheme does not necessarily load balance the TBs. We therefore, also use a runtime load balancing scheme on top of the static partitioning, where if there are TBs waiting (to be allocated to a CU) in the queue in a GPM and there are idle GPM(s), the queued TBs are migrated to the nearest idle GPM.

Other Policies: We also evaluated an online locality aware placement policy where the first group of TBs is placed in the centre GPM and the subsequent groups are assigned to the GPMs spirally out of the central GPM. This policy showed performance within $\pm 3\%$ compared to the simple placement policy of starting from a corner GPM and moving row first. For offline policies, we considered other access cost metrics such as summation of $\#access^2 * hop$ (this allows the most connected TB clusters to be placed closest) and $hop^2 * \#access$ (this minimizes maximum latency of data accesses). However, placements generated from these metrics have 2% poorer performance on average compared to the $\#access * hop$ metric, except 7% benefit when using $\#access * hop^2$ on 24-GPU system for *color* which is an irregular application and is network latency bound. Therefore, in Section 5.7, we only discuss and analyze the online policy with simple placement and offline policy with $data * hop$ metric for cluster placement.

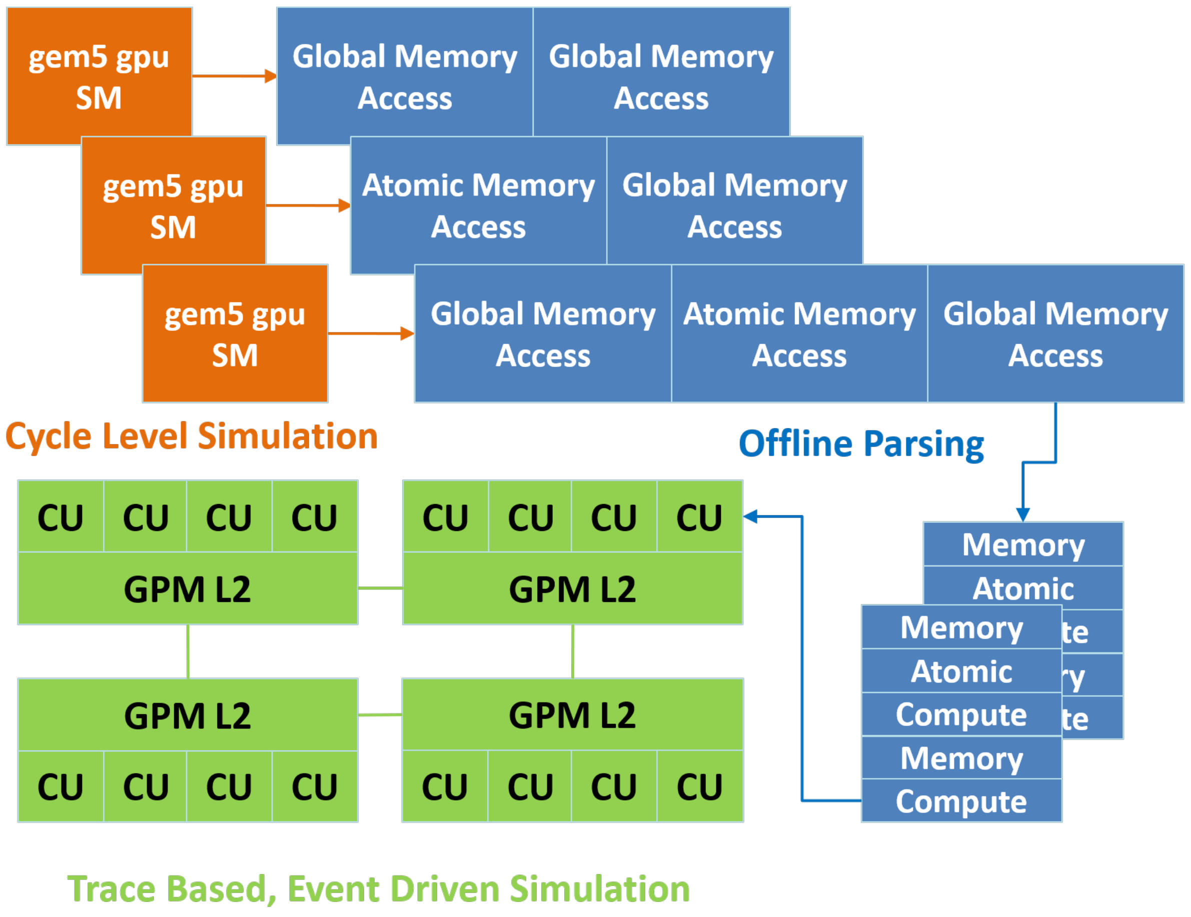


Figure 5.15: Simulator Workflow

5.6 Methodology

Current architectural simulators such as gem5 [219] and GPGPU-Sim [220] were designed to study systems with at most dozen of compute units: they are simply unable to simulate waferscale processors in a reasonable time-frame. In order to study the scalability of GPU applications on a waferscale processor, a more abstract simulation model is necessary.

Figure 5.15 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 LSQ (Load-Store Queue) of each CU. Next, we run each benchmark in fast-forward through Linux boot

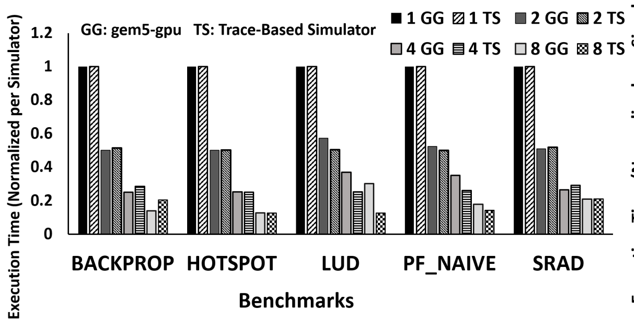


Figure 5.16: CU Scaling for Gem5-GPU and Trace-based Simulator.

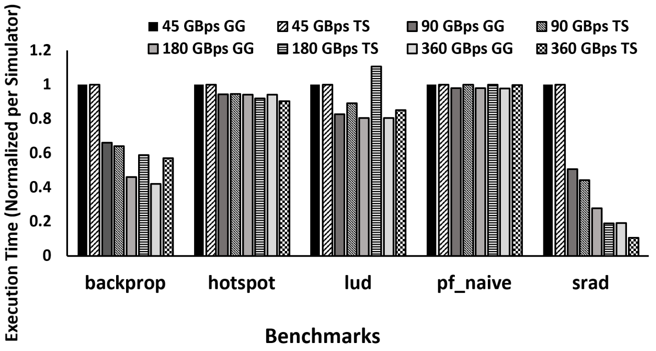


Figure 5.17: DRAM Bandwidth Scaling for Gem5-GPU and Trace-based Simulator for 8 CUs.

until the beginning of the ROI (region of interest), where we take a checkpoint[11]. Finally, we run the entire ROI of the application in detailed mode, collecting a memory trace of every global read, write, and atomic operation and their associated threadblock. 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 reality, the local warp scheduler will overlap computation and memory accesses by switching out warps upon cache misses.

We validated our trace-based simulator against gem5-gpu for a small number of compute units.

[11]For each application, the ROI is a single contiguous code section, run with a large enough input size such that the trace produces around 20,000 threadblocks.

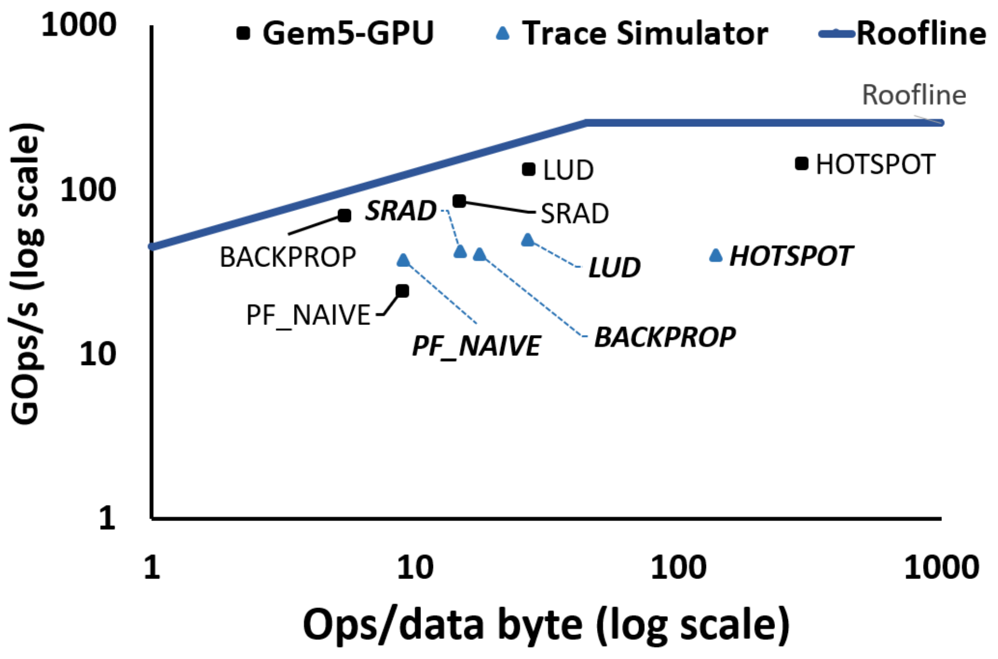


Figure 5.18: Roofline Plot Comparison between Gem5-GPU and our Trace Simulator.

Figure 5.16 compares normalized performance across the two simulators for different number of CUs.[12] Figure 5.17 compares 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[13]. 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 [221] as visual representations of the interplay of bandwidth, data locality and computation resources in the two simulators. Visual inspection of Figure 5.18, roofline plots of an 8 CU system, show the same general characteristics and application positioning between gem5-gpu and our trace-based simulator. This builds further confidence that application characteristics, such as compute to memory ratio, data locality and

[12]We were unable to generate validation data for *bc* and *color* as the workload datasets were too large to finish on our gem5-GPU setup

[13]Coherency, operating system effects, threadblock scheduling runtime and sophisticated memory coalescing schemes are not modelled accurately in the trace simulator leading to errors. Also, optimizations such as using local shared memory or warp scheduling techniques to minimize memory latency are not modelled.

bandwidth bottlenecks are preserved in our trace-based methodology.

For the results in Section 5.7, we evaluate a 24 GPM waferscale GPU running at 575 MHz and a 40 GPM waferscale GPU running at 408.2 MHz, following from the VFS scaling explored in Section 5.4. We compare against a single MCM-GPU, a 6 package, 24 GPM MCM-GPU and a 10 package, 40 GPM MCM-GPU. Our baseline scheduler is dynamic scheduling similar to the one proposed in [186, 222], i.e., round-robin within GPM first and first touch page placement, where a page is mapped to the GPM from which it is first accessed. Our systems are evaluated on the five benchmarks from Rodinia [200] and two irregular workloads from Pannotia suite [185] which we could run successfully on gem5-gpu (Table 5.9). We use a mesh network topology for the waferscale inter-GPM networks and for scale-out MCM-GPU, we consider the MCM packages interconnected using an on-board mesh network.

Table 5.9: 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
color	Pannotia	Graph Coloring
bc	Pannotia	Social Media

5.7 Results

To quantify performance and energy benefits of waferscale GPU architectures, we compare the benefits of our proposed waferscale architectures against 24 and 40 GPM systems built using multiple MCM-GPUs where each MCM-GPU houses 4 GPMs, alongside the local DRAMs. The MCM-GPUs are assumed to be integrated using conventional PCB-based integration technology. Comparisons are done based on two scheduling and data-placement policies. One is online distributed scheduling of thread group (round robin within GPM first) coupled with first touch page placement as proposed in [186] (RR-FT) and the other is our offline partitioning and placement approach (MC-DP) (Section 5.5).

We first compare the results using MC-DP. Figures 5.19 and 5.20 show comparative raw performance and EDP results for different configurations across a variety of benchmarks when using MC-DP. We see that applications such as *backprop, hotspot and particlefilter_naive* show performance benefit, up to 4.9x and 6.1x speedup, from 24-GPM and 40-GPM systems respectively implemented using MCM-GPU over a single MCM-GPU (4-GPM) . However, applications like *lud and color* show performance degradation when run on either MCM-24 or MCM-40 systems. This is because, large memory footprint and irregular access patterns of these applications cause significant inter-MCM data transfers which ultimately dominate performance scaling.

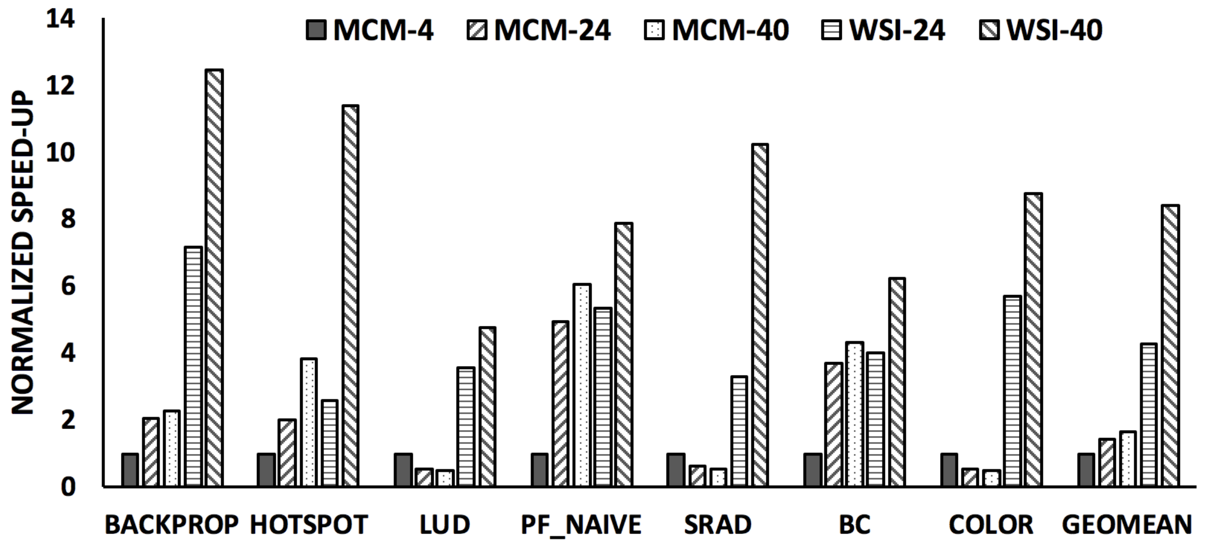


Figure 5.19: Performance improvement for Waferscale GPUs vs MCM package based conventional systems.

While scheduling on multi-MCM GPUs was first performed within a GPM in a round robin fashion, larger kernels were still split and spread across multiple MCM modules for load balancing. Also, as the number of MCM modules in the network increases, percentage of multi-hop communication increases, which further gets bottle-necked by the inter-MCM on-board bandwidth.

On the other hand, all the applications see substantial speedup from both 24-GPM and 40-

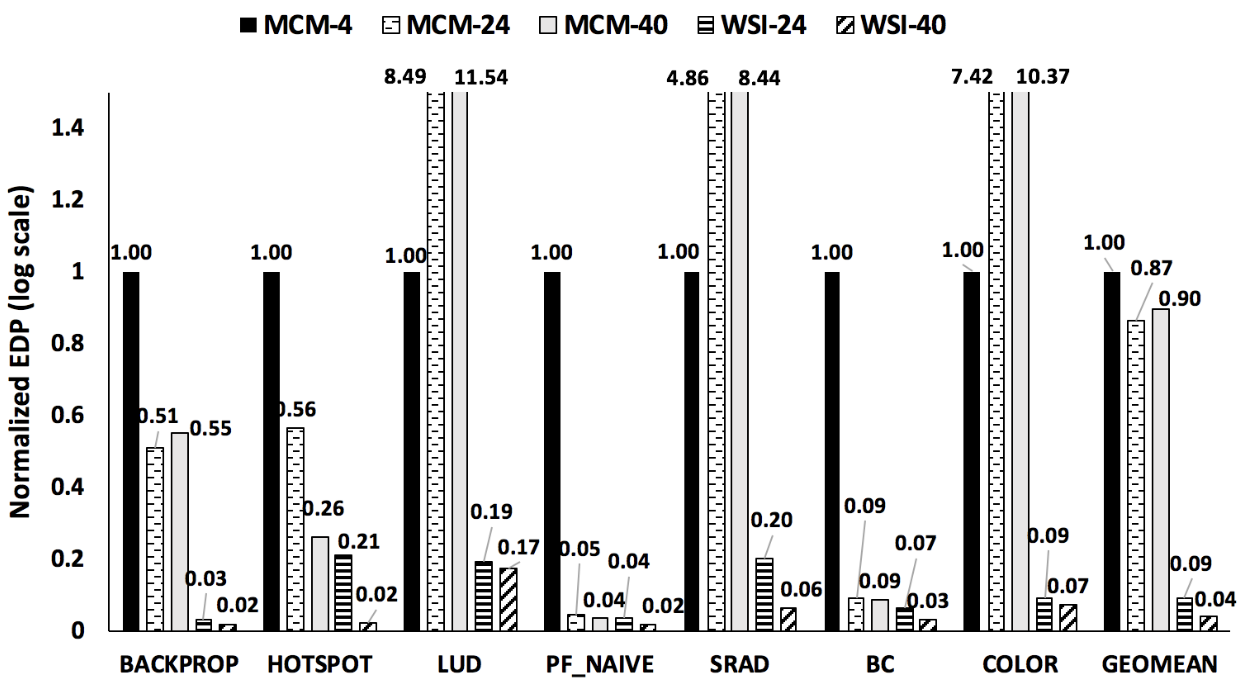


Figure 5.20: EDPs for waferscale GPUs vs MCM package based conventional systems.

GPM waferscale GPU architectures, up to 10.9x (2.97x, on average) and 18.9x (5.2x, on average) respectively when compared to comparable MCM-GPU based architectures. Such large speedups are because of the very high inter-GPM bandwidth available across the entire wafer. Similarly, average EDP benefit of 9.3x and 22.5x can be obtained from waferscale integration. This is because of the major reduction in execution time as well as 10x better energy efficiency of inter-GPM communication.

Moving on to the RR-FT policy, it is seen that the performance gap between waferscale systems and MCM-GPU based systems is about 2x higher when using RR-FT over MC-DP. This reduction in performance gap between MCM-GPU and waferscale system when going from RR-FT to MC-DP means that scale-out MCM-GPU based systems benefit more than waferscale based systems when using MC-DP. This is because in MCM-GPU based systems, the cost of inter-MCM on-board communication is much higher than that in waferscale systems. As a result, our offline policy, which helps to reduce this communication through intelligent scheduling and data placement, brings significant improvement in the performance of the scale-out MCM-GPU systems. Thus, our offline policy is well suited for scale-out MCM-GPU systems as well.

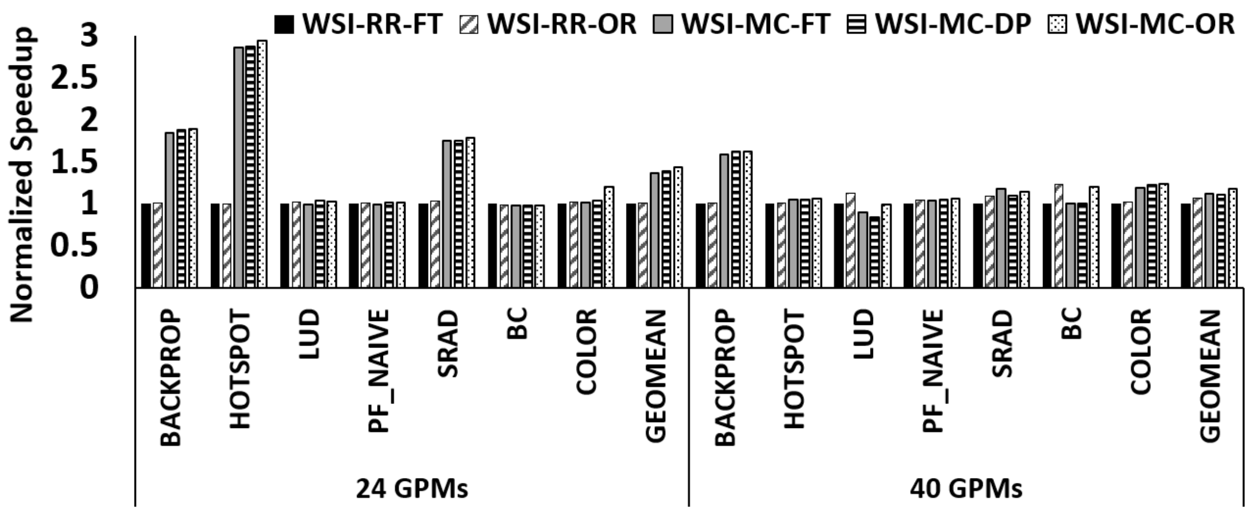


Figure 5.21: Performance of different scheduling and data placement policies.

We now perform a more detailed comparison of our offline partitioning and placement approach with the baseline locality-aware distributed scheduling policy for waferscale based systems. We evaluate the baseline policy with a realistic first touch page placement (RR-FT policy) as well as with an oracular data placement (RR-OR) that guarantees either that the remote accesses have no overhead or that there are no remote accesses. We simulated RR-OR by putting all DRAM pages in all the GPMs' local DRAM in our simulations. Similarly, we considered three variants of our offline approach. In the first case (MC-FT), only the thread block schedules were used from the offline partitioning results alongside a first-touch page placement policy. In the second case (MC-DP), we considered the data placement output from our partitioning and placement framework. In the third case (MC-OR), we considered the maximum speedup possible using these thread block schedules, i.e., again all DRAM pages were placed in all the GPMs' local DRAM.

As shown in Figure 5.21, we observe that a runtime RR-FT policy performs on an average 7% worse compared to RR-OR policy; this is similar to the observation made in [186] in context of MCM-GPUs. We also observe that our partitioning and GPM placement policy significantly outperforms RR based policies (both FT and oracular). For applications such as *backprop, hotspot, srad*, large performance benefits of up to 2.88x are attainable over RR-FT for the 24 GPM case. Benefits are upto 1.62x in 40 GPM case. Such large benefits come from the fact that our partitioning scheme clusters together the thread blocks which touch the same DRAM pages. This minimizes the

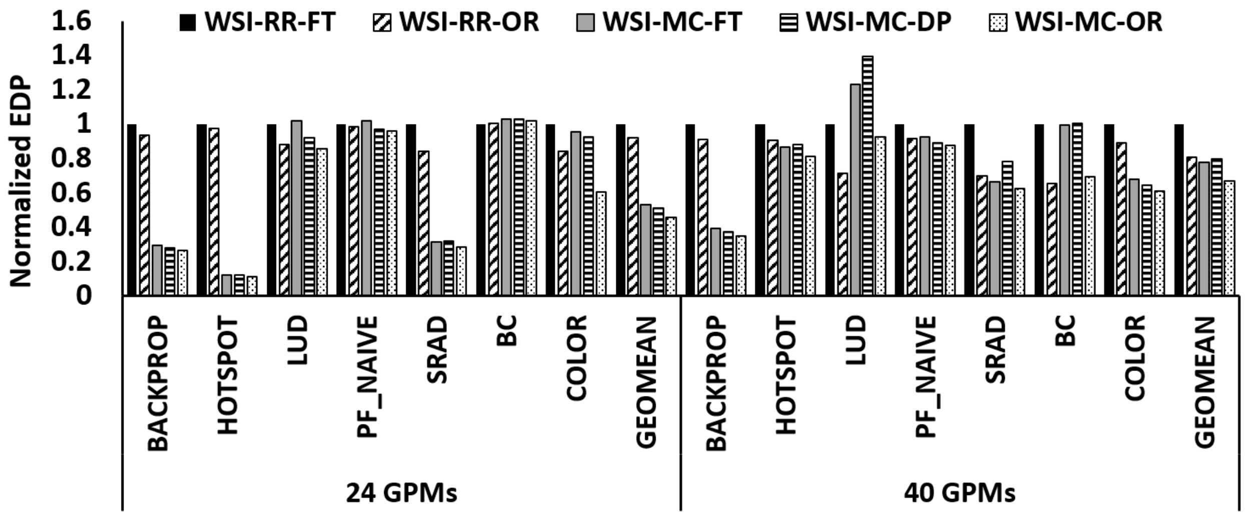


Figure 5.22: EDP of different scheduling and data placement policies.

total number of remote accesses made; this also makes caching more effective because data locality within GPM is strongly preserved and therefore cache hit rate increases. Overall, on an average, our offline policy (MC-DP) results in 1.4x (24 GPMs) and 1.11x (40 GPMs) benefits over the baseline RR-FT policy and comes within 16% of maximum speed up possible (MC-OR). This results in an average EDP benefit of 49% and 20% respectively for 24 and 40 GPM systems respectively (shown in Figure 5.22).This also indicates that online scheduling optimizations based on programmer and compiler hints [223, 224] can help achieve better performance on WS-GPU than the RR-FT online policy. Note that the relative benefit of MC-DP over RR-FT is smaller for the 40 GPM system compared to 24 GPM system. This can be attributed to the fact that TBs get distributed across more number of GPMs as the system size grows and therefore the benefit of caching decreases.

As mentioned before, we use 575 MHz as the operating frequency of the GPMs. At higher GPM frequencies, WSI-GPU benefits increase since communication is more of a bottleneck when the GPMs are running faster as frequency of data access requests increase. E.g., on average, WS-GPU-24 outperforms MCM-24 by an additional 7% at 1GHz vs 575MHz.

For the topology with 40 GPMs, we also considered non-stacked configuration. In a non-stacked configuration, GPMs need to be run at even lower voltage and frequency (0.71V/360MHz). The resulting system has 14% lower performance on average compared to the stacked configuration.

Finally, our thermal analysis assumed efficient forced air cooling model. Liquid or phase change cooling solutions can increase the sustainable TDP, enabling even higher compute density [225]. We estimate that a 2X increase thermal budget from liquid cooling can increase performance of WS-GPU-40 by additional 20-30% compared to baseline MCM-40.

5.8 Summary and Conclusion

Waferscale processors can dramatically reduce communication overheads. However, they have had unacceptable yield. Emerging integration technologies such as a Silicon-Interconnection Fabric (Si-IF)-based integration [4, 37], where pre-manufactured dies are directly bonded on to a silicon wafer, may enable waferscale processors without the corresponding yield issues. Therefore, time is ripe to revisit waferscale architectures. In this work, we showed that it is feasible and useful to architect a modern day waferscale system. Using a waferscale GPU as a case study, we showed that while a 300 mm wafer can house about 100 GPU modules (GPM), only a much scaled down GPU architecture with about 40 GPMs can be built when physical concerns are considered. We also studied the performance and energy implications of waferscale architectures. We showed that waferscale GPUs can provide significant performance and energy efficiency advantages (up to 18.9x speedup and 143x EDP benefit compared against equivalent MCM-GPU based implementation on PCB) without any change in the programming model. We also developed thread scheduling and data placement policies for waferscale GPU architectures. Our policies outperformed state-of-art scheduling and data placement policies by 2.88x (average 1.4x) and 1.62x (average 1.11x) for 24 GPM and 40 GPM cases respectively. Finally, we built the first Si-IF prototype with interconnected dies - the 100% yield we observe for our prototype, coupled with high bond yield reported for previous Si-IF prorotypes, demonstrates the technological readiness for building waferscale GPU architecture.

CHAPTER 6

Designing a 2048-Chiplet, 14336-Core Waferscale Processor

Waferscale processor systems can provide the large number of cores, and memory bandwidth required by today's highly parallel workloads. One approach to building waferscale systems is to use a chiplet-based architecture where pre-tested chiplets are integrated on a passive silicon-interconnect wafer. This technology allows heterogeneous integration and can provide significant performance and cost benefits. However, designing such a system has several challenges such as power delivery, clock distribution, waferscale-network design, design for testability and fault-tolerance. In this work, we discuss these challenges and the solutions we employed to design a 2048-chiplet, 14,336-core waferscale processor system.

6.1 Introduction

The proliferation of highly parallel workloads such as graph processing, data analytics, and machine learning is driving the demand for massively parallel high-performance systems with a large number of processing cores, extensive memory capacity, and high memory bandwidth [198,226]. Often these workloads are run on systems composed of many discrete packaged processors connected using conventional off-package communication links. These off-package links have inferior bandwidth and energy efficiency compared to their on-chip counterparts and have been scaling poorly compared to silicon scaling [9]. As a result, the overhead of inter-package communication has been growing at an alarming pace.

Waferscale integration can alleviate this communication bottleneck by tightly interconnecting a large number of processor cores on a large wafer. Multiple recent works have shown that waferscale processing can provide very large performance and energy efficiency benefits [11,227] compared to conventional systems. Recently, Cerebras has successfully commercialized a waferscale compute engine. Similarly, in the BrainScaleS/FACETS [228] project, a waferscale brain emulation engine was built. These approaches rely on building one large monolithic waferscale chip with custom cross-reticle interconnections. Monolithic waferscale chips are, however homogeneous, and so cannot integrate components from heterogeneous technologies such as DRAM or other dense memory technologies. Moreover, in order to obtain good yields, redundant cores and network links need to be reserved on the waferscale chip.

A competing approach to building waferscale systems is to integrate pre-tested known-good chiplets (in this work, we call un-packaged bare-dies/dielets as chiplets) on a waferscale interconnect substrate [11]. Silicon interconnect Fabric (Si-IF) is a candidate technology which allows us to tightly integrate many chiplets on a high-density interconnect wafer [229]. Si-IF technology provides fine-pitch copper pillar based ($10\mu m$ pitch) I/Os which are atleast 16x denser than conventional μ-bumps used in an interposer based system [230], as well as $\sim 100\mu m$ inter-chiplet spacing. Therefore, it provides global on-chip wiring-like characteristics for inter-chiplet interconnects. Moreover, in a chiplet-based waferscale system, the chiplets can be manufactured in heterogeneous technologies and can potentially provide better cost-performance trade-offs. E.g., TBs of memory capacity at 100s of TBps alongside PFLOPs of compute throughput can be obtained which is suitable for big-data workloads in HPC and ML/AI.

Large scale chiplet assembly based system design, however, has its unique set of challenges which encompass a wide range of topics from the underlying integration technology to circuit design and hardware architecture, and their impact on software. This work, for the first time, attempts to build a fine-grained chiplet-based waferscale processor prototype. The system comprises an array of 1024 tiles, where each tile is composed of two chiplets, for a total of 2048 chiplets and about 15,000 mm^2 of total area.

The scale of this prototype system forced us to rethink several aspects of the design flow. Because this is the first attempt at building such a system, there were several unknowns around the manufacturing and assembly process. As a result, fault tolerance and resiliency, was one of the primary drivers behind the design decisions we took. We also ensured that the design decisions were not too complex, such that they could be reliably implemented by a small team of graduate students within a reasonable amount of time. The several challenges we faced while architecting and designing this system are as follows:

(1) How should we deliver power to all the flip-chip bonded chiplets across the wafer?

(2) How can we reliably distribute clock across such a large area?

(3) How can we design area-efficient I/Os when a large number of fine-pitch copper pillar-based I/Os need to be supported per chiplet, and how do we achieve very high overall chiplet assembly and bonding yield?

(4) What is the inter-chip network architecture and how do we achieve resiliency if a few chiplets fail?

(5) What is the testing strategy when I/O pads have small dimensions and how do we ensure scalability of the testing schemes?

(6) How can we design the chiplets and the substrate with the uncertainty and constraints of the manufacturing process ?

In this work, we explain the challenges and possible solutions (including design decisions for our prototype system) for building scale-out chiplet-assembly based systems. To the best of our knowledge, **this is the largest chiplet assembly based system ever attempted**. In terms of active area, our prototype system is about 10x larger than a single chiplet-based system from NVIDIA/AMD etc. [231, 232], and about 100x larger than the 64-chiplet Simba (research) system from NVIDIA [233].

6.2 Overview of the Waferscale Processor System

To understand the design and implementation challenges as well as the opportunities when using a chiplet based waferscale processor system, we architected and designed a 2048-chiplet based 14,336 core processor system. Here, we first provide a brief overview of the architecture of the overall system, the chiplets and the intra-chiplet and inter-chiplet network.

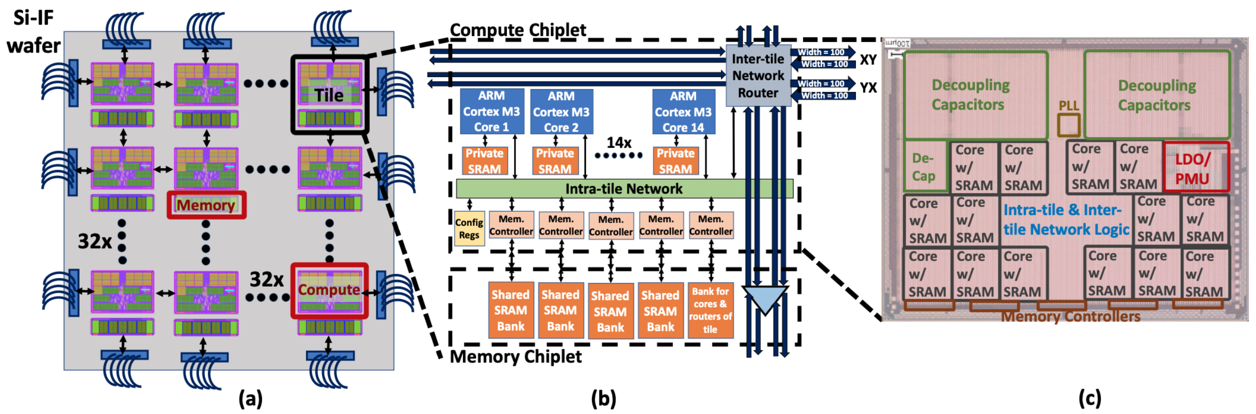


Figure 6.1: (a) Waferscale Processor System Overview showing 32x32 tile array where each tile comprises of a compute chiplet and a memory chiplet. (b) Detailed overview of the compute and memory chiplets. (c) Micrograph of the compute chiplet.

Table 6.1: Salient Features of the Waferscale Processor System

# Compute Chiplets	1024	**# Memory Chiplets**	1024	**# Cores per Tile**	14
Compute Chiplet Size	3.15mm x 2.4mm	**Memory Chiplet Size**	3.15mm x 1.1mm	**Network B/W**	9.83 TBps
Private Memory per Core	64KB	**Total Shared Memory**	512 MB	**Total # Cores**	14336
Compute Throughput	4.3 TOPS	**Shared Memory B/W**	6.144 TB/s	**# I/Os per Chiplet**	2020(C)/1250(M)
Total Area (w/ edge I/Os)	15100 mm^2	**Nominal Freq./Voltage**	300 MHz/1.1V	**Total Peak Power**	725W

Overall System Architecture We designed a scalable tile-based architecture for our system. This architecture can scale to a 32×32 tile array (see Figure 6.1), for a total of 1024 tiles. Each tile is comprised of two chiplets: a *Compute* chiplet and a *Memory* chiplet. It contains a total of 14 independently programmable processor cores and 512KB of globally shared memory. We architected this system as a unified memory system where any core on any tile can directly access the globally shared memory across the entire waferscale system using the waferscale interconnect network. Scaling a network across 14,336 cores however is challenging. Therefore, we designed a hierarchical network scheme with an intra-tile crossbar network and a waferscale inter-tile mesh network. The salient features of the system are listed in Table 6.1.

The chiplets are designed and fabricated in the TSMC 40nm-LP process and terminated at the top copper metal layer where the fine-pitch I/O pads were built. The waferscale substrate is a passive substrate containing the interconnect wiring between the chiplets and copper pillars to connect to the chiplet I/Os. The chiplets are flip-chip bonded on to the waferscale substrate as shown in Figure 6.3 and we would connect the entire waferscale system to the power supply and external controllers using edge connectors.

Compute Chiplet As shown in Figure 6.1, the compute chiplet contains 14 ARM CORTEX-M3 cores and their private SRAMs (64KB each), memory controllers (to access the banks in the memory chiplet), network routing infrastructure for inter-tile network and a chiplet-level intra-tile crossbar interconnect (implemented using ARM BusMatrix IP) to connect all these components. The power delivery related components are also contained within the compute chiplet. The network routing infrastructure was built around the open-source BSG IPs [234], but includes other custom units needed to support two independent networks, adapters to communicate with the intra-tile network and support various memory-mapped functionality.

Memory Chiplet The memory chiplet comprises five 128KB SRAM memory banks. Four of these banks are addressable using the global shared memory address space while one bank can be accessed only by the cores and network routers on the same tile. All these banks can be accessed in parallel and are connected to the intra-tile network through the memory controllers on the compute chiplet. The memory chiplet also provides buffered feedthroughs for the north-south interconnect links and two banks of decoupling capacitors. Note that though the two chiplets in a tile are architecturally heterogeneous, we implemented them in the same technology node for ease of design. However, this chiplet can be easily implemented in a newer or denser memory technologies for higher memory capacity and/or area savings.

Features of the waferscale substrate The waferscale substrate is built using the Si-IF technology. The I/O (i.e., copper pillar) pitch we use in our prototype is $10\mu m$ (minimum that the technology offers). The interconnect wiring pitch is $5\mu m$ (minimum offered is currently $4\mu m$). With two layers of signaling, the edge interconnect density we achieve is 400 wires/mm.

We validated the system design and architecture discussed in this work by building and testing a reduced-size multi-tile system. Figure 6.2 shows the current prototype system. We were also

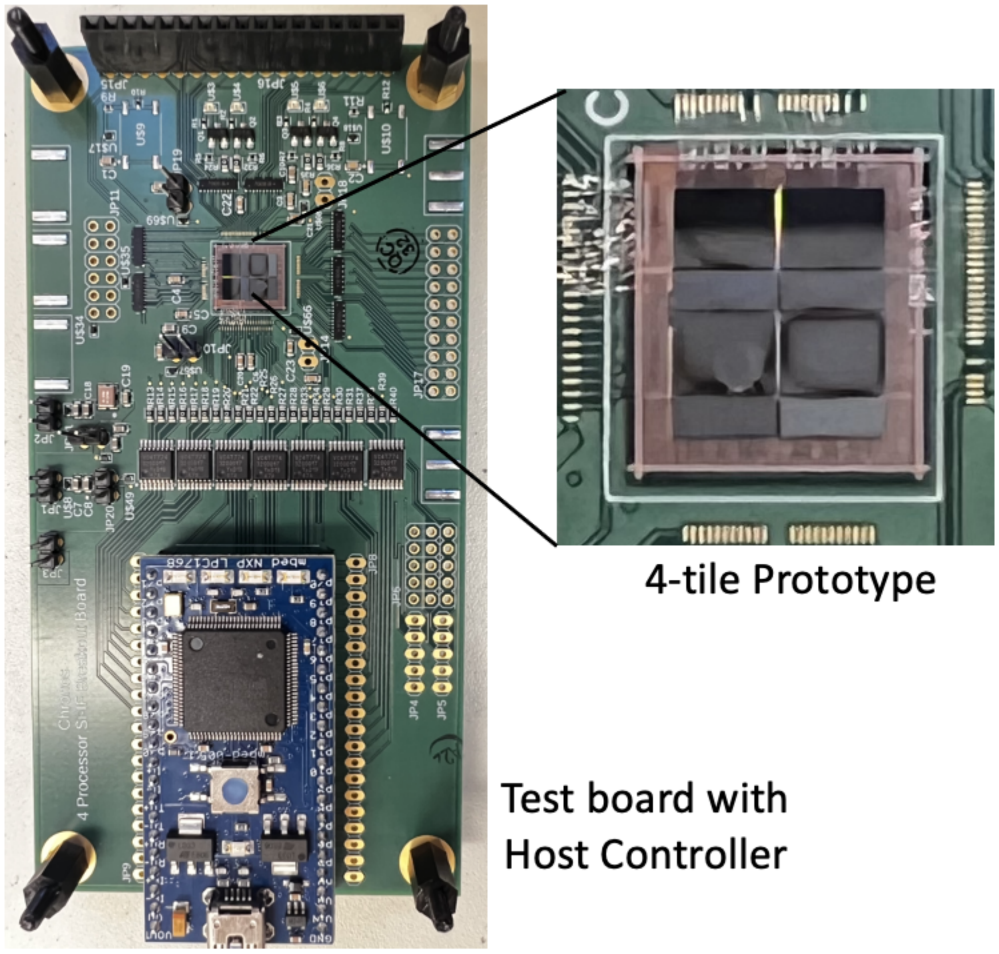


Figure 6.2: A 4-tile prototype system is shown.

successfully able to run various workloads including graph applications such as breadth-first search (BFS), single-source shortest path (SSSP), etc. on this system. Next, we will discuss each of our design decisions in detail.

6.3 Waferscale Power Delivery and Regulation

Here, we ask the question: How to deliver power reliably to all the chiplets which are flip-chip bonded on to a $\sim 15000 mm^2$ large waferscale interconnect substrate? Unlike a monolithic waferscale system where the power can be directly supplied to the top-most metal layer (face side), the chiplets are flip-chip bonded on to the thick waferscale substrate. As a result, either power can be delivered through the backside of the wafer using through-wafer-vias (TWVs) [235], which are $700\mu m$ deep vias across a full-thickness wafer, or can be delivered at the edge of the wafer. Since the integration of TWV technology in a Si-IF wafer is still under development and not ready for prime-time yet, we chose to use edge power delivery for our prototype system.

The peak power per tile is about 350mW when operating at a voltage of 1.21V (fast-fast corner). Therefore, about 290A of current needs to be delivered to the chiplets across the wafer. The number of metal layers in the substrate is restricted to four in order to maximize yield. Since two metal layers are dedicated to inter-chip signaling, two layers are available for power distribution. Maximum thickness of metal layers in the Si-IF technology is $2\mu m$ and thus, the resistance of the power distribution network would result in large voltage droop if the current that needs to be delivered is very large. As such, we considered two different strategies for power delivery: (1) High voltage (say 12V) power delivery at the edge and using down conversion (buck or switched capacitor based converters) near the chiplets [11], which would lower the current delivered through the power planes by $\sim$12x, (2) Higher voltage power delivery (say 2.5V) at the edge and using low-dropout (LDO) based regulation in the chiplets, which would mean that a larger amount of current needs to traverse the PDN planes and, therefore, would sustain larger losses in the power delivery planes.

The first option of using down-conversion near the chiplets has high area overheads because bulky off-chip components such as inductors and capacitors need to be placed on the wafer. We estimate that about 25-30% of the area would be occupied by these components. Moreover, integrating these components on the wafer would result in disruption of the regular structure of the chiplet array and increase the inter-chiplet distance, which would diminish the benefits of fine-pitch interconnects. This scheme would result in increased design complexity.

Since, this prototype is a sub-kW system, we chose to avoid this complexity in lieu of some power efficiency loss coming from the resistive power loss and poorer LDO efficiency. In our scheme, the chiplets near the edge would receive power at much higher voltage (2.5V) and the chiplets away from the edge would receive power at lower voltage due to resistive power loss related voltage droop. Our estimates show that the chiplets at the center would receive power at roughly 1.4V during peak power draw from all the chiplets. This however, makes the LDO design challenging as it has to produce a stable voltage of 1.1V (nominal) for the logic devices while the DC supply voltage can vary between 1.4V and 2.5V depending on where the chiplet is placed on the wafer. We

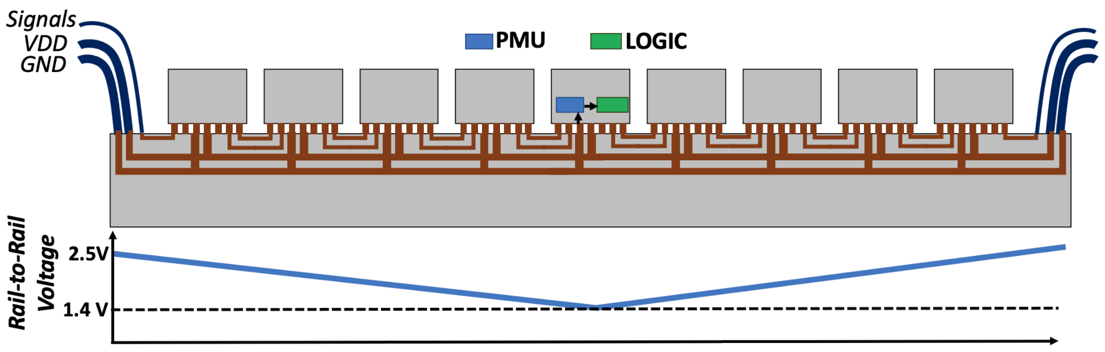


Figure 6.3: Power is delivered from the edge. The chiplets at the edge of the wafer receive power at 2.5V. There is voltage droop as we move towards the center of the wafer and the chiplets at the center receive power at 1.4V.

built a custom LDO which can track this wide input voltage range.

The other challenge is that the LDO regulator has to support up to 350mW of peak power while sustaining up to 200mA current demand fluctuation (worst case) within a few cycles. In order to achieve good regulation under these operating conditions, the LDO regulator needs sufficient decoupling capacitance at the output. Such high capacitance requirements are usually fulfilled using off-chip discrete decoupling capacitors. However, in our waferscale system, off-chip capacitors can only be placed around the edge of the array. As a result, the chiplets at the center of the array can be as far as 70 mm away from the nearest capacitor. Hence, we designed a custom on-chip decoupling capacitor and dedicated ∼35% of the total tile area to decoupling capacitance giving about 20 nF per tile. In the future, incorporation of deep trench decoupling capacitors [236] (currently under development) in to the waferscale substrate has the potential to significantly improve PDN performance and will also reduce the area overhead of on-chip decoupling capacitors. The eventual design ensures that the regulated voltage is always between 1.0V and 1.2V across process/voltage/temperature corners. We omit the circuit level details for brevity.

6.4 Waferscale Clock Generation and Distribution

Next, we ask the question: How do we provide clock to all the chiplets across the >15,000mm^2 waferscale substrate?

We included a phase-locked loop (PLL) in the compute chiplet which can take an input clock with frequency between 10 and 133MHz and generate an output clock with frequency up to 400 MHz. Therefore, one option is to distribute a slow clock across the wafer using a passive clock distribution network (CDN) built on the Si-IF. However, there are two challenges in such a scheme. First, the parasitics of a passive CDN which spans an area of about 15,100 mm^2 and has 1024 sinks are very large (>450pF and >120nH). So, the clock distribution can only be done at sub-MHz frequency. Also, getting a good crystal oscillator which can drive large capacitive load while ensuring absolute jitter performance of sub-100 pico-seconds is hard. Second, the PLL IP we used requires a stable reference voltage for reliable operation. However, the voltage regulation in the chiplets away from the edge is not perfect and the regulated voltage could fluctuate between 1.0V and 1.2V. As a result, stable clock can only be generated near the edge of the wafer where the chiplets can access near-by off-chip decoupling capacitors. Therefore, in this system, a fast clock (up to 350 MHz) will be generated in one of the edge tiles and then forwarded throughout the tile array using forwarding circuitry built inside every tile. Next, we briefly describe the clock selection and forwarding circuitry.

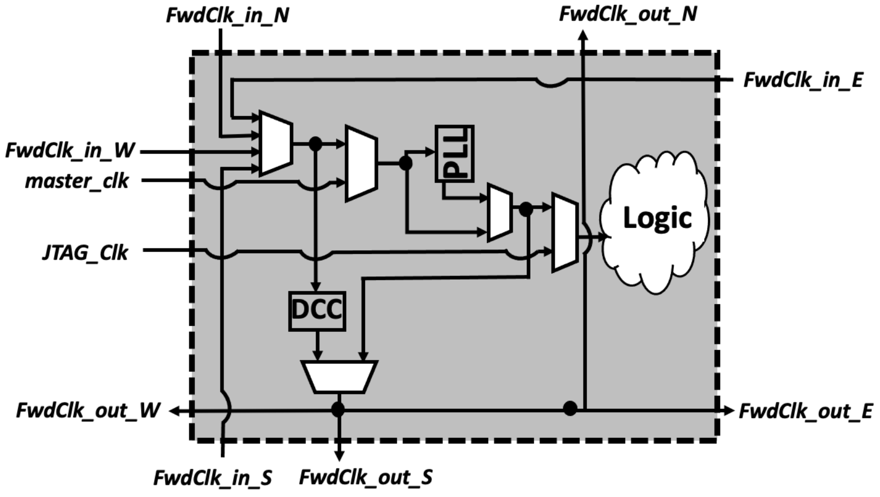


Figure 6.4: Schematic of clock selection and forwarding circuitry

Clock Selection and Forwarding

The clock selection and forwarding circuitry is a part of the compute chiplet. As shown in Figure 6.4, the compute chiplet has multiple clock inputs: master (slow) clock, software-controlled test/JTAG clock and four forwarded clocks (one from the neighboring chiplet on each side); and four outputs

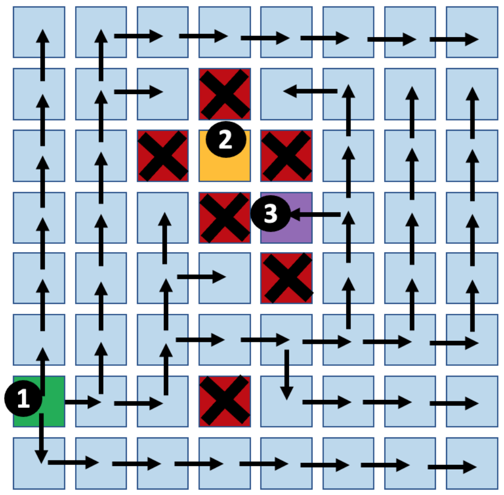


Figure 6.5: A clock forwarding configuration is shown for a system with faulty tiles. All tiles except the yellow one can receive the forwarded clock.

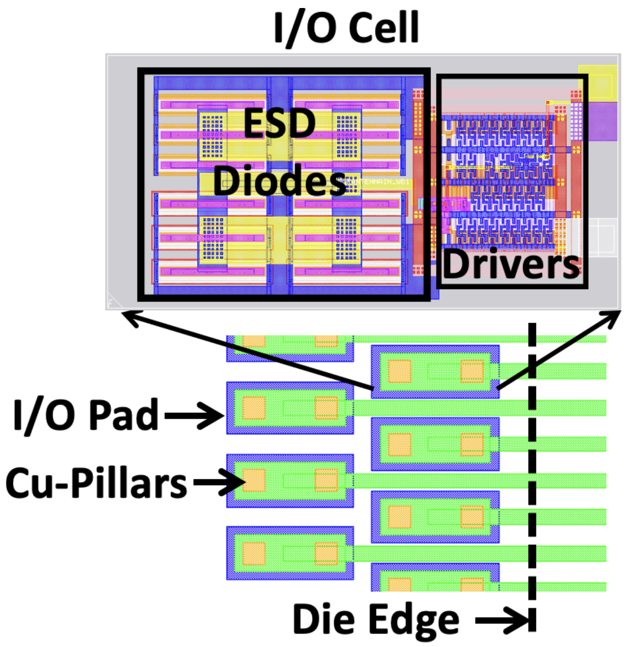


Figure 6.6: Fine-pitch I/O layout with ESD protection circuitry and two Cu-pillars per I/O pad.

to forward a clock to the neighboring chiplets on all sides. During testing and program/data loading phases, the JTAG clock is selected as the functional clock for the tile. During the program execution phase however, either one of the four forwarded input clocks or the master clock can be selected as the functional clock for the tile logic. If the frequency of the selected tile clock needs to be multiplied, it can be optionally passed on to the PLL. Moreover, one of these five clocks is selected to be forwarded to all the neighboring tiles.

During boot-up, the clock selector circuitry defaults to the software-controlled JTAG clock. Using JTAG, we then initiate the *clock setup* phase. In this phase, first we select one or multiple edge tiles and configure them to generate a faster clock from the slower system clock that is provided from an off-the-wafer crystal oscillator source. The generated faster clocks from the edge chiplets are forwarded to their neighboring chiplets. The non-edge chiplets are then configured for the auto-clock selection phase. In this phase, the clock selection circuitry selects the forwarded clock which starts toggling and is the first to reach a pre-defined toggle count (default is 16). Once a forwarded clock is selected, the clock setup phase for that tile terminates and the selected clock is forwarded to its neighboring tiles. This ensures that no live-lock scenarios occur in the clock forwarding process.

However, one issue with such a clock forwarding scheme is that the fast clock can accrue duty cycle distortion because of pull-up/pull-down imbalance in the buffers, inverters, forwarding unit components and inter-chiplet I/O drivers [237]. As the clock traverses across multiple tiles in the array, this duty cycle distortion can potentially kill the clock, e.g., a 5% distortion per tile could kill the clock with in just 10 tiles. In order to avoid this issue, we forward an inverted version of the clock. This ensures that the distortion is alternated between the clock cycle halves. The half-cycle phase delay and any jitter introduced is not a concern since our inter-chiplet communication uses asynchronous FIFOs [234]. Moreover, we also implemented a duty cycle distortion correction (DCC) unit [238], which can correct any residual distortion.

Resiliency in Clock Forwarding Network

Faulty chiplets can potentially disrupt the clock forwarding mechanism. Our clock generation and forwarding scheme however, has resilience built in. Because any chiplet at the edge can generate a faster clock, there isn't a single point of failure in clock generation. Moreover, because every non-edge tile receives a toggling clock from all four directions, this ensures that if at least one of the neighboring chiplets out of the four is not faulty, then the clock can reach that chiplet and be further forwarded. By induction, it can be proven that the generated fast clock can reach all non-faulty tiles on the wafer, unless all the neighboring tiles of a specific tile are faulty.

Figure 6.5 shows one possible clock forwarding configuration for an 8x8 tile array with faulty tiles. The edge tile ❶ generates the faster clock that gets forwarded across the entire wafer. Even with 6 faulty tiles in a 64 tile mesh, all tiles, except tile ❷, receive the forwarded clock. Tile ❷ has faulty tiles on all four sides and hence, is unable to receive the generated clock. Even otherwise, this tile would have been rendered unusable since there is no available path for other tiles to communicate with this tile using the waferscale inter-tile network. On the other hand, tile ❸ can still receive the forwarded clock even when surrounded by three faulty tiles. This is because it has one non-faulty neighbor from which it receives the generated clock.

6.5 I/O Architecture

In this section, we ask the question: Since, each chiplet needs to support a large number of I/Os (transceiver circuitry and Cu pads) for fine-pitch copper pillar interconnects, how do we design area-efficient I/Os and achieve high bonding yield?

The Si-IF technology allows inter-chiplet links to be as short as 200-300 μm. As a result, the links can be easily driven by small, energy efficient I/O circuitry that can operate at 1GHz. Besides, the Si-IF technology also offers fine pitch copper pillars (10μm pitch) for bonding the chiplet on to the substrate and fine pitch interconnect wiring (4 μm pitch) for inter-chiplet communication [229]. In order to support a large number of I/Os without large area overhead, the size of the I/O cells need to be small. Moreover, if the I/O cells are large, they have to be placed at a distance from the I/O pads, thereby, significantly reducing the energy benefits of short inter-chiplet Si-IF links. Therefore, if the I/O circuitry can be completely encompassed under the pad, it would enable us to obtain optimal energy efficiency for the I/O transceivers. The transmitter was designed using simple appropriately-sized cascaded inverters which can drive signals at 1GHz for link length of up to 500μm. The receiver was designed using two minimum sized inverters. Managing electrostatic discharge (ESD) in small I/O cell area is a challenge but fortunately, unlike packaged parts which usually have to deal with large ESD events corresponding to 2kV human body model, bare-die chiplet to wafer bonding only needs to address the less stringent 100V human-body model (HBM) or machine-model (MM) specifications [239] (similar to silicon interposers).

The final area of the I/O cells, along with the stripped down ESD circuitry, was about $150\mu m^2$. This is larger than the area that could be accommodated under one copper pillar. Therefore, we designed the I/Os such that two copper pillars can land on each pad. This also enhances the bonding-related yield. For a single pillar, the expected bonding yield is $>$99.99% [229]. With two pillars per pad, per-I/O bonding yield can be improved significantly. With over 2000 I/Os per chiplet, bonding yield for a chiplet would therefore improve from 81.46% to 99.998%. This is critical for system yield since our waferscale system comprises of 2048 chiplets (i.e., at the wafer-level this would reduce expected number of faulty chiplets from 380 down to 1). As shown in Figure 6.6, in order to achieve the maximum I/O density per mm of chiplet edge, the I/O pads were placed such that the two pillars landing on each I/O pad would be orthogonal to the chiplet edge. Overall, this I/O design is area-efficient (total I/O area is only $0.4mm^2$), energy-efficient (0.063pJ/bit) and improves system yield dramatically.

6.6 Waferscale Network Architecture and Resiliency

The next question we ask is: How do we architect the waferscale network and ensure good connectivity among working chiplets when a small number of chiplets may fail?

We use a mesh network to connect the chiplets across the wafer. The network routers reside on the compute chiplet. In order to avoid deadlocks, we use dimension-ordered routing (DoR). However, as mentioned in Section 6.5, even with an excellent bonding yield of 99.998% per chiplet, the overall system of 2048 chiplets might have one or few faulty chiplets. Using Monte-Carlo simulation we estimate the percentage of source destination pairs that will get disconnected if there is a single path between any pair of chiplets. As shown in Figure 6.7, with just five faulty chiplets (out of 2048) in the wafer, $>$12% of paths get disconnected.

In order to overcome this issue, we designed two independent networks across the wafer; one

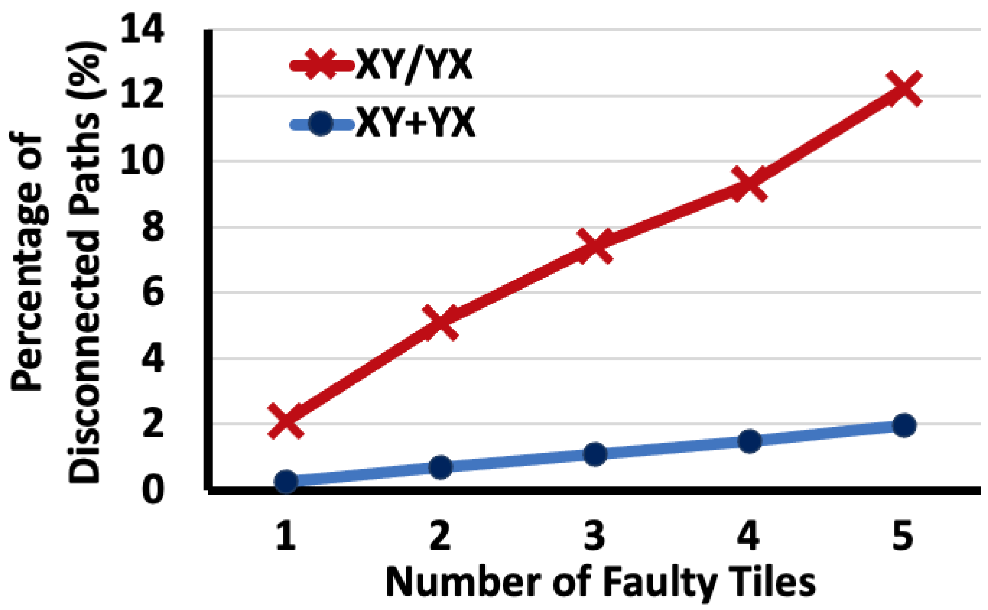


Figure 6.7: Average percentage of disconnected source-destination pairs for the conventional scheme with one DoR network versus when two networks are used. This result is obtained using a set of randomly generated fault maps.

with X-Y dimension-ordered routing and the other with Y-X dimension ordered routing as shown in Figure 6.8. With this, most chiplet pairs (all pairs where the two chiplets are not in the same row/column) on the wafer have two distinct paths between them. This dramatically reduces the number of paths that get disconnected when a certain number of chiplets on the wafer are faulty. For example, with five faulty chiplets on the wafer, the percentage of disconnected paths reduces from $>$12% to $<$2% as we go from a single DoR network to two independent DoR networks on the wafer. The paths that still get disconnected with two DoR networks mostly connect those pairs of chiplets that are in the same row/column. Moreover, in our system, network request-response communication happens using the complimentary networks (this is baked in to the router hardware). As shown in Figure 6.8, if a request from chiplet A to chiplet B is sent along the X-Y direction, the response from B-to-A is sent in the Y-X direction in order to ensure that the same path is taken by the request-response pair. This makes sure that two-way communication between chiplets is possible whenever one non-faulty path exists. This also avoids deadlocks between request-response pairs.

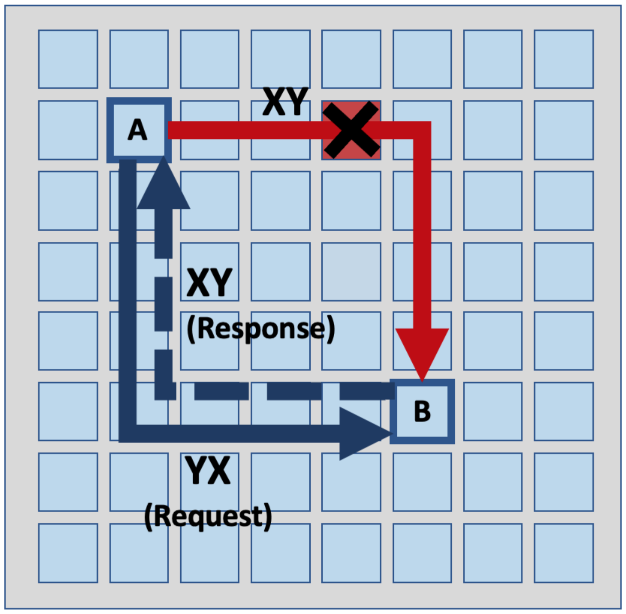


Figure 6.8: Fault-tolerant waferscale mesh network architecture.

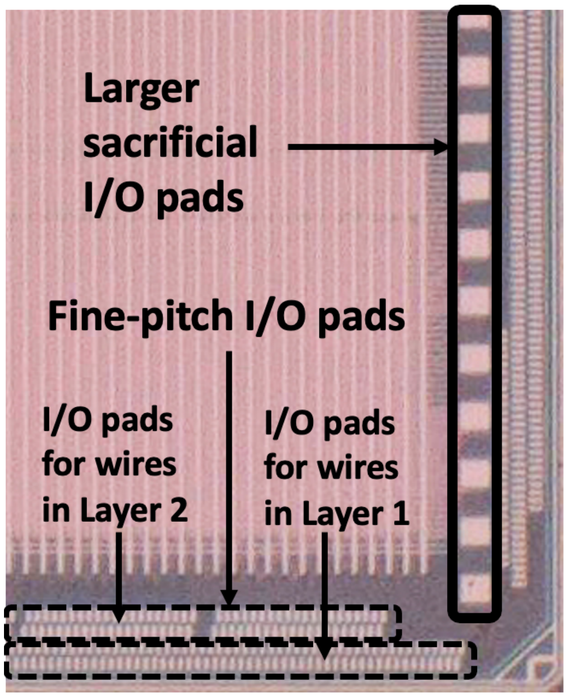

Larger sacrificial I/O pads
Fine-pitch I/O pads
I/O pads for wires in Layer 2
I/O pads for wires in Layer 1

Figure 6.9: Larger pads are for probing and fine-pitch pads are for inter-chiplet communication.

Given the length of our chiplet edge, we can support 400-bit wide parallel inter-chiplet network link escaping each side of the tile. The width of an entire packet in our case is 100 bits. Thus, we divide the inter-chiplet links into four separate parallel wide buses. Two of them are dedicated to the X-Y network and the other two are dedicated to the Y-X network. The two buses corresponding to each DoR network are ingress and egress links.

The task of choosing the correct network to use is left up to the kernel software. Once a system is fully assembled, we identify the faulty tiles and store them in a fault-map. The kernel software then uses the information of the fault map to decide the network to use for a source-destination tile pair. If both the paths are available between two tiles, the kernel software is used to distribute the source-destination pairs to the network in a way such that both the networks are equally utilized. While doing so, we ensure packet consistency (i.e., packets arrive in order) by allocating all communication between a source-destination pair to a single network only.

Additionally, we can also use kernel software to circumvent the issue of disconnected paths. Every time a packet needs to be sent through a path with a faulty tile, it can divert the packets to an intermediate tile and then route it from the intermediate tile to the final destination. The response packet will also follow the same path. However, this will require cores to allocate cycles towards network routing instead of executing the actual application process and hence, can adversely affect the overall performance. Since our solution of using two DoR networks significantly reduces the number of disconnected paths, as compared to a single network, this performance impact is expected to be minimal. In the future, we will incorporate sophisticated routing schemes [240, 241] for improved waferscale fault tolerance as well as performance.

6.7 Testing Infrastructure

Chiplet-based waferscale technology promises to provide better system yield than a monolithic approach. However, it depends on the identification of known-good dies (KGD) and reliable chiplet assembly on the Si-IF. Therefore, we ask the question: How to do design our test scheme for pre-bond testing as well as testing the system post assembly?

Test infrastructure inside a tile As shown in Figure 6.10, the ARM CORTEX-M3 core provides debug access through a Debug Access Port (DAP, based on IEEE 1149.1 JTAG protocol minus boundary scan). External communication with the DAP port is done using a JTAG interface. Each tile in our system has fourteen cores and, therefore, fourteen DAP interfaces. One option was to bring all the fourteen DAP interfaces out to the edge of the compute chiplet; and eventually to the edge of the wafer for testing (using ARM-based MBED microcontrollers, in our case). However, such a scheme would require a lot of I/Os at the edge of the wafer, e.g., for just the 32 chiplets at the edge, a 1792-bit interface is needed. Though handling this many I/Os at the edge is possible using advanced connectors or by using serialization/de-serialization at the edge of the wafer, this was beyond the scope of this work. Therefore, we daisy-chained all the DAP interfaces inside the compute chiplet (as shown in Figure 6.10) and so, only one JTAG interface is required to connect to the multiple DAPs in a chain. We also provision the daisy-chain such that it can be extended to include DAPs across multiple chiplets.

During testing, all the cores would usually run the same set of test instructions. Also, upon analysis of many irregular workloads, we found that majority of the cores would actually run the same program (albeit independently). Therefore, in order to minimize the program loading time, we provision for the same program to be broadcasted to multiple cores in a tile. The optimization is to broadcast the input at TDI$_{tile}$ (Test Data In) to the TDI pin of all the DAP ports and the TDO (Test Data Out) of the first core is forwarded to TDO$_{tile}$. Thus, in this mode, the external controller sees only one DAP per tile and, therefore, the JTAG bit shifting latency reduces by 14x.

6.7.1 Pre-bond Testing

Pre-bond testing is essential for identifying KGD parts. However, there are two issues which make pre-bond testing of un-packaged chiplets designed with fine-pitch IO pads for Si-IF assembly difficult: (1) the fine-pitch IO pads with 10 μm pitch and 7 μm width are not amenable to probe-card-based testing. The probe pitch usually is larger than 50 μm. (2) Once the probes land on a pad, it damages the planarity of the pad surface which is critical for reliable subsequent direct metal-to-metal bonding.

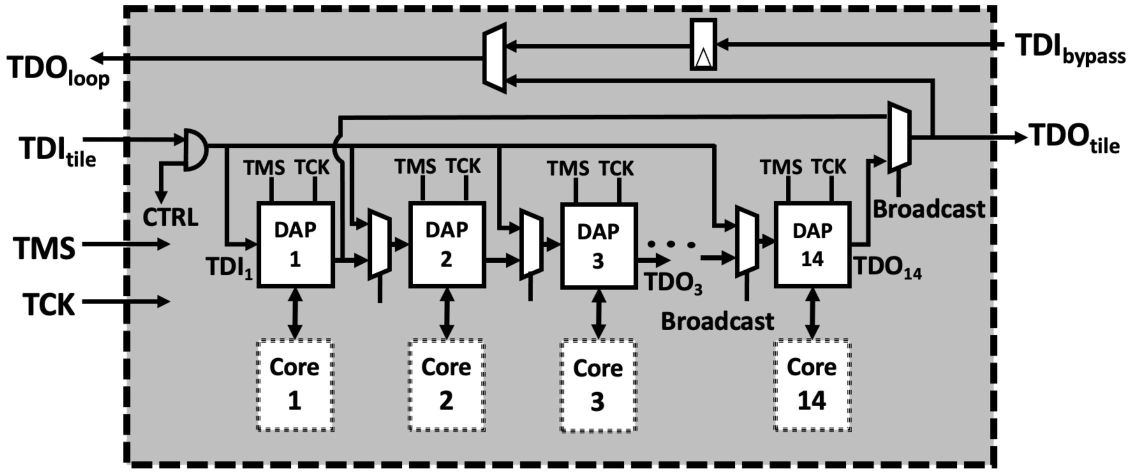


Figure 6.10: Schematic of the test circuitry inside a tile.

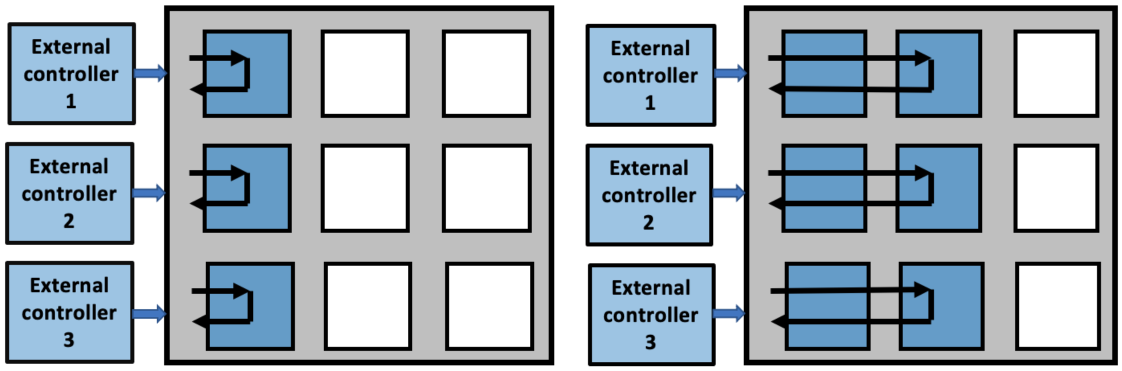


Figure 6.11: Progressive unrolling scheme is shown. The JTAG chain is unrolled progressively to identify the faulty chiplet in the chain.

Therefore, in order to get around these issues, we designed larger duplicate pads for the JTAG and some auxiliary test signals. These larger pads are designed such that probe-card testing can be done, while their fine-pitch counterparts are used for bonding to the Si-IF. Using this approach, we can thoroughly test the chiplets and eliminate faulty chiplets before bonding. Once the chiplets are tested, we ensure that, for die-to-wafer bonding, we don't use the larger pads which are probed. We only have copper pillars for bonding the fine-pitch pads which are not probed. The same JTAG interface can be used to load test routines and programs in the chiplet after bonding, but now using the fine-pitch pillars.

6.7.2 Post Assembly Testing

After pre-bond testing, the non-faulty chiplets are passed on to the die-to-wafer bonding process. Past work on fine-pitch die-to-wafer bonding [229] has shown to achieve excellent bonding yield (>99.99%). In our waferscale system, the total number of inter-chip I/Os is 3.7M+. Even with the I/O redundancy scheme described in section 6.5, a few bonding-related failures may still occur. Therefore, it is important to pin-point the location of the faulty chiplets. Moreover, since the number of chiplets to test in a waferscale system is very large, this testing process needs to be done at high throughput and demands scalability.

Progressive multi-chiplet JTAG chain unrolling As shown in Figures 6.10 and 6.11, we designed the JTAG chaining mechanism in a way where the TDO$_{tile}$ signal can either be forwarded to the next tile in the chain or can loop-back towards TDO$_{loop}$ (similar to the under-development IEEE P1838 standard [242] for 3D devices). Therefore, each chiplet in the chain can be tested progressively and independently. On power-up, the default mode is the chain loop-back mode. The first chiplet in the chain is tested first. Once the chiplet passes the test, its test mode is changed so that the TDO$_{tile}$ from the first chiplet is forwarded to the second chiplet. The second chiplet is still in the loop-back mode and, therefore, the TDO$_{tile}$ signal from the second chiplet is eventually brought out through the TDI$_{bypass}$ and TDO$_{loop}$ signals of the first chiplet to the external controller. The chain is progressively unrolled and the test procedure is repeated for all the chiplets in a chain. This helps to identify the faulty chiplet as the chain unrolls. This mechanism can also be used for during-assembly testing to intermittently check for failures in a partially bonded system. This scheme would help to identify and discard partially populated faulty systems and minimize wastage of KGD chiplets.

Multi-chain JTAG To achieve high-throughput and scalability, we adopted a multi-chain debug methodology. Instead of creating one JTAG daisy-chain with 1024 tiles, we chose to split the array in to 32 chains with each running across the rows. This has two primary benefits: (1) Testing and program/data loading to the tiles in the rows can be done in parallel. As a result, this can speedup these processes by up to 32x, speeding up loading all the memory on the system from 2.5 hours (with a single chain) to roughly under 5 minutes. (2) The TMS and TCLK signals are broadcast to the tiles in a chain. Splitting the chains allows us to have independent TMS and TCLK signals for each row and helps reduce the load on these signals; this would enable us to run these signals at up to 10 MHz.

6.8 Waferscale Substrate and its Impact on Design

For such a large integration substrate, we were unsure of the wafer substrate yield (a problem exacerbated by fabrication in a research facility). As a result, we designed the chiplet I/Os in such a way that, even with one routing layer, we would have a working processor system, albeit with reduced shared memory capacity. We have two sets of I/O columns on each side of chiplet (as shown in Figure 6.9), one set per layer of signal routing. The first set comprises of the two I/O columns closest to the die edge and consists of all the absolutely essential network link I/Os. It also comprises of I/Os corresponding to two out of the five memory banks in our memory chiplet. The other set of I/O columns that would require the second routing layer, consists of the non-essential I/Os and the ones corresponding to the remaining three memory banks. Thus, if we have just a single layer of routing, we will be able to build the entire processor system by connecting just the I/Os in the first set. The only downside would be the reduction of shared memory capacity by 60%.

Since the size of the wafer is much larger than the maximum size of a reticle, the Si-IF substrate had to be designed such that it is step-and-repeatable.This won't be a problem if a direct-write lithography system is employed but most commercial and research foundry patterning processes employ steppers. The entire wafer is divided into smaller identical reticles and is fabricated by stitching these reticles. Each reticle consists of 72 tiles (12x6). The inter-chiplet links, within each reticle have width of 2 μm and spacing of 3 μm. However, at the edge of each reticle, the links escaping are made fatter (width increases to 3 μm and spacing reduces to 2 μm), while keeping the pitch constant, in order to reduce the impact of reticle stitching error [243]. Besides, a number of I/Os from each of the tiles at the edge of the mesh needs to fan-out to the edge of the wafer and connect to the external connectors. We designed the fan-out wiring and the edge I/O pads into each reticle. The chiplet slots on the Si-IF substrate from the edge reticles would remain un-populated and the external connectors would connect to the pads in these reticles. To ensure that these I/O pads don't cause an issue where chiplets are bonded, we use a custom block etch process to remove the pads wherever they are not needed. If a foundry supports multiple reticles per wafer, the edge of the wafer can also be printed using a separate mask.

The Si-IF substrate for this processor system consists of four metal layers. The bottom two layers are built as dense slotted planes and are dedicated to power supply. The top two layers are sparse and are dedicated to inter-chiplet signal routing. The major challenge in using today's conventional tools for designing the interconnect substrate is the sheer scale of the system. The memory footprint when designing a four layer $>15000\ mm^2$ wafer using current commercial tools explodes, which, in turn, leads to very large design time. Hence, we developed our own lightweight custom router for designing the four layer waferscale substrate. The current version of the router supports only jog-free routing for inter-chiplet connections, which is sufficient for this prototype. Developing a general, scalable router for large (but low wiring density) chiplet assemblies is part of our future work.

6.9 Summary and Conclusion

Chiplet assembly is a very promising approach to build scale-out heterogeneous computing systems. We describe our experience designing largest ever such chiplet assembly based waferscale processor (at least 10X larger than largest known commercial chiplet-based systems) and develop a design methodology for the same. We highlight challenges and potential solutions in power delivery, clock distribution, network design, design for testability and fault-tolerance for waferscale systems. To best of our knowledge, this is the first work discussing design methodology challenges for large chiplet-assembly based systems. Currently, we have designed and assembled a 2x2 array of tiles and have fully verified the functionality of a tile running at 300 MHz. Our ongoing work aims at characterizing the waferscale prototype and developing design methods for higher-power waferscale systems.

CHAPTER 7

Optimizing Multi-GPU Parallelization Strategies

Deploying deep learning (DL) models across multiple compute devices to train large and complex models continues to grow in importance because of the demand for faster and more frequent training. Data parallelism (DP) is the most widely used parallelization strategy, but as the number of devices in data parallel training grows, so does the communication overhead between devices. Additionally, a larger aggregate batch size per step leads to statistical efficiency loss, i.e., a larger number of epochs are required to converge to a desired accuracy. These factors affect overall training time and beyond a certain number of devices, the speedup from leveraging DP begins to scale poorly. In addition to DP, each training step can be accelerated by exploiting model parallelism (MP). This work explores hybrid parallelization, where each data parallel worker is comprised of more than one device, across which the model dataflow graph (DFG) is split using MP. We show that at-scale, hybrid training will be more effective at minimizing end-to-end training time than exploiting DP alone. We project that for Inception-V3, GNMT, and BigLSTM, the hybrid strategy provides an end-to-end training speedup of at least 26.5%, 8%, and 22% respectively compared to what DP alone can achieve at scale.

7.1 Introduction

Deep learning (DL) models continue to grow and the datasets used to train them are increasing in size, leading to longer training times. Therefore, training is being accelerated by deploying DL models across multiple devices (e.g., GPUs/TPUs) in parallel. Data parallelism (DP) is the simplest parallelization strategy [244–246], where replicas of a model are trained on independent devices using independent subsets of data, referred to as mini-batches. All major frameworks (e.g. TensorFlow [247], PyTorch [248]) support DP using easy-to-use and intuitive APIs [249]. However, as the number of devices used to exploit DP increases, the global batch size also typically increases [1]. This poses a fundamental problem for data parallel scalability because for any given DL network, there exists a global batch size beyond which converging to the desired accuracy requires a significantly larger number of iterations. This is primarily due to the reduced statistical efficiency of the training process [250]. In addition, as the number of devices employed increases, the synchronization/communication overhead of sharing gradients across devices increases, further limiting overall training speedup.

Model parallelism (MP) is a complementary technique in which the model dataflow graph (DFG) is split across multiple devices while working on the same mini-batch [245, 251]. MP has been traditionally used to split large models (which can not fit in a single device's memory), but employing MP can also help speed up each training step by placing and running concurrent operations on separate devices. Unfortunately the amount of parallelism that exists in today's models is often limited [252, 253], either by the algorithm or by its implementation. Therefore, using MP *alone* to obtain performance through parallelization typically does not scale well to a large number of devices. Additionally, maximizing the speedup from MP is often non-trivial [254, 255]. Optimizing MP requires carefully splitting the model to take into account the overhead of communicating activations (during the forward pass) and gradients (during the backward pass) between dependent operations (placed on separate devices) in order to achieve the maximum possible speedup.

This work studies which parallelization strategies to adopt to minimize end-to-end training time for a given DL model on available hardware. We ask the question: how can we improve the scaling obtained from DP, by combining MP and DP to achieve the best possible end-to-end training time at a given accuracy? The novel insight of this work is that when the number of devices (and hence global batch size) grows to a point where scaling from DP slows significantly, MP should then *be used in conjunction* with DP to continue improving training times. The speedup obtainable via MP is critical to this tipping point. We show that every network will have a unique scale at which DP's scaling and statistical efficiency degradation can be overcome by MP's speedup. This work makes

[1] We discuss our methodology in Section 7.4.2 and other possibilities in Section 7.7.

the following contributions:

- We show that when DP's inefficiencies become large, a hybrid parallelization strategy where each parallel worker is model parallelized across multiple devices will further scale multi-device training.
- We develop an analytical framework to systematically find this cross-over point (in terms of number of devices - e.g., GPUs or TPUs - used to train a model) that indicates which parallelization strategy to use when optimizing training of a model, on a particular system.
- We show that hybrid parallelization outperforms DP alone at different scales for different DL networks. We implement 2-way model parallel versions of Inception-V3, GNMT, and BigLSTM, and project that using them, hybrid training provides a speedup of at least 26.5%, 8%, and 22% respectively above DP-only training at scale.
- We propose *DLPlacer*, an integer linear programming based tool to find optimal operation-to-device placement to maximize MP speedup. We demonstrate DLPlacer's effectiveness by using it to derive an optimal placement for the Inception-V3 model [253], showing the obtained 1.32x model parallel speedup with two GPUs is within 6% of that predicted by the tool.

7.2 Background

Neural Network Training: In neural network training, first a batch of inputs is forward propagated through the network to calculate the losses from each input. The losses are back propagated through the network to compute the gradients. The average of the batch's gradients is then used to update the weights. The size of the batch is chosen such that the compute resource of the device used for training is fully utilized. This process is called stochastic batch gradient descent [256, 257]. One forward and backward pass with gradient update to the weights is typically referred to as a training *step*. One iteration through the training data set where all inputs are processed once involves multiple steps and is referred to as an *epoch*. The training process, shown in Figure 7.1 is run for multiple epochs until a desired training accuracy is reached.

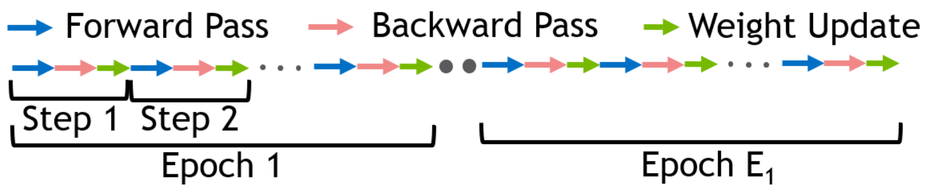
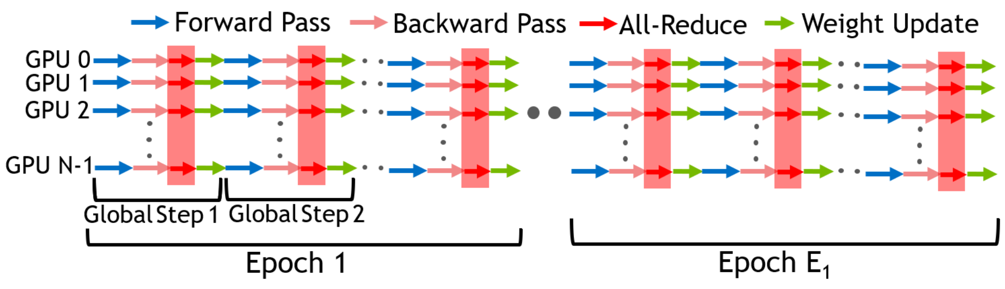


Figure 7.1: Deep learning training flow

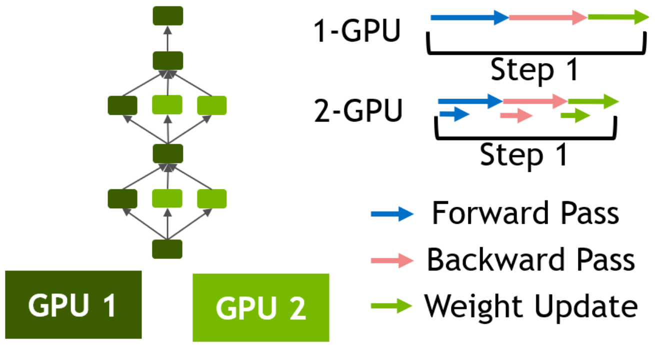
Data Parallel Training: To accelerate training using DP, a full set of model parameters (i.e., weights) are replicated across multiple devices/workers. As Figure 7.2a shows, each worker performs a forward and backward pass independently on a different batch of inputs first. Gradients are then communicated across workers and averaged; after which, each worker applies the same set of gradient values to the model weights. The communication of the gradients across the workers is done using *all-reduce* communication. The method of updating the model after each iteration (using the average of all gradients) is called synchronous stochastic gradient descent *(sync-SGD)* and is the most widely used technique for data parallel training. In this work, we call the batch of inputs per worker a *mini-batch* and the collection of all the mini-batches in a training step a *global batch*.

Model Parallel Training: The model DL is split by placing different operations of it's DFG onto different devices. This approach has been traditionally used for models whose parameters will not fit into a single device's memory [244, 252]. However, MP can provide per step training speedup [245, 255] even when the entire model fits on one device by executing independent operations concurrently on separate devices, as shown in Figure 7.2b. Splitting a DFG among multiple devices is non-trivial for many networks. The communication overhead of moving data between devices may be so large that it may outweigh the gains of MP. Thus, when dividing a network's DFG, characteristics such as compute intensity of each device, inter-device network bandwidth, and even network topology must be considered.

An alternative approach to obtaining speedup is to split up a model across multiple devices using pipelining [258]. This enables splitting a model across multiple devices when a model does not have parallel branches and is sequential in nature. Networks are partitioned into groups containing one or a few layers of the network, where each group is placed on a different device. To orchestrate parallel execution, a mini-batch is split into yet smaller micro-batches and each device processes a



(a) Data parallel training



(b) Model parallel training

Figure 7.2: Different Training Parallelization Strategies

different micro-batch sequentially but concurrently. While subtly distinct, for the purposes of this work we consider pipeline parallelism as an implementation instance of MP.

7.3 Decomposing End-to-End Training Time

End-to-end training time for a DL model depends on three factors: the average time per step (T), the number of steps per epoch (S) and the number of epochs (E) required to converge to a desired accuracy. Therefore, the total training time, i.e., time to converge (C) can be expressed as:

$$C = T \times S \times E \tag{7.1}$$

T is determined by primarily by compute efficiency, i.e., given the same training setup, algorithm, and mini-batch size, T depends solely on the compute capability of a device; better performing hardware provides smaller T values. S on the other hand, depends on the global batch size and number of items in the training dataset. All items in the dataset are processed once per epoch, therefore the number of steps per epoch (S) is equal to the number of items in the data set, divided by the *global batch size*. The number of epochs to converge (E) depends on the global batch size and other training hyper-parameters.

7.3.1 Quantifying Data Parallel Training Time

In data parallel training, the network parameters (weights) are replicated across multiple worker devices and each worker performs a forward and a backward pass individually on a distinct batch of inputs (shown in Figure 7.2a). In this work, we focus on synchronous stochastic gradient decent ($sync-SGD$) for weight updates. In $sync-SGD$ workers are synchronized, i.e., the gradients from workers are shared and network parameters are updated such that all workers have the same parameters after each step. An alternative approach uses asynchronous updates, usually with a parameter server. When scaling to a large number of devices, this approach performs poorly [259]. Therefore, we use a ring-based all-reduce mechanism for data parallel training which provides superior performance and scalability over parameter server based approaches and primarily supports $sync-SGD$. We call the batch of inputs per worker a *mini-batch* and the collection of all the mini-batches in a training step a *global batch*. When using DP alone to accelerate training, the speedup from employing N-way data parallelism (SU_N) compared to training *on a single device* can be expressed as:

$$SU_N = \frac{T_1}{T_N} \times \frac{S_1}{S_N} \times \frac{E_1}{E_N} \tag{7.2}$$

T_1 is the average training time per step when only one device is used for training, while T_N is the time per step when N data parallel devices (with a constant mini-batch size per device) are used. T_N is always larger than T_1 because in DP, after each device has performed a forward and backward pass, the gradients must be exchanged between the devices using all-reduce communication (see Figure 7.2a)[2]. Due to this communication overhead, $\frac{T_1}{T_N}$ will never be larger than one and is typically

[2]Additionally, text and speech networks often exhibit straggler effects where processing some mini-batches take longer than others and therefore, in sync-SGD, devices with a shorter execution time of a mini-batch will suffer from under utilization [260]

less than one. We call this ratio of $\frac{T_1}{T_N}$ the scaling efficiency (SE_N) of N-way DP.

S_1 is the total number of steps required per epoch when one device is used, while S_N is the number of steps per epoch when N devices are used. When a single device is used, the global batch size is equal to the mini-batch size. In N-way data parallelism each device performs an independent step with its own mini-batch of data, therefore the *global batch size* is N-times the mini-batch size per device. Thus, $\frac{S_1}{S_N}$ is also equal to N.

E_1 is the number of epochs required to converge when one device is used, while E_N is the number of epochs required when N devices are used. At larger global batch sizes (higher N), the gradients from a larger number of training samples are averaged which results in model over-fitting as well as a tendency to get attracted to local minima or saddle points. This eventually leads to poor generalization of the network [250, 261–265] and therefore more epochs are typically required to converge. As such, $\frac{E_1}{E_N}$ is usually less than one. Equation 7.2 can thus be simplified as:

$$SU_N = SE_N \times N \times \frac{E_1}{E_N} \tag{7.3}$$

When training at larger device counts (N) both SE_N and $\frac{E_1}{E_N}$ decrease. At large global batch sizes, hyper-parameter tuning (which is a challenging and time consuming task) can be used to try and minimize the increase in number of epochs required for convergence. However for any particular network, beyond a certain global batch size it has been often observed that the number of epochs required to converge increases rapidly, even with hyper-parameter tuning [261]. We describe how we calculate the values for SE_N, E_1, and E_N in detail in Section 7.4.

7.3.2 Quantifying Model Parallel Training Time

As shown in Figure 7.2b, MP enables more than one device to work on the same mini-batch at the same time. This directly reduces the time taken for one training step; term T in Equation 7.1. We call this speedup from M-way MP, SU^M and it can be measured using real hardware by splitting a model across multiple devices and measuring per step execution time or estimated using a numerical model. Note that the SU^M speedup already includes the communication cost of data movement between dependent operations placed across multiple devices.

As previously noted, the global batch size does not increase when employing MP. Therefore, the number of steps per epoch (term S in Equation 7.1) and number of epochs required to converge (term E in Equation 7.1) do not change. As such, improving SU^M reduces convergence time by solely reducing term T in equation 7.1 while the other two terms remain constant. We find that typically the inherent parallelism of a given model or its implementation, limits the achievable SU^M. As a result, *MP alone* is not been considered a broadly applicable scalable parallelization strategy. However, we show that MP *can be combined* with DP to extend training scalibility beyond today's limits.

7.3.3 Hybrid Data and Model Parallel Training:

In Section 7.3.1, we introduced the speedup obtained by $N-way$ DP in Equation 7.3. Now, let's assume we have scaled our training system up to N devices using $N-way$ DP and are happy with the training speedup achieved. If additional devices (say $M \times N$ devices, where M is an integer) were to become available for training, how should we best use these devices for distributed training? Our goal is to identify when to continue to use DP alone, and when to combine DP with MP to obtain the highest possible training speedup. Using DP alone, the speedup from $M \times N$ devices compared to one device is (substituting $M \times N$ for N in Equation 7.3):

$$SU_{M\times N} = SE_{M\times N} \times M \times N \times \frac{E_1}{E_{M\times N}} \tag{7.4}$$

A few observations are important when comparing the speedup from $M \times N$-way DP (Equation 7.4) and speedup from N-way DP (Equation 7.3): First, scaling efficiency is generally lower for the system with $M \times N$-way DP compared to $N-way$ DP [266, 267]. This is because all-reduce communication happens between a larger number of devices. Depending on the values of N, M, and system configuration, all-reduce communication potentially crosses slower inter-node links that leads to increased all-reduce times and reduces $SE_{M\times N}$ [268,269]. Second, since global batch size is larger at $M \times N$ devices (to maintain a constant mini-batch size), the number of steps per epoch is smaller by a factor M compared to N-way DP. Third, the number of epochs required, $E_{M\times N}$, is greater than or equal to E_N. These factors all trend towards lower efficiency as the number of

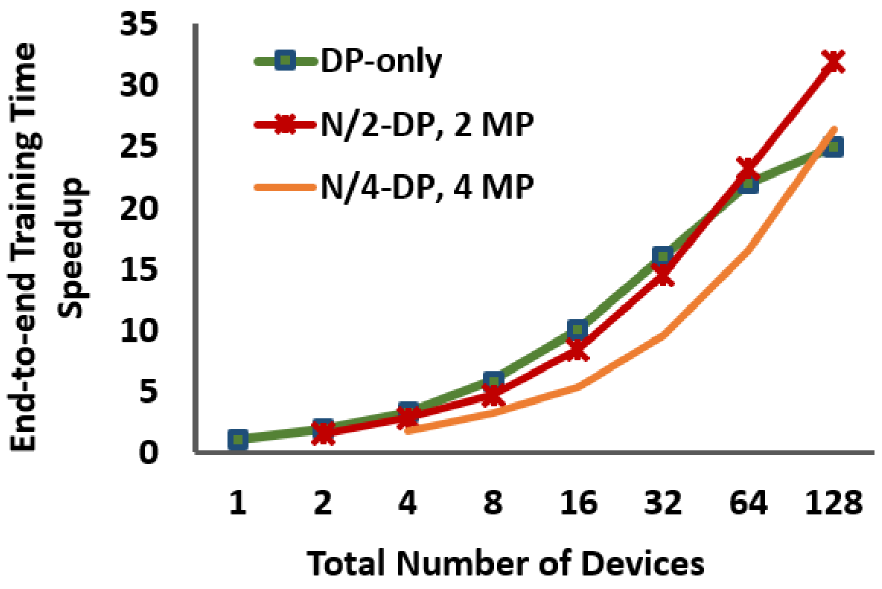


Figure 7.3: An example plot showing the speedup obtained from DP alone, and the hybrid strategy. N refers to the total number of devices used for training.

devices employed in DP training grows.

When using $M \times N$ devices in a hybrid parallelization strategy of N-way DP where each worker uses M-way MP, we consider each worker's per step speedup to be SU^M. Thus the overall training speedup can be expressed as:

$$SU_N^M = SU^M \times SE_N \times N \times \frac{E_1}{E_N} \tag{7.5}$$

When comparing hybrid N-way DP with M-way model parallel workers, versus N-way DP with single GPU workers, the global batch size will remain the same. This is because in the $M \times N$-device configuration, every M devices are grouped into a single data-parallel worker. Thus, the number of steps per epoch remains the same as that of N-way DP at N and $\frac{E_1}{E_N}$ remains unchanged as well. As such, the per-step speedup achieved through MP increases the overall training speedup by a factor of SU^M, when comparing Equations 7.3 and 7.5.

7.3.4 Choosing the Best Parallelization Strategy

By substituting Equations 7.4 and 7.5 into Equation 7.6 we can determine the conditions under which using hybrid parallelization will be better than DP scaling alone. Equation 7.6 shows that if the speedup obtained from MP (for a given model parallelization step) is large enough to overcome the scaling and statistical efficiency loss that comes from increased communication, synchronization overhead, and global batch size increase respectively, employing a hybrid MP and DP strategy will improve network training time.

$$\begin{gathered} SU_N^M > SU_{M\times N} \\ SU^M \times SE_N \times N \times \frac{E_1}{E_N} > SE_{M\times N} \times M \times N \times \frac{E_1}{E_{M\times N}} \\ SU^M > M \times \frac{SE_{M\times N}}{SE_N} \times \frac{E_N}{E_{M\times N}} \end{gathered} \tag{7.6}$$

Figure 7.3 illustrates this concept using a hypothetical scenario. Let's assume implementing MP provides a 45% and 65% improvement with two and four GPUs respectively. The DP-only strategy scales well up to 32 devices after which the improvement in speedup slows down. This enables a hybrid 32-way DP & 2-way MP hybrid parallelization strategy to perform better than 64-way DP given the scaling and statistical efficiency losses at 64 devices, for this example.

Similarly, a hybrid 16-way DP & 4-way MP hybrid strategy outperforms DP-only when scaling from 32 to 128 devices. However in this example, this hybrid strategy's performance is not as good as the hybrid strategy of 32-way DP & 2-way MP. The reason is that 4-way MP's per step speedup (SU^4) does not overcome the trade-off (of using four machines for each data-parallel worker) as efficiently as 2-way MP's per step speedup, SU^2 (when using two machines for each data-parallel worker). Depending on these relative improvements at any device count, the choice of parallelization strategy is critical to the training speedup obtained when scaling to yet larger number of devices. This choice depends on the DL network's properties and system configuration parameters as described above, so there is no one size fits all solution to efficient scale-out multi-device training.

7.4 Methodology

We use the following DL models in our evaluations with their default hyper-parameters, unless otherwise specified:

- *Inception-v3* [253] is used for image recognition and visual feature extraction. The network is composed of multiple blocks, each with several branches of convolution and pooling operations. These branches can be executed in parallel. We use the implementation provided with the public NVIDIA Tensorflow container 18.07 [270] and train the network using the Imagenet dataset [271]. We scale the initial learning rate linearly with the increase in global batch size as originally proposed by Goyal et al. [261]. For measuring epoch counts, we train the model until a training loss of 6.1 is achieved.
- *GNMT* [252] is a language translation network with attention mechanism [252, 272]. We use 4 LSTM layers of size 1024 in the encoder and decoder. We use the public repository at [273] as the basis of our implementation. We use exponential learning rate warm-up for 200 training steps. The learning rate decay is started after 6000 steps and decays for a total of four times after every 500 iterations with a decay factor of 0.5. Such a technique has been shown to scale well when global batch size is scaled. We train the network using the WMT'16 German-English dataset [274] until a BLEU score of 21.8 is achieved.
- *BigLSTM* [275] is a large scale language modelling network. It consists of an input embedding layer of size 1024, 2 LSTM layers with hidden state size of 8192, and a Softmax projection layer of size 1024. We implemented the network in the public NVIDIA PyTorch container v19.06, used a learning rate of 0.1, and trained using the 1 billion word language modelling dataset to a perplexity of 67.

7.4.1 System Configuration and Evaluation Points

For our experiments, we use an NVIDIA DGX-1 [276] with 4 Tesla V100 GPUs [277] connected via NVLink [278] with 16GB of memory capacity. In the BigLSTM experiments we used a similar system but with GV100 cards having 32GB of memory, because this network requires more capacity to execute on a single GPU. We use NCCL2.0 based all-reduce communication for gradient sharing.

In order to project when hybrid training will perform better than DP alone, we need to measure the *epoch counts to convergence* and *scaling efficiency* for DP (defined in Section 7.3.1) for different GPU counts. We also require the speedup achieved via MP when M GPUs are used for a model-parallel worker in a hybrid strategy. Without loss of generalization, we use $M = 2$ for the DL models we use to make a case for future hybrid parallelization strategies. The value chosen for M for an arbitrary DL model will always depend on the speedup obtained from M-way MP and slowdown in scaling efficiency the DP implementation incurs.

7.4.2 Measuring Epoch Counts to Convergence

Typically, epoch counts to convergence for DP on N compute nodes is obtained by running the training on N nodes. We select mini-batch sizes to saturate single GPU throughput or lower if the desired mini-batch size is limited by GPU memory capacity. We perform experiments on a 4-GPU NVIDIA DGX system, so the maximum global batch size possible to measure is $4 \times B$, where the mini-batch size is B. To emulate larger global batch sizes (corresponding to more than four GPUs), we use the delayed gradient update approach [279] where multiple mini-batches are processed per GPU before the gradients are shared for weight update. For example, to emulate a batch size of $16 \times B$ that would be used in a 16 GPU system, each GPU runs the forward and backward propagation of four mini-batches before the GPUs share the gradients (using NCCL 2.0 based all-reduce [269]) and update weights. This methodology allows us to measure the effect of global batch size on the epoch counts required for reaching a desired accuracy, at higher device counts than we have in our physical system. It is worth noting that even though we complete training of a DL model to find E_N, in practice, many DL models are often re-trained many times during development or as new data becomes available. Our proposed systematic modelling approach helps find the best parallelization strategy for optimizing the turnaround time of such subsequent training runs.

Learning rate schedules are sometimes optimized to keep epoch counts to convergence low at large global batch sizes. For example, the learning rate schedules we use for GNMT and Inception V3 were tuned accordingly for this purpose. However, in general, hyper-parameter tuning is time

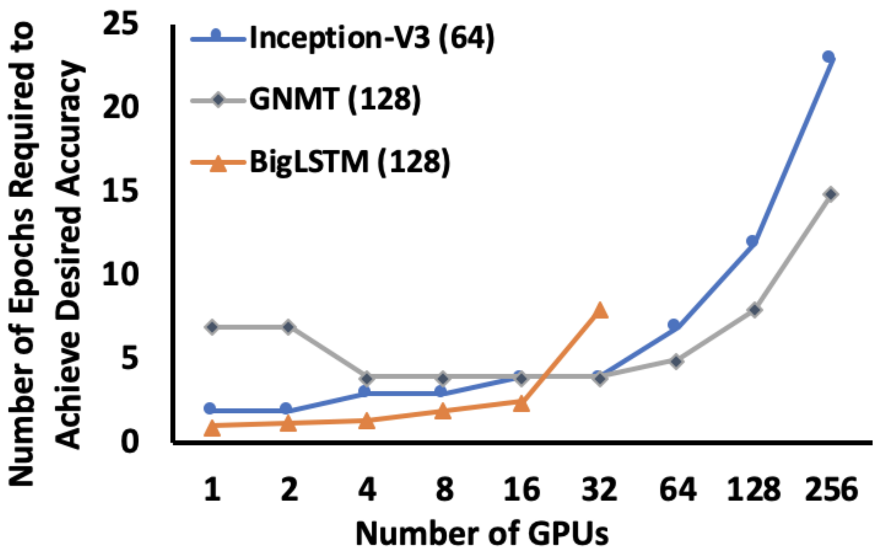
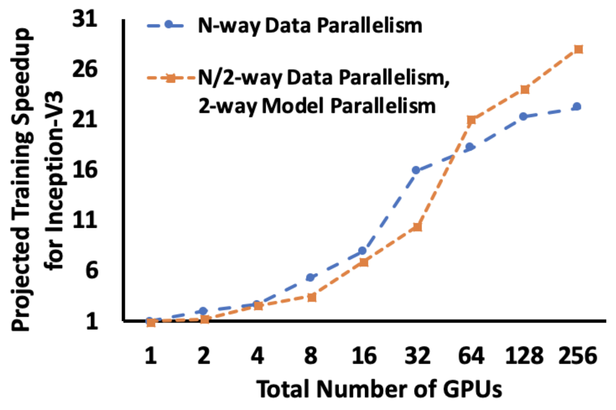


Figure 7.4: Number of epochs required for the networks to converge versus increasing global batch size with increase in the number of GPUs. We emulated larger global batch sizes corresponding to large number of GPUs using the technique described in Section 7.4.2

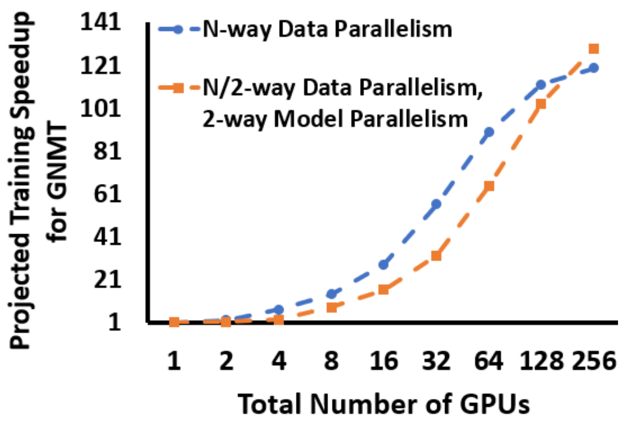
consuming and requires many training runs. Similar to prior work [261], we find that even with such tuning, beyond a certain global batch size, the number of epochs required to converge increases rapidly. As such, the proposals of this work are orthogonal to such efforts.

7.4.3 Estimating Scaling Efficiency

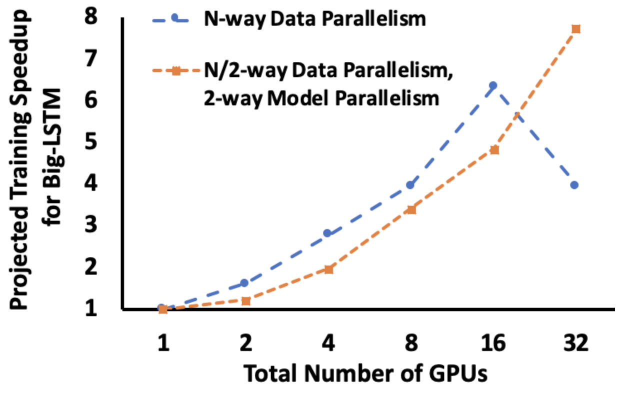
Unlike the methodology we use for emulating larger global batch sizes than what our physical system allows, we can not obtain the scaling efficiency (SE_N) of data parallel training on larger number of GPUs, when using just four. Thus, we conservatively assume a scaling efficiency (SE_N) of 1, i.e., the time overhead of communication and synchronization after each step is negligibly small compared to the time taken for the forward and backward passes. This optimistic assumption *minimizes the impact of hybrid parallelization*, but reflects the reality that framework developers are constantly working to improve overheads that hinder DP scaling efficiency. In fact for CNNs such as ResNet-50, relative scaling efficiency of $> 95\%$ has been achieved for 2048-way DP [280].



(**a**) Inception-V3



(**b**) GNMT



(**c**) BigLSTM

Figure 7.5: Projected speedup of hybrid MP-DP parallelization vs DP-only parallelization

7.4.4 Model Parallel Splitting

Inception-V3's implementation allows a traditional model parallel mapping of independent operations to different GPUs. As such, we split the model's DFG across two GPUs using DLPlacer, later described in Section 7.6. We observed that beyond 2-way splitting of the Inception-V3 DFG, the MP speedup is marginal (see Figure 7.8). For GNMT and BigLSTM, we split their DFGs using pipeline parallelism [258]. Pipeline parallelism is appropriate for implementing MP on these networks due to the use of optimized libraries and fused RNN kernels in their implementations. Pipelining could similarly be useful for models which do not have parallel branches and are sequential in nature (e.g., ResNet, AmoebaNet).

It is worth noting that the original GNMT implementation [252] uses 8-way MP. However, since we use a system with V100 GPUs that have 14x more FLOPs compared to the K80 GPUs used in that prior work, the ratio of communication overhead to computation is larger in our configuration. We use up-to-date CuDNN libraries with fused RNN kernels and observe that splitting the model beyond 2-way provides marginal per-step speedup because of kernel overheads and pipeline imbalance.

7.5 Evaluation

Figure 7.4 shows the number of epochs required to hit the desired accuracy versus the number of GPUs (workers) used in data parallel training. The number of epochs generally increases with an increasing number of GPUs (i.e., with increasing global batch size). For Inception-V3, the number of epochs increases sharply from four to seven as the global batch size increases beyond 2048 (i.e., 32 GPUs) and grows to 23 epochs at a global batch size of 16384 (i.e., 256 GPUs). For GNMT, the epoch count decreases slightly when going from two to four GPUs because the hyper-parameters used are tuned for large global batch sizes. Even with these tuned hyper-parameters, as the GPU count increases beyond 64, the number of epochs required grows rapidly. In BigLSTM, beyond 16 GPUs (i.e., global batch size of 2048), the number of epochs increases rapidly and in fact, 3.2 times the number of epochs is required for 32-way DP compared to 16-way DP. Beyond 32-way DP, the training did not converge within a meaningful time limit. Therefore for each network, as we scale up the number of GPUs used in DP training, $\frac{E_1}{E_N}$ becomes smaller which ultimately hinders the overall speedup achievable through data parallel training alone.

As described in Section 7.3, splitting each network across two GPUs using model parallelism results in per-step speedup when done successfully. Table 7.1 shows the measured MP speedups on our test system for our evaluated networks. Using the number of epochs required and per step speedup from MP, together with the conservative estimates of scaling efficiency, we can then calculate the minimum projected speedup (over DP-only) that can be obtained by implementing a hybrid parallelization strategy across different GPU counts. It is worth noting that the MP speedup achieved on Inception-V3 using expert manual placement of operations was 21%. In Section 7.6 we discuss DLPlacer, a tool we developed for optimizing operation-to-device placement, which improves the MP speedup for Inception-V3 to 32%.

Inception-V3 As shown in Figure 7.5a, beyond 32 GPUs, a hybrid parallelization strategy performs better than DP-only. This is because of the sharp increase in the number of epochs required when the global batch size grows beyond 2048 which saturates the speedup obtainable from DP-only parallelization. When moving from 32 GPUs to 64 GPUs, it is better to use the additional 32 GPUs to do 2-way MP, and our estimates show that the hybrid-strategy will outperform DP alone by at least 15.5%. As the numbers of GPUs grow further, only marginal speedup can be obtained from DP-only parallelization and at 256 GPUs, the hybrid-strategy will be atleast 26.5% better than the DP-only strategy.

GNMT As shown in Figure 7.5b, GNMT scales very well to a large number of GPUs using DP alone. However, even with tuned hyper-parameters for larger batch sizes, DP-only speedup starts to slow down beyond 64 GPUs and dramatically slows down when moving from 128 to 256 GPUs. The hybrid parallelization strategy with 2-way MP and 128-way DP outperforms 256-way a DP strategy by 8%. If the hyper-parameters would not have been tuned for large batch sizes, the gains from hybrid parallelism would be larger and the tipping point would occur at a lower number of GPUs.

Table 7.1: MP splitting strategy and the speedup obtained when split across 2 GPUs

Network	**MP splitting strategy**	**Speedup**
Inception-V3	Partitioned w/ DLPlacer	1.32x
GNMT	Pipeline Parallelism	1.15x
BigLSTM	Pipeline Parallelism	1.22x

BigLSTM As shown in Figure 7.5c, beyond 16 GPUs, BigLSTM does not scale well with an increasing number of GPUs. This is because the statistical efficiency of training decreases rapidly with increasing global batch size, and therefore the significantly larger number of required epochs offsets the throughput increase of multiple GPUs. At 32-GPUs, the large loss in statistical efficiency impacts the overall training speedup of DP-only strategy and the speedup drops significantly. As a result, the hybrid policy provides a 1.22x speedup over the best performing scale of DP-only which happens at 16-GPUs, as Figure 7.5c shows.

In summary, these results show that when statistical efficiency loss reduces the effectiveness of DP-only parallelization, hybrid parallelization (combining DP with MP) will enable higher performance than employing DP alone. Notably, using real scaling efficiency loss values (we conservatively assumed $SE_N = 1$), the improvements from hybrid parallelization would be more pronounced since $\frac{SE_{2N}}{SE_N}$ is often smaller than 0.9 for large LSTM based networks. Based on Equation 7.6, the smaller the ratio, the higher the speedup from hybrid parallelism (SU_N^M) compared to data-parallelism alone ($SU_{M\times N}$).

7.6 Maximizing MP Performance

Maximizing the speedup obtained from MP for a given model improves the scalability of hybrid parallelism. For some networks, optimal placements are easy to achieve by examining a network's DFG. For others, finding the optimal operation-to-device placement that results in the maximum per-step speedup is non-trivial. To this end, we developed an integer-linear programming (ILP) based device placement tool called DLPlacer. DLPlacer maximizes resource utilization by extracting parallelism between operations in a model while also minimizing the communication overhead of

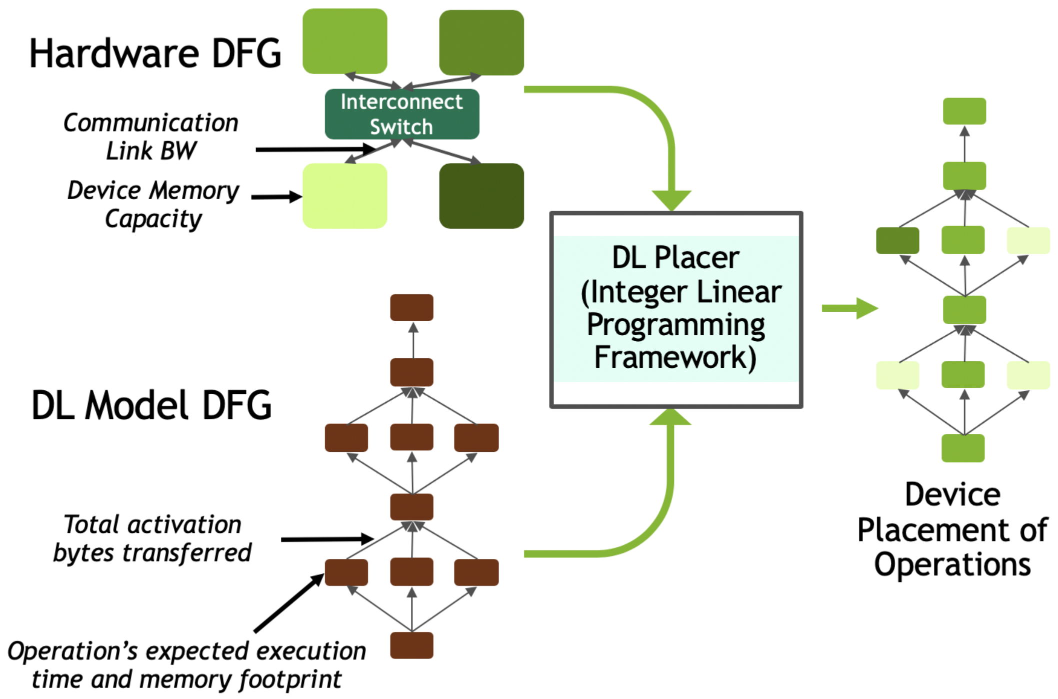

Hardware DFG
Interconnect Switch
Communication Link BW
Device Memory Capacity
DL Model DFG
Total activation bytes transferred
Operation's expected execution time and memory footprint
DL Placer (Integer Linear Programming Framework)
Device Placement of Operations

Figure 7.6: DLPLacer Flow Diagram.

moving data between the compute nodes.

Figure 7.6 shows DLPlacer's tool flow. We express a DL model as a compute DFG, with a set of vertices K corresponding to compute operations and a set of uni-directional edges E showing operation dependencies. For example, for some $k1, k2 \in K$ and $e_{k1,k2} \in E$, $e_{k1,k2} = 1$ means $k2$ is dependent on $k1$. The expected execution time of a vertex is represented as $\Delta(k)$ and the memory footprint of the vertex for a given batch size is represented as $M(k)$. Edge weight ($D(e)$) corresponds to the number of bytes exchanged between the operations it connects. The node and edge weights can be obtained by profiling a model on a compute device (e.g., GPU) or can be analytically calculated, with the former approach being more robust and the latter more flexible.

Using similar notation, we express a system as a hardware graph [281,282] where a set of compute (e.g., GPUs) nodes N and router nodes (network switches) R are connected through a set of physical links L. As an example, for $n1$, $n2 \in N$ and $l_{n1,n2} \in L$, $l_{n1,n2} = 1$ means nodes $n1$ and $n2$ are connected. We assume physical links are bidirectional, so $l_{n1,n2} = l_{n2,n1}$. The bandwidth of the physical link is denoted by $B(l)$.

DLPlacer's ILP solver minimizes per step training time by providing an assignment of compute

DFG operations on to the hardware graph, a schedule, and a communication routing of activations, weights, and gradients. This is done by mapping model DFG's vertices to compute nodes, dependency edges to physical link mapping (if dependent vertices are placed in separate devices) and determining the execution start time of each vertex on a device. This mapping must satisfy a series of constraints and variables which are described next. A summary of all the variables is provided in Table 7.2

Table 7.2: Summary of notations used in the ILP

Notation	**Meaning**
Inputs : Computation DFG	
K	Set of compute operation/kernel vertices
E	Set of edges between vertices
$\Delta(K)$	Expected execution time of compute vertex
$M(K)$	Memory footprint of the compute vertex
$D(E)$	Number of data bytes transferred in an edge
Inputs : Hardware Graph	
N	Set of compute hardware nodes
R	Set of router nodes
L	Set of physical links connecting routers and hardware nodes
$B(l)$	Bandwidth of the physical links
$Mem(N)$	Device memory capacity
Variables: Outputs	
$P_{kn}(K,N)$	Mapping of compute vertex to hardware node
$T_k(K)$	Time a vertex is launched on the hardware
$C_{el}(E,L)$	Mapping of dependency edges to physical links
Variables: Intermediate	
$\Delta_e(L)$	Delay of communication of edge e

Now, we describe the constraints in details.

Placement of compute operation vertex: If the binary variable $P_{kn}(k,n)$ = 1, then vertex k is mapped to node n. Each operation of the compute DFG should be mapped to only one node on the hardware graph, therefore this gives us the following constraint:

$$\forall k \ \sum_{n} P_{kn}(k,n) = 1 \tag{7.7}$$

Routing of Activation Data: The output from a vertex need to be routed to the dependent vertices through the physical communication links. Each edge e needs to be mapped to a sequence of one or more links l. The path for communication must start from the origin vertex and end at the destination vertex if the origin and destination vertices are different. Therefore, for the source and destination nodes, exactly one link should be allotted for the edge, and for all other nodes (including router nodes). This constraint can be formulated as follows:

$$\begin{gathered} \forall e,n,k_i,k_j \ |e_{k_i,k_j} = 1 \ \ if P_{kn}(k_i,n)! = P_{kn}(k_j,n) \\ \sum_{l|l_{n,nx}=1} C_{el}(e,l) = 1 \ \ \forall nx \in N \end{gathered} \tag{7.8}$$

To find a contiguous path, we enforce for all non-source and non-destination nodes (includes router/switch nodes) that either two links should be allotted, one for the incoming traffic and one for the outgoing traffic or no links should be allotted.

$$\begin{gathered} \forall e,k_i,k_j \mid e_{k_i,k_j} = 1, n \mid P_{kn}(k_i,n) = P_{kn}(k_j,n) = 0 \\ \sum_{l|l_{n,nx}=1} C_{el}(e,l) = \sum_{l|l_{n,ny}=1} C_{el}(e,l) \forall nx,ny \in N \cup R \end{gathered} \tag{7.9}$$

Scheduling of Vertices: Vertices need to be scheduled such that their dependencies are met. We calculate the time at which a vertex can begin executing by considering the start time of the other vertices it is dependent upon and the execution time and communication delay of the input activations.

$$\begin{aligned}&\forall k_{src}, k_{dest}, e \mid e_{k_{src},k_{dest}} = 1,\\ &T(k_{dest}) \geq T(k_{src}) + \Delta(k_{src}) + \Delta_e(L)\end{aligned} \tag{7.10}$$

This equation ensures that a vertex k_{dest} can begin only after all the vertices it is dependent upon (k_{src}) has finished executing and the input activations have been communicated to the device where k_{dest} is placed.

$\Delta_e(L)$ is the time to communication the edge data. The amount of data that need to be routed between two vertices is the amount of total output activation (dependent on mini-batch size). We assumed that the time for communication would depend on the number of links it need to traverse and the bandwidth and latency of these links. Therefore, $\Delta_e(L)$ can be computed as follows:

$$\forall e \in E, \Delta_e(L) = \sum_{l \in L} C_{el}(e,l) * (D(e)/B(l) + L(l)) \tag{7.11}$$

Another timing related constraint comes from the fact that multiple operations can be mapped to a device but co-located vertices cannot be scheduled on the same device at the same time. The start of the execution of consecutive operations on a device should atleast be separated by the execution time of the operation which starts earlier among the two. Note that this constraint is unnecessary for operations which lie on the dependency path of each other because of the previous constraint. Therefore, this constraint can be formulated as follows:

$$\begin{aligned}&\forall k_x, k_y, n \mid P_{kn}(k_x,n) = P_{kn}(k_y,n) = 1 \ and \ e_{k_x,k_y}! = 1,\\ &\quad if\ T(k_x) > T(k_y):\\ &\quad\quad T(k_x) \geq T(k_y) + \Delta(k_y)\\ &\quad else:\\ &\quad\quad T(k_y) \geq T(k_x) + \Delta(k_x)\end{aligned} \tag{7.12}$$

Device memory capacity constraint: This constraint ensures that the summation of the memory footprint of all the vertices placed on a device does not exceed the device memory capacity.

$$\forall n \in N, \;\; Mem(n) \geq \sum_{k \in K} P_{kn}(k,n) * M(k) \tag{7.13}$$

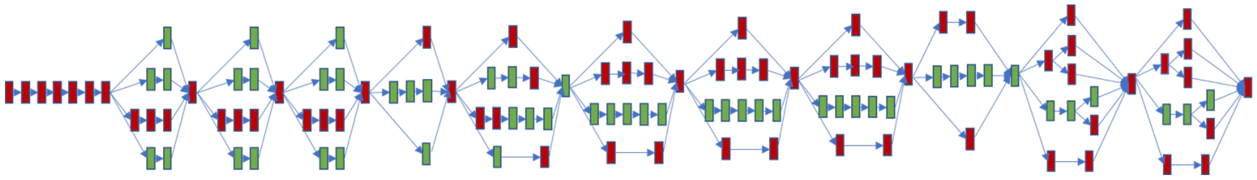


Figure 7.7: DLPlacer's placement solution for Inception-V3. Different colors denote different devices.

In this DLPlacer framework, we assumed the following:

1. Two operations which are co-located on a device are executed back-to-back, without any delay in between the end of one operation and the beginning of the other.
2. Communication of tensors between devices are overlapped with computation.

Based on these constraints and assumptions DLPlacer predicts the training speedup for a given MP solution. In our work, we considered operations at the granularity of tensorflow operations (e.g., conv2D, conv3D), however DLPlacer can be used to even find placements when the operations are partitioned into finer granularity operations (e.g., partitioned by channels, filters etc.). But, such fine grained operation splitting requires framework support for correct back-propagation and therefore was not a focus of this work. Note that because of framework-induced overheads and unmodeled operating system effects, correct prediction of the exact speedup is difficult. Modelling these overheads is challenging and often depends on the mapping of kernels to high-level operations (e.g., mapping of CuDNN [283] kernels to convolution/FC/etc.), device architecture, and the runtime implementation. Despite the challenges in accurate prediction, we believe ILP based MP optimization is worthwhile to pursue based on the observed improvements over manual optimization.

Inception-V3 Case Study

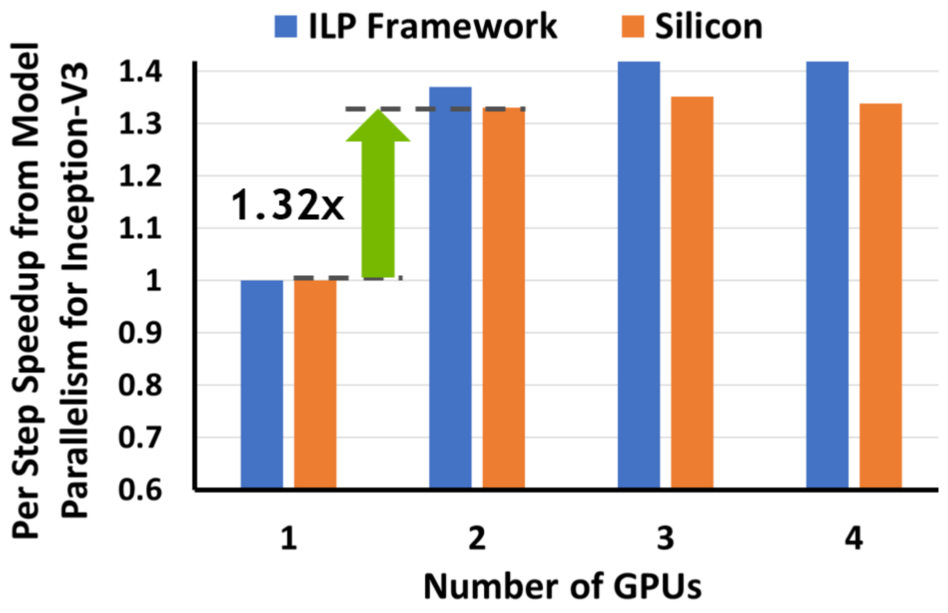


Figure 7.8: Normalized per-step speedup from model parallelism as estimated by DLPlacer and obtained from silicon experiments for the Inception-V3 network.

As inputs to DLPlacer, we analytically calculate the execution and communication times of the operations in the Inception-V3 DFG. For example, given the input/output tensor sizes of a convolution operation, we calculate the number of floating point operations (FLOPs) required, and based on advertised compute capability of NVIDIA's V100, we calculate the operations' expected execution time. Similarly, communication time between nodes is calculated based on the tensor sizes of the nodes in the model DFG along with NVLink bandwidth and latency. The placement solution of Inception-V3 using 2-GPUs is shown in Figure 7.7. We implement the placement directives from DLPlacer using Tensorflow's $tf.device()$ command, and we have validated DLPlacer's speedup estimation against real hardware performance.

In Figure 7.8, the blue bars show the normalized per-step speedup estimated by DLPlacer for the optimal placement solution it finds. DLPlacer's runtime on an 18-core Xeon-E5 system to find Inception-V3's placement solution is ∼11-18 minutes depending on the number of device nodes in the hardware graph. The orange bar for each configuration shows speedup as measured on real silicon with DLPlacer's placement applied to the Tensorflow implementation. The speedup-ups measured by DLPlacer are within 6% of the actual speedup obtained from the silicon runs. It is interesting to note that the 1.32x speedup obtained with the real silicon 2-GPU placement is almost the same as what is optimally obtainable with three or four GPUs. This is due to the limited parallelism available in the network, which DLPlacer almost completely exploits with a 2-GPU placement. Identifying a 2-GPU placement that gives this performance by simple observation of the network and without using a tool like DLPlacer is non-trivial. DLPlacer essentially finds placement with the shortest possible critical path among many feasible placement solutions and places the operations on the critical path in one GPU so as to avoid communication overhead. This shows the importance of such a tool for maximizing performance obtainable from MP while using minimum number of GPUs.

7.7 Related Work

This work identifies scaling and statistical efficiency losses as the largest challenges to scalable data parallel training, but researchers are improving the scalability of both data and model parallel training rapidly. We summarize the most significant related advancements here.

7.7.1 Hybrid Parallelization

Previous work [284–286] has also used hybrid parallelization for scaling DL training. To the best of our knowledge, none of these proposals provides a systematic method to identify which strategy is best for scaling-out network training at different device counts. Das et al. [284] perform hybrid training on CPUs and maintain the global batch size by shrinking the mini-batch size per CPU, but do not incur a statistical efficiency loss because a small mini-batch size is large enough to saturate CPU throughput. Maintaining a constant global batch size while shrinking the mini-batch size (per compute device) can also be done for GPUs, however GPUs typically require larger mini-batch sizes to maintain high utilization. Yadan et al. [285] show that a hybrid (2-way DP, 2-MP) approach performs better than both MP-only and DP-only when training AlexNet on a 4-GPU system, but do not discuss the cause of the results or evaluate this effect across different GPU counts. Dean et al. [245] used hybrid parallelism to train models which would not fit in a single GPU's memory. Therefore, in each data parallel worker, the model replica is model parallelized across multiple devices. However with increase in capacity of memory capacity in today's GPUs, large models such as Inception-V3, GNMT etc. can fit in to a single GPU memory while using sufficiently large mini-batch size to saturate the compute throughput. Moreover, using model parallelism for models that do not fit in a single GPU's memory is largely orthogonal to the issue we address in this work. Amir et al. [287] have shown that hybrid parallelization strategy can result in lower communication overhead over both MP and DP. None of these works, however, have provided a systematic analysis of finding what parallelization strategy would minimize the end-to-end training time when a set of N compute devices are available for training. Moreover, implementing hybrid parallelism is often tricky because finding the optimal strategy to split a model is non-trivial and is dependent on the model DFG and system hardware.

7.7.2 Orthogonal Parallelization Strategies

Exploiting model parallelism is just one way to achieve per step speedup without increasing global batch size. Other strategies exist that can be combined with, or used in place of, model parallelism to augment data parallel scaling under our proposed model. Jia et al. [288] propose layer-wise parallelism for CNNs where each network layer can use an individual parallelization strategy. A combination of the 4D tensor dimensions can be used to parallelize a given layer and exploring multiple dimensions may provide larger runtime benefits than MP. However, such a technique is not yet supported by most frameworks and is evaluated using a custom framework (Legion [289]). Similar to GPipe [258] (discussed in Section 7.2), Harlap et al. [290] propose partitioning a DL model's DFG into multi-layer stages and applying pipeline parallelism. To enable maximum device utilization, PipeDream uses asynchronous weight updates which can lead to poor statistical efficiency as the number of devices increases. It is likely that one or a combination of the layer-wise, pipeline, and model parallelism techniques can be combined with DP training to maximize end to end training performance and efficiency.

7.7.3 Alternate Techniques to Improve DP Scaling

Data parallel training employing sync-SGD suffers from poor scaling efficiency due to synchronization overheads. Prior work [291–294] has attempted to address this by using asynchronous SGD. However, asynchronous SGD can still result in poor statistical efficiency while making performance debugging difficult. Hyper-parameter tuning is a broad approach to improving statistical accuracy and training convergence. Techniques such as tuning and scaling learning rates [250, 261–263, 295, 296], or auto-tuning the momentum [297] are several important examples. However, these techniques are very problem specific, require extensive knowledge of the DL models, and are very time consuming for developers. Furthermore, hyper-parameter tuning is not always effective [298].

Other works [299, 300] propose using a different learning algorithm, called model averaging, for training with small batches. An average model can asymptotically converge faster, but finding the asymptotic region is difficult [301]. Koliousis et al. [300] use multiple learners (each using a small batch size) run on many GPUs, and an average main model is used to synchronously track the learning. Model averaging is not yet mainstream or supported by popular DL frameworks and thus requires custom re-implementation of the DL models.

7.7.4 Reinforcement Learning-based Device Placement

Prior work has shown that by using reinforcement learning-based (RL-based) placement of operations onto devices, MP can achieve training speedup and that the RL generated placement is non-trivial [255]. However, RL-based approaches can be long-running and compute-intensive with no notion of optimality. On the other hand, DLPlacer can provide optimal device placement solutions, though can still be compute intensive for complex DFGs and when system graph contains a large number of devices. However, it should be noted that for simpler DFGs, simple heuristics could achieve near-optimal placement results.

7.7.5 Framework Support

As we discuss in Section 7.4.4, we implement MP differently for our BigSLTM and GNMT evaluation compared to Inception-V3. This is mostly driven by the baseline implementations of BigLSTM and GNMT, which makes it very non-trivial to exploit intra-layer MP in these networks. We use pipeline parallelism for exploiting inter-layer MP for these two models. While a given network's implementation can be one hurdle to exploiting MP, the framework it is implemented with can also add to the complexity. TensorFlow and Pytorch have different levels of support for assigning operations or tensors to different devices, but neither provide any automatic intra-layer parallelism extraction support. DSSTNE [302], is Amazon's deep scalable sparse tensor network framework which has more complete support for extracting intra-layer parallelism. However, it only supports fully connected layers and therefore is not a versatile framework for implementing different types of DL networks such as CNN and RNN based networks. Also, this framework is not broadly used and as such was not a focus of our evaluation in this work.

7.8 Conclusion

This work demonstrates the benefits of combining model-parallelism (MP) with data-parallelism (DP) to overcome the inherent scaling and statistical efficiency losses that data-parallel training has at scale. We analyze the end-to-end training time of DP to understand how scaling and statistical efficiency loss impacts training scalability, and show that the MP speedup achieved for a given DL model is critical to the overall scalability of a hybrid parallelization strategy. We demonstrate that when the global batch size in DP grows to a point where DP-only training speedup drops off significantly, MP can be used in conjunction with DP to continue improving training times beyond what DP can achieve alone. We evaluate the performance benefits of such a hybrid strategy and project that for Inception-V3, GNMT, and BigLSTM, the hybrid strategy provides an end-to-end training speedup of at least 26.5%, 8%, and 22% respectively compared to what DP alone can achieve at scale.

CHAPTER 8

DeepFlow: A Cross-Stack Pathfinding Framework for Distributed AI Systems

Over the past decade, machine learning model complexity has grown at an extraordinary rate, as has the scale of the systems training such large models. However there is an alarmingly low hardware utilization (5-20%) in large scale AI systems. The low system utilization is a cumulative effect of minor losses across different layers of the stack, exacerbated by the disconnect between engineers designing different layers spanning across different industries. We propose CrossFlow, a novel framework that enables cross-layer analysis all the way from the technology layer to the algorithmic layer. We also propose DeepFlow (built on top of CrossFlow using machine learning techniques) to automate the design space exploration and co-optimization across different layers of the stack. We have validated CrossFlow accuracy with distributed training on real commercial hardware and show case several DeepFlow case studies demonstrating pitfalls of not optimizing across the technology-hardware-software stack for what is likely, the most important workload driving large development investments in all aspects of computing stack.

8.1 Introduction

Over the last decade, the demand on compute and memory resources for AI workloads has grown by multiple orders of magnitude [303]. As AI models grow in size along with the volume of training data, distributed training on cutting-edge scale-out systems composed of a large number of accelerators and processors has become the norm. However, it has often been noticed that large scale AI training suffers from poor resource utilization. E.g., recent analysis reveals 5-20% utilization across 1000s of GPUs [304] when training large-scale language-models. Such poor utilization of resources is becoming a source of major concern. Inefficiencies across different layers of the compute stack [305, 306] (from hardware micro-architecture to software parallelization strategies) and the design imbalance across different layers are among a few factors resulting in such low system utilization. Different layers of the stack, technology nodes, hardware architecture, network topology, model architecture, parallelism strategy are designed across different organizations and retrofitted into the large-scale systems. The distributed nature of the design makes cross-layer optimization challenging if not impossible. For example, high-level design decisions like batch size, model architecture, and parallelism strategy are exploited at algorithmic level stress underlying hardware components (network, memory bandwidth or compute resources) in different ways which call for different architectural designs, network topologies and technologies to ensure high system utilization.

Despite this, the distributed AI training hardware landscape often focuses on just a small set of parallelism strategies for a fixed hardware design [305]. Exploring the trade-offs between parallelization strategy (e.g. data parallelism and model parallelism) and performance (run-time) is often done in an ad-hoc manner. There is no methodical framework or research that explores the trade-offs between low-level hardware technology details and high-level algorithmic design (such as model architecture, parallelism strategy and batch size) on over performance and utilization of compute and memory resources. As a result, we set out to develop a framework that could enable across-the-stack analysis and allow us to look at the optimal points in the vast technology, system and algorithm design space. Towards that goal, we develop **CrossFlow**, a performance modeling framework that enables "what-if" analysis across different layers of the stack, and **DeepFlow** that builds on *CrossFlow* and uses machine-learning based techniques to automate the design space search. [1] CrossFlow is an end-to-end performance modeling tool based on an analytical model which takes the entire system-architecture into account and is more sophisticated than a simple Roofline analysis and less time-consuming than simulation. The framework provides a templatized interface for defining technology (minimum operating voltage, bitcell area, etc), chip (compute cores, memory hierarchy, etc.) and system-level architecture (node-level organization, intra-node network, and inter-node network), machine-learning model's compute graph, and parallelization

[1] We will open-source and publicly release the tool.

strategies and predicts run-time per iteration step. Key contributions of this work include:

- We develop the *first* full-stack pathfinding framework, DeepFlow, for large deep learning (DL) training: the driving workload for most future technology, hardware and software development.
- We validate CrossFlow performance prediction against measurements on real commercial hardware (NVIDIA P4, V100 and DGX-1) running kernels and DL application in both single and distributed settings, observing near perfect correlation and 10% - 16% error.
- As examples, we conduct a variety of case studies looking at impact of a variety of high-cost technology innovations on eventual performance of distributed DL training and show that many of them may not deliver on the promise.

CrossFlow and DeepFlow can be used to bridge researchers across different layers of the stack (often spanning across different industries) to communicate their needs.

8.2 Motivation

The optimal system that should be used to train a particular machine learning model varies based on the high-level algorithmic decisions that are made. Different batch sizes, degrees of parallelism, and parallelism strategies stress the underlying hardware resources in different ways. Hence, different hardware designs are required in order to maximize resource utilization. One important metric that guides the optimal design point at hardware-level is computation intensity. Computation intensity is defined as the ratio of the number of computation flops to number of accesses to main memory. This ratio dictates the optimal ratio of computation throughput to memory bandwidth in the underlying hardware accelerator.

Figure 8.1 shows the computation intensity distribution of different degrees and types of parallelism for different number of V100 GPUs. We performed this analysis for a GEMM problem of size $(64K, 64K, 64K)$ elements split across these GPUs. Each boxplot shows the spread of computation intensity for different levels of parallelism. The spread within each parallelism degree is the result of

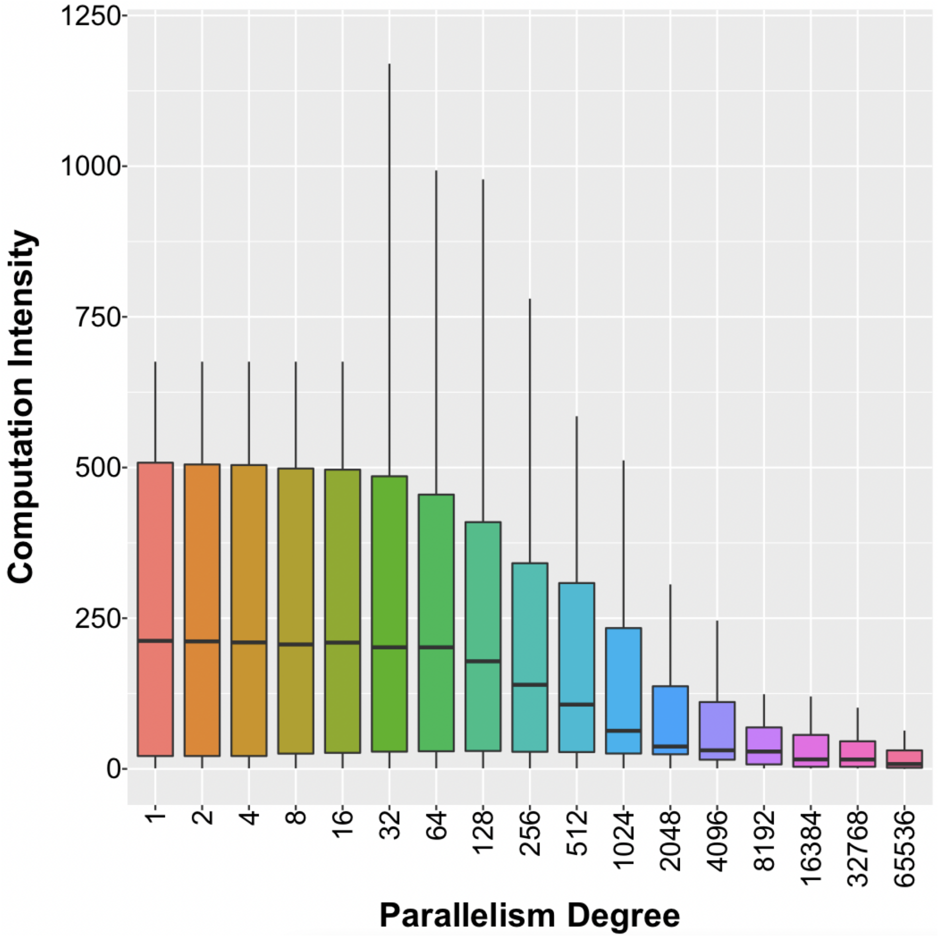


Figure 8.1: Impact of Parallelism Degree on Computation Intensity.

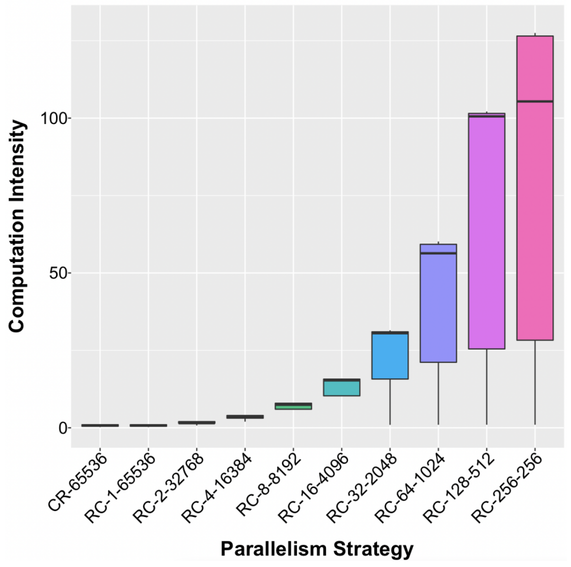


Figure 8.2: Impact of Parallelism Strategy on Computation Intensity.

different parallelization strategies for a given parallelism degree and also different tiling strategies for any given parallelization strategy. It is clear from this figure that computation intensity is much smaller at higher degrees of parallelism, implying the need for a different design point.

Besides the parallelism degree, the choice of parallelization strategy has a direct impact on the utilization of the underlying hardware. Figure 8.2 shows the distribution of computation intensity across different parallelization strategies for the same problem specifically parallelized across 65536 V100 GPU cards. Each boxplot shows a different parallelization strategy. RC or CR refers to Row-Column or Column-Row distributed GEMM (a.k.a kernel parallelism, more details in Section 8.3.3). It is clear from the figure that computation intensity is different across different parallelization strategies, implying the optimal design point would be different for different parallelization strategies.

Therefore, the optimal hardware architecture strongly depends on high-level algorithmic and software decisions. Hence, an exploration framework is needed that would allow one to obtain and analyze the various possible optimal system and algorithm design combinations. Furthermore, large training workloads are rapidly becoming the applications driving massive investments in semiconductor technology development all the way down to fabrication equipment, making such a cross-layer pathfinding framework immensely valuable to ML engineers, system architects and technology developers alike.

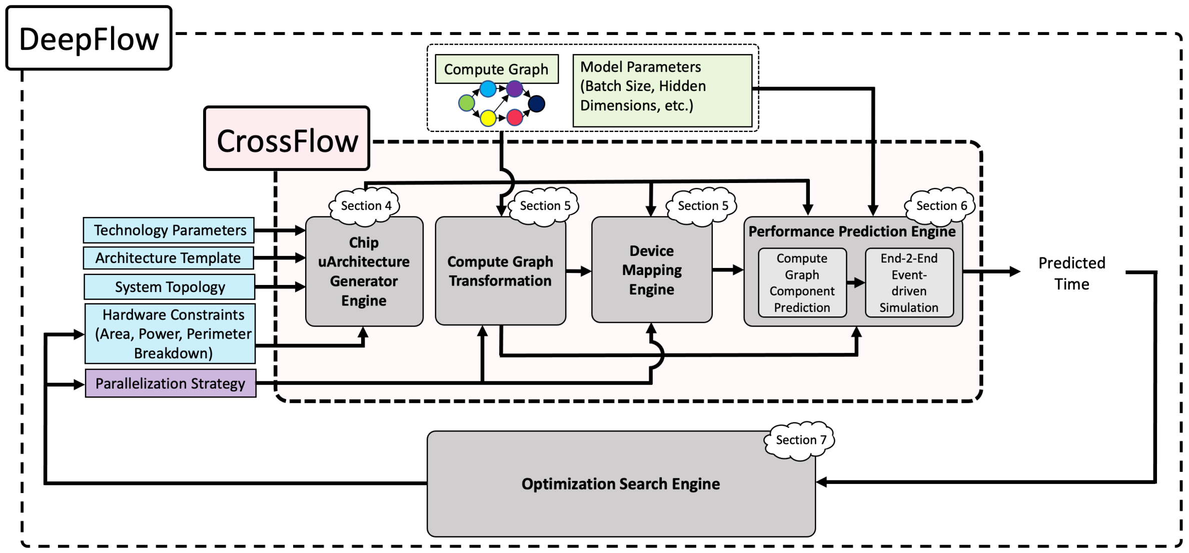

DeepFlow
Compute Graph
Model Parameters (Batch Size, Hidden Dimensions, etc.)
CrossFlow
Section 4
Section 5
Section 5
Section 6
Technology Parameters
Architecture Template
System Topology
Hardware Constraints (Area, Power, Perimeter Breakdown)
Parallelization Strategy
Chip uArchitecture Generator Engine
Compute Graph Transformation
Device Mapping Engine
Performance Prediction Engine
Compute Graph Component Prediction
End-2-End Event-driven Simulation
Predicted Time
Section 7
Optimization Search Engine

Figure 8.3: DeepFlow Overview.

8.3 DeepFlow Overview

Figure 8.3 shows an overview of the DeepFlow framework. DeepFlow takes the following set of **inputs**: (1) System design hierarchy (e.g., the number of accelerator nodes per device, the number of devices in the system, the network topology connecting nodes within a device and across the devices), (2) Architecture template of each accelerator node which provides a high-level definition of its components and how those components fit together. The purpose of the template is to provide a blueprint for the accelerator without committing to any specific hardware parameters. (3) Technology parameters for each hardware component (e.g. energy per flop), (4) Design budgets for each hardware component (area, power, perimeter), (5) Machine learning model specification in the form of a high-level compute graph, parameters of each compute node (kernel type, tensor dimensions), and (6) Parallelism strategy (data, model, kernel, and/or pipeline parallelism dimensions) which distributes the compute graph across the entire system. (7) Device mapping strategy which defines mapping of parallel shards onto hardware nodes. Given these inputs, DeepFlow predicts the end-to-end performance of one iteration (i.e., single batch) of the model and finds an optimal hardware-software-technology design point as **output**.

DeepFlow is composed of two major components. CrossFlow which operates in a stand-alone mode and can predict performance for any input configuration; and a search and optimization engine (SOE) which enables design space search.

8.3.1 CrossFlow Building Blocks

u-Architecture Generator Engine (AGE) AGE takes the following set of **inputs**: (1) Design constraints (i.e the power, area and perimeter budget and breakdown across micro-architectural components such as cache, network, compute cores). This breakdown can be provided manually by users or automatically by the Search and Optimization Engine (SOE, Section 8.3.2). (2) Technology parameters such as energy per flop, energy per data bit transfer for each level of memory and network hierarchy, threshold and maximum gate voltage, integration substrate parameters such as bump/interconnect pitch. We provide a wide range of standard and future technology libraries as baseline. (3) Architecture template which is a blueprint of the underlying accelerator chip without committing to any specific hardware parameters.

Given these input, AGE performs a frequency-voltage-area scaling optimization to generate the following **output** parameters such that design budgets for all component are met: (1) Compute throughput. (2) Capacity for different levels of memory hierarchy. (3) Bandwidth to each level of memory hierarchy. (4) Inter-node as well as intra-node network bandwidth. These parameters are then utilized by the performance prediction engine (PPE) to estimate the execution time of each kernel.

Compute Graph Transformation and Device Placement Engine (DPE) The parallelization strategy and device mapping are critical in deciding the overall execution time. Here, we first transform the model graph to a 'super-graph' to reflect the parallelization strategy provided by users or SOE engine. For example, to apply data parallelism, the model graph is replicated and appropriate edges are added to model the gradient exchange. After generating the transformed graph, DPE assigns the vertices of the transformed graph to the system nodes following a heuristic approach to minimize the communication overhead.

Performance Prediction Engine (PPE) With the device mapping for all the vertices of the compute (super-)graph known, the next step is to calculate the overall execution time for a forward pass and/or a backward pass. Using hierarchical roofline model, the performance prediction engine (PPE) evaluates the execution time of each vertex of the compute (super-)graph. Next, we use an event-driven simulator to calculate the overall end-to-end execution time such that resource constraints (e.g. one kernel at a time per GPU) and scheduling constraints (e.g. prioritize launching kernels in the next layer before kernels on the same layer) are met.

8.3.2 Search and Optimization Engine (SOE)

Co-optimizing micro-architectural parameters and the parallelization strategy that minimizes the overall end-to-end execution time requires navigating a large space of design parameters. Search and optimization engine (SOE) enables the automatic design space search and finds an optimal design point which meets the design constraints and minimizes the overall execution time. SOE takes inspiration from ML-assisted search algorithms, in particular gradient decent search with momentum and builds on top of the CrossFlow modeling engine.

8.3.3 Parallelism Strategy Space

There are a myriad of ways to parallelize a model across a large multi-node system. Exploring the parallelism space and finding the optimal strategy is critical to overall performance and system utilization. DeepFlow explores kernel, data and layer parallelism strategies. It uniquely identifies each parallelism strategy by the following notations: `R{KP1}_C{KP2}_d{DP}_p{LP}` or `CR{KP1}_d{DP}_p{LP}` depending on the choice of kernel parallelism. `KP1` and `KP2` are the parameters of kernel parallelism. For Row-Column (`RC`), `KP1` and `KP2` would refer to the number of ways we shard the first matrix across rows and the second matrix across columns. For Column-Row (`CR`), we would only need one parameter to specify the strategy; `KP1` will refer to the number of ways we cut the first matrix across columns and the second matrix across rows. One can think of this parallelism strategy as outer-product. `DP` represents the number of model replicas and data shards assigned to each to exploit data parallelism. `LP` is the number of ways we cut layers into stages to exploit pipeline parallelism.

8.4 uArchitecture Generator Engine

The uArchitecture generator engine, AGE, takes three sets of inputs: (1) A technology components library, where the characteristics of each component such as cores, different types of memories, network interfaces, etc. are defined, (3) Architecture template, where the overall high-level chip and system organization (such as compute and memory hierarchies) is provided, (2) Hardware resource allocation, where area, power, and chip perimeter budgets are provided for the different components of the system. Using this information, the AGE generates the final uArchitecture parameters (such as overall compute throughput, memory bandwidths at different memory levels, network bandwidth) as shown in Figure 8.3.

8.4.1 Technology Components Library

A system is generally composed of many primitive components or building blocks such as the compute units, SRAM banks, DRAM, interconnect network components (on-chip and off-chip), etc. A library of these components and their associated technology parameters are provided as input to the tool through a *tech_config* YAML file. We classify these components in to three primary categories: compute, memory and network.

Compute Attributes for the minimal compute components such as general purpose compute units, accelerator units (e.g., matrix-multiplier units, vector-matrix multiply units, systolic array) etc. are specified under this category. When a compute component is added to the library, the compute attributes listed in Table 8.1 will have to be defined for that component. The tool user can add any type of compute component in the library ranging from a simple scalar unit to a complex unit comprising of a bundle of tensor cores and capture the micro-architectural characteristics in the final architecture template file.

Memory The memory components in a system can be built out of different technologies (e.g., SRAM, DRAM, MRAM, RRAM, 3D-XPoint). Also, these memory components can be used in two ways: on-chip memory and off-chip memory. A library of fine-grained memory components can be created and stored under this category which is utilized to construct different levels of the memory hierarchy. The characteristics of the on-chip components are described at the granularity of a bank because the smallest on-chip memory unit available to a system designer is usually a memory bank. The parameters of a memory bank such as capacity, bit area, periphery overhead etc. are taken as inputs. On the other hand, we model the off-chip memory components such as DRAM, 3D-XPoint, etc., at device level granularity, e.g., an HBM stack. This is because the off-chip components are usually obtained at a device level granularity. For off-chip memories, other parameters such as memory controller area, I/O bus width per device, etc. need to be defined. This information is then used to precisely model the capacity and throughput of different levels of the memory hierarchy under the given area and power constraints.

Compute	Technology Node	Nominal Area
	Nominal Voltage	Threshold Voltage
	Nominal Frequency	Min & Max Voltage
	Nominal OP rate	Min & Max Voltage
On-chip Memory	Technology	Latency
	Dynamic energy per bit	Static energy per bit
	Area per bit and total area overhead	Bank Capacity
	Controller area overhead per bank	Controller power overhead per bank
Off-chip Memory	Technology	Number of links per device
	Dynamic energy per bit	Nominal Voltage
	Static power per bit	Nominal Frequency
	Device Capacity	Minimum Voltage
	Device Area	Maximum Voltage
	Memory Controller and I/O Area	Access Latency
Network (intra-node and inter-node)	Nominal Voltage	Number of links per mm
	Nominal Frequency	Threshold Voltage
	Nominal Energy per Link	Minimum Voltage
	Nominal Area per Link	Link Latency

Table 8.1: Different technology components.

Network The inter-chip network component is either intra-node or inter-node communication link. In the case of a multi-chip module (MCM) where multiple compute dies and memory devices are integrated on a 2.5D integration substrate within the same package, the inter-die communication is done using high density and energy-efficient links on the 2.5D substrate. These links are considered as intra-node links. On the other hand, the off-package communication links between nodes are considered as inter-node links. The attributes that need to be defined for inter and intra-die communication network components are provided in Table 8.1. In case of a waferscale system, the entire wafer could be considered as a single node.

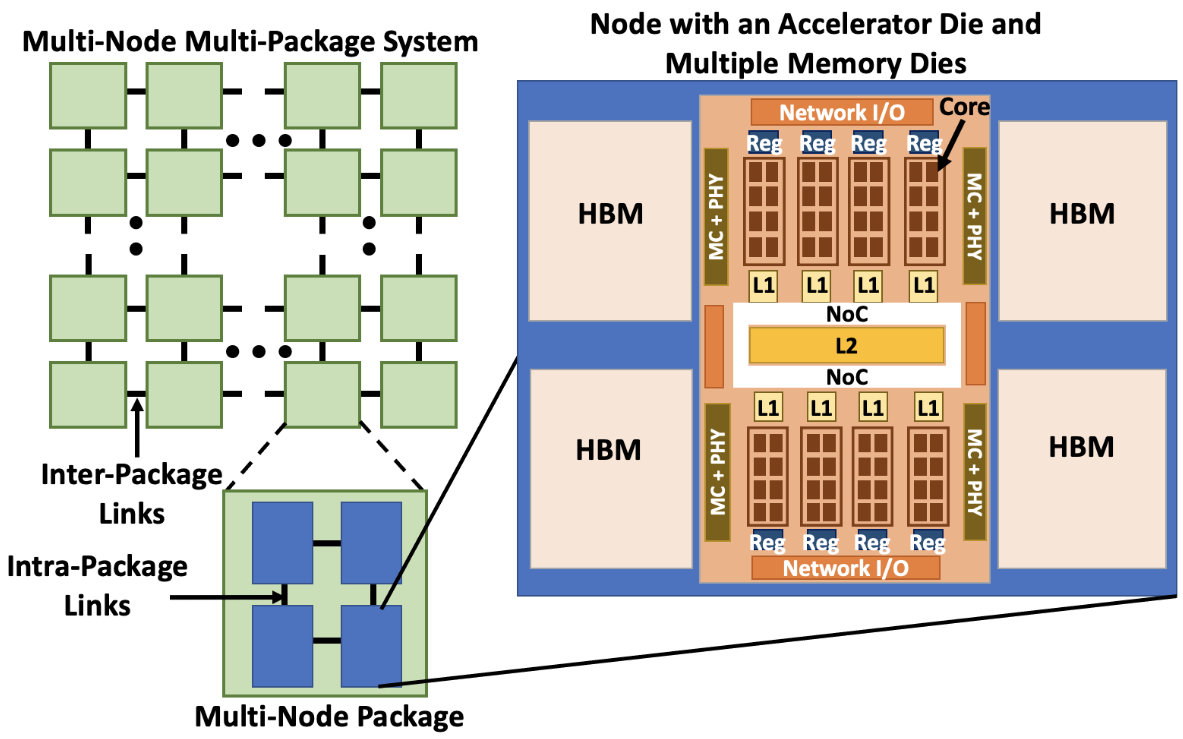


Figure 8.4: Architecture Template: Overview of a hardware system whose characteristics can be configured in DeepFlow.

8.4.2 Architecture Template

Once all system components are instantiated from the technology library, the next step is to hierarchically organize one or multiple components from each category to construct the overall system. Distributed machine learning training is done on scale-out multi-node system, as shown in Figure 8.4. Such a system consists of multiple individually packaged nodes which communicate through off-package interconnects (such as NVLink, Infiniband etc.) that form the inter-node network. Inside each package, there can be multiple different accelerator nodes connected using an intra-node network. Each accelerator within the package typically consists of one accelerator die that is connected to its own off-chip main memory components (such as HBM, as shown in the figure). Each accelerator die itself can be composed of smaller compute units.

DeepFlow provides a rich template that can be used to specify the overall architectural organization of such an accelerator system. The template is used to specify the high-level micro-architectural organization of the compute units, organization of the memory hierarchy, inter-chip network topology (both intra- and inter- node) and the system hierarchy. Next we describe in detail how the template is organized and how different system configurations can be achieved using this template.

Compute unit As shown in the accelerator die architecture in Figure 8.4, compute units are often organized in hierarchies. E.g., in an NVIDIA GPU, multiple tensor cores are bundled in a streaming multi-processor (SM) and the SM as a whole interacts with the cache hierarchy. In DeepFlow one can express such hierarchy by defining *minimal compute units* or MCUs and "MCU bundle". MCU is the smallest granularity of computation that we expose to the tool user. It defines the dataflow model and layout (e.g. MCU can be a systolic array that its height and width are configurable as input). Meanwhile, MCU bundle defines the number of MCUs that are bundled together and are exposed to the first level of memory hierarchy.

In dataflow architectures such as Eyeriss, TPU etc., data can flow directly between different cores. Hence, the tool allows one to define the type of dataflow within a MCU bundle. Currently the performance model supports three types of dataflow: *weight stationary, activation stationary and output stationary*. The tool can also find the best dataflow strategy among the three for any given kernel.

Software runtime, scheduling overheads and the architecture of the cores often restrict the maximum compute utilization. For example, the tensor-cores in NVIDIA V100 incurs fill-drain related under-utilization during tensor loading from the registers and therefore achieves a maximum utilization of 85%. To account for such overheads, a maximum utilization value can be defined which derates the core throughput by that factor.

Memory Hierarchy and Scope The memory hierarchy is defined by initializing multiple memory levels from the highest to the lowest level (i.e., registers to the main memory) as shown in Figure 8.4. Each level of memory has two attributes: (1) Memory technology component from the technology component library which defines the physical attributes of the memory as outlined in Table 8.1, and (2) Scope of the level which defines the set of components from the next level of memory hierarchy that are accessible from this level of memory hierarchy. For example, the keyword 'global' for scope indicates that the memory level is accessible to all the instances of the higher memory levels.

Network Topology In DeepFlow , we support two levels of network hierarchy: intra-package and inter-package. For each level, a different topology (e.g. mesh, torus, crossbar) can be defined.

8.4.3 Hardware Resource Allocation

Hardware design under a limited area and power budget is a fine art of finding the right balance (breakdown of resources) across different u-architectural components. The area and power allocation (and for some components perimeter) of each u-architectural component derives the design and specification of that component. We define resource distribution across different components of the compute chip as input parameters, where resource can be area, power and perimeter of the compute chip. The input definition includes the total area and power budgets for the entire compute node. The total perimeter is inferred from area. The area budget is usually dictated by packaging constraints. For example, if the compute and memory dies are assembled on a 2.5D silicon interposer-based interconnect substrate, the total area of the node will be limited by the maximum size of the interconnect substrate that can be fabricated. Compare this to a waferscale system which houses an entire node on a wafer where the total area budget can be as large as 70,000 mm^2.

On the other hand, the power budget of the node is determined by the cooling infrastructure that is used to extract heat from the node and the power delivery constraints.

We also define the distribution of budget among different components of the compute node as a percentage breakdown. As shown in the YAML snippet below, fractions of the total area is distributed across cores, levels of memory hierarchy and network components. Similarly, the fraction of the compute chip's power and perimeter gets devoted to different hardware components.

```yaml
area_breakdown:
    node_area_budget: 1230 #mm2
    proc_chip_area_budget: 815 #mm2
    core: 0.35
    L2: 0.14
    L1: 0.1
    L0: 0.2
    DRAM: 0.05
    network:
      intra_package: 0.06
      inter_package: 0.1
```

Given the overall resource allocation and distribution, the AGE performs a series of optimizations (voltage-frequency scaling) to find an optimal parameter settings for each u-architectural component. An optimal parameter setting is one that utilizes the most of the allocated budget. Note that an unbalanced resource allocation may leave some of the budget under-utilized. While we allow users to provide a manual breakdown of resources as input, we highly recommend to use SOE (Search and Optimization Engine) to find the best setting which minimizes the total waste (in other words maximize the overall resource utilization). Examples of such output parameters include compute throughput, capacity and bandwidth to the different memory levels and the intra- and inter-node network bandwidths.

8.4.4 Micro-architectural Parameter Generation

Next, the tool generates the micro-architectural parameters for each component of the architecture. Given the architecture template alongside the resource breakdown among the different components, and the technology parameters, we find the maximum throughput for each component. E.g., We find the maximum number of cores that can fit in the given area allocation and find the voltage-frequency points to maximize compute throughput under the power budget. Similarly, for on-chip caches, we find the memory capacity and memory bandwidth at each level that can fit in the area budget while taking the network and controller overhead in to account. For off-chip memories and network interfaces, we use the energy per bit information alongside the PHY area, bump pitch as well interconnect wiring pitch to determine the maximum bandwidth that can be realized on the chip.

These architectural parameters, throughput, bandwidth, capacity etc., are then provided as input to the performance prediction engine. Details about how we model and calculate these parameters is explained next.

Core For deep learning models, the kernels are usually highly parallel in nature and therefore, our goal is to maximize total compute throughput under the area and power budgets allocated for compute. Given the area budget, we first compute the maximum number of MCUs (minimal compute units) that can fit within the area allocated. The nominal frequency and voltage for each

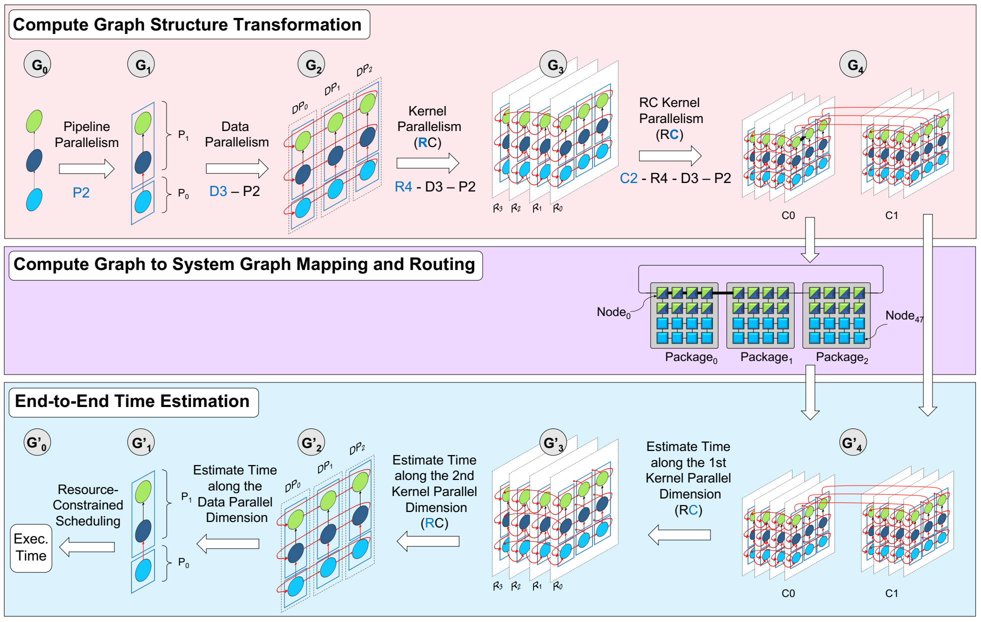
MCU is an input to the model, therefore the nominal power for each MCU and the entire core can be derived very easily. If the nominal power exceeds the power budget, we scale down the frequency and voltage. If we hit the minimum voltage limit set in the component description, we reduce the number of MCUs till we satisfy the total power budget allocated to the compute units. This explains a case where the core design is power-bound and not area-bound.

Once we determine the total number of cores and the frequency of operation, we compute the compute throughput by appropriately scaling the nominal flop rate.

Register and Cache Memory The total area and power budgets allocated to each level of on-chip memory is split between the memory banks and the network circuitry that connects the memory banks at each level to u-architectural components at the next level that are under its scope. We assume this interconnect to have a crossbar topology. The total number of components under its scope and the number of banks in that memory level determine the area and power overheads of the network. We iteratively determine the total number of banks possible at each level of memory hierarchy such that the total area of the banks and the network at every level satisfies the area budget allocation. Once we determine the number of memory banks, we calculate total static power of all the banks and we allocate the remaining power budget to dynamic access energy. The available dynamic energy budget determines the maximum achievable throughput.

Main Memory Main memory has two major components that collectively control the overall capacity and bandwidth but are housed in two different places. Memory controller which is placed on the compute chip, and the memory devices are placed outside the compute die within the same package. The area allocation to each component determines the maximum number of memory devices that can be supported, which in turn determines the total memory capacity. Meanwhile power and perimeter allocation dictates the number of links (that can fit along the compute die), and the frequency of each link which collectively determine the overall off-chip memory bandwidth.

Network The off-chip network links (intra and inter-package) consume both power and area on the compute die. Moreover, the wires need to escape the periphery of the die which gets determined by the interconnect density and the available chip perimeter. The maximum number of links that can be accommodated in the compute die is limited either by the area available to fit in the link I/O cells or the amount of perimeter available for the links to escape the die periphery. Therefore, the tool uses the area per link, the available area budget, wiring density and the die perimeter budget to find the maximum number of links that can fit in the chip. Next, the tool uses the standard voltage-frequency scaling methodology to find the operating point for each link such that the total network related power is within the power budget allocated. The network bandwidth is then calculated by multiplying the total number of links and the operating frequency of each link. We perform this step for the intra-node network and inter-node network separately.



Compute Graph Structure Transformation
G0
G1
G2
G3
G4
Pipeline Parallelism
P2
P1
P0
Data Parallelism
D3 – P2
DP0
DP1
DP2
Kernel Parallelism (RC)
R4 - D3 – P2
R3 R2 R1 R0
RC Kernel Parallelism (RC)
C2 - R4 - D3 – P2
C0
C1
Compute Graph to System Graph Mapping and Routing
Node0
Node47
Package0
Package1
Package2
End-to-End Time Estimation
G'0
G'1
G'2
G'3
G'4
Exec. Time
Resource-Constrained Scheduling
Estimate Time along the Data Parallel Dimension
Estimate Time along the 2nd Kernel Parallel Dimension (RC)
Estimate Time along the 1st Kernel Parallel Dimension (RC)

Figure 8.5: An Example of a Compute Graph Transformation, Device Mapping and Routing, and End-to-End Time Estimation: (top) Cross-edges are shown in red. We only show a subset of cross-edges for kernel parallelism. Blue solid borderlines indicates separate hardware nodes. At every parallelization stage, black hashed lines show graph replication along that dimension. A replica is a graph with a similar structure, however, the kernel size and/or data size could be different for each replica. The original graph is a simple 3-layer feed-forward neural network that is divided into two sub-graphs (P2). Then for each pipeline stage, batch size is distributed across three workers (D3). Then for each data shard of each pipeline stage, the kernels are distributed in a row-column fashion across a 4×2 torus (RC-K4-K2). (middle) Mapping a 4-D hyper-cube into a 2-D mesh: a greedy layout mapped in the following order: kernel(R), kernel(C), pipeline and data. The bolded black edge in G4 is mapped onto a 4-hop path in the system graph. (bottom) backward pass time estimation.

8.5 Compute Graph Transformation and Device Mapping Engine

Given the ML model description (in form of a *compute graph*) and the distributed system topology (in form of a *system graph*), we find an optimal mapping from vertices and edges in the compute graph to hardware nodes and network links in the system graph. However, before mapping, we transform the compute graph into a *super-graph* to reflect the parallelism strategies specified as input.

8.5.1 Compute Graph Structure Transformation

Each parallelism strategy is a form of graph transformation where the sub-graph to be replaced is a single node, so essentially all nodes would be replaced with the same replacement graph. For example, data parallelism (with the ring-all-reduce implementation) would *replace* each node in the original graph with a ring of length N (for an N-data parallel strategy). The new edges on the ring will be marked as *cross-edge* to capture the fact that they connect compute nodes hosted on separate devices. Kernel parallelism (e.g. `RC_K{KP_1}_K{KP_2}`) would *replace* each node in the compute graph with a 2-dimensional torus of `KP_1 X KP_2` dimension (assuming the reduction algorithm along each dimension is ring-all-reduce). Similarly, new edges on the torus would be marked as cross-edge. Pipeline parallelism *replaces* a single node with a single node, so essentially the graph structure does not change: pipeline parallelism simply slices the original graph into multiple sub-graphs, each hosted on a separate hardware node and computation across consecutive batches can be pipelined through consecutive sub-graphs. Edges connecting sub-graphs would be marked as cross-edge. Figure 8.5 shows the composition of multiple parallelism strategies applied in sequence (pipeline, data and kernel parallelism, respectively). G_0 is the original compute graph and G_4 is the final transformed graph.

8.5.2 Device Mapping and Routing Engine

Data parallelism, kernel parallelism and pipeline parallelism would require that each parallel shard to be hosted on a separate physical device. Hence, device mapping happens at the granularity of a parallel shard. We want parallel shards that are close in the parallel space to be mapped onto nodes that are close in the physical space to minimize communication. However, the transformed graph usually has higher dimension than the system graph. Figure 8.5 shows such example, where the final transformed graph (G_4) is 4-D hypercube and the system graph is a 2-D torus. Therefore, it will not be possible to map all adjacent nodes in the compute graph to adjacent nodes in the system graph. We adopt a greedy approach to conduct such mappings: We start with a parallel dimension, map all parallel shards along that dimension to adjacent nodes in the hardware. If the number of shards along the parallel dimension is larger than the hardware dimension we are mapping onto, we wrap-around to the next immediate dimension. We continue this process along other dimensions in a specific order, until all nodes are mapped. The order at which we walk along the parallelism dimensions results in different mappings. For 4 different parallelism strategies, we explore $(4!) = 24$ possible orderings to pick the best mapping. Once node mapping is determined, we take a last step to map edges to physical links. An edge that connects to adjacent node in the compute graph may map to a multi-hop path as shown in Figure 8.5. As a result, one physical link would be shared across multiple edges. The number of logical edges sharing a physical link is an important factor for effective bandwidth estimation. We use $X-Y$ routing to map edges in the compute graph to paths in the system graph. Overall, the whole transformation step followed by device mapping is necessary to find an accurate estimation of *edge* timing.

8.6 Performance Prediction Engine

Once mapping is decided for each node and each edge in the transformed graph, performance prediction engine estimates timing for each node and each edge. We then use a resource-constrained scheduling algorithm to find the end-to-end timing. We explain these steps in more details.

8.6.1 Hierarchical Roofline

We use hierarchical roofline analyses [221] to predict the timing of each node in the transformed compute graph. We estimate the operational intensity (`OI = #flops/#memory accesses`) of each node of the compute graph first. For systems with multiple levels of memory hierarchy, we adopt a hierarchical roofline analysis. Hierarchical roofline predicts if an application is compute-bound, L1-bound, L2-bound, memory-bound, etc. To accomplish that, we search over the space of tiling strategies at each level of memory hierarchy and accurately estimate the number of memory accesses to different levels of memory hierarchy. We explain this in detail next.

8.6.2 Memory Hierarchy Modeling

The number of accesses to each level of memory hierarchy is a function of the underlying hardware (memory capacity at each level) and the algorithmic implementation (loop ordering and tiling strategies).

For any given input configuration, we explore N^L random tiling strategies which meet the memory capacity requirement at each level. N is the number of tiling strategies at each level and L is the number of levels of memory hierarchy. Empirically, we found that for $L = 3$, $N \approx 20$ results in a reasonably accurate estimation.

For a given tiling strategy, it is easy to find the number of times each tile needs to be re-streamed from the next level of memory hierarchy. We start from the lowest level (e.g. main memory for a GPU-like architecture) and walk upward to estimate the number of accesses. For each level (except for the highest level), the number of accesses from the higher level is dictated by the tiling parameters at both levels. For the highest level, the number of accesses is determined by the dataflow strategy exploited at MCU units.

8.6.3 DataFlow Model

The number of accesses to the highest level of memory hierarchy (i.e. register file) will be determined by the number of instructions executed in the execution engine and the dataflow strategy governing mapping and communication between those engines (e.g. weight stationary, activation stationary and output stationary [307, 308]). The execution engine structure dictates how many times a piece of data could be reused internally before accessing the register file. We refer to this number as *reuse factor (K)*. In a 2-D systolic array with size N_x and N_y, and an input GEMM with size $T0_x$, $T0_y$ and $T0_z$ (which is the tile sizes at L_0), each data element could be reused $T0_x/N_x$ or $T0_y/N_y$ or $T0_z/N_z$

times, depending on which matrix is stationary. Given the reuse factor, we estimate the number of accesses to register files as follows:

$$\#RegAccess = \#Flops \times \frac{N_x.N_y + K.N_x + K.N_y}{2.K.N_x.N_y} \tag{8.1}$$

8.6.4 Inter/Intra-Package Communication Modeling

We use throughput analysis to calculate the inter/intra-package communication timing: As discussed in Section 8.5, compute graph to system graph mapping generates the information about logical edge to physical link mapping. The effective bandwidth for each link is downrated by the number of logical edges sharing the link. The physical link with minimum effective bandwidth is the bottleneck bandwidth which we use for time estimation for all edges.

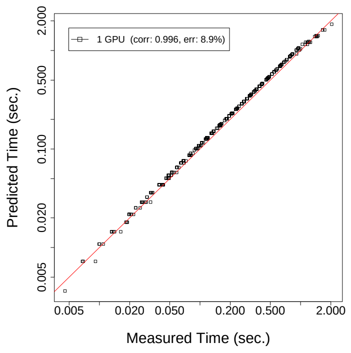


Figure 8.6: GEMM Validation on P4.

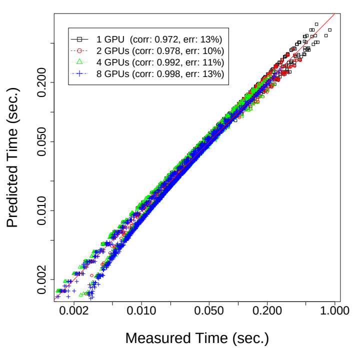


Figure 8.7: GEMM Validation on DGX.

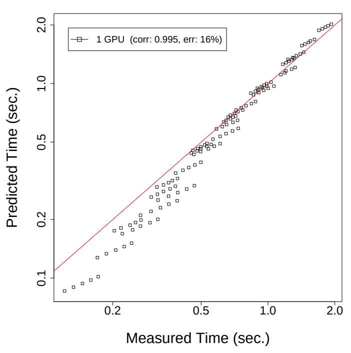


Figure 8.8: LM Validation on V100.

8.6.5 End-to-End Time Estimation

We use an event-driven simulation to estimate end-to-end timing. Event-driven simulation is basically a resource-constrained critical path analysis. Since multiple compute nodes can map into the same hardware node, event-driven simulation is necessary to avoid resource conflicts/hazard and respect resource scheduling constraints (e.g. not more than *k* kernels can run in parallel on each hardware node).

We apply event-driven simulation at the original compute graph granularity where the only parallelism to account for is pipeline parallelism: data parallelism and kernel parallelism would essentially create replicas of the original graph (where the kernel size and/or data size would be different for each node). Given that all replicas by definition are hosted on separate hardware nodes, they can all start and stop at the same time (assuming a homogeneous distribution of data along model replicas and homogeneous distribution of sub-kernels across data replicas) and their timing is deterministic. Hence, there is no need for event-driven simulation at the super-graph granularity.

Figure 8.5 shows an example of an end-to-end time estimation of a backward pass for a simple 3-layer feed-forward neural network, with 2-level pipeline parallelism (P2), 3-level data parallelism (3D), and 8-level kernel parallelism (`R4-C2`). We start with kernel parallelism and then data parallelism to resolve the compute time for each node in the graph (this can be applied in any order). Once time for all nodes in the original compute graph is resolved, we use event-driven simulator to account for pipeline parallelism and resource scheduling constraints.

8.7 Design Space Exploration Engine

We denote the set of hardware ***parameters*** to explore as $W = \{\{A_i\}_0^{H-1}, \{P_i\}_0^{H-1}, \{R_i\}_0^{H-1}\}$, where H is the number of micro-architectural components in the hardware accelerator node, and A_i, P_i and R_i capture the percentage of the overall area, power and perimeter allocated to each component, respectively.

Our **objective** is to find the optimal W^* that minimizes the total run time, $f(W)$, such that $\sum_{i=0}^{H-1} A_i \leq 1$, $\sum_{i=0}^{H-1} P_i \leq 1$, and $\sum_{i=0}^{H-1} R_i \leq 1$. The objective function f does not have a closed form, but we can calculate it by querying the performance model (CrossFlow). This problem is an example of a *constrained black-box continuous* optimization. Since the objective function evaluation (i.e. querying CrossFlow) is considerably cheap (milliseconds to multiple seconds), we use a variation of projected gradient descent (GD) optimization to solve for W^* (see 8.7). Empirically, we found that GD with exponential averaging in the parameter space (rather than gradients) works the best for our problem.

$$
\begin{aligned}
&W_t = W_{t-1} - \eta \mathrm{g_t} \qquad \hat{W}_t = \frac{W_t}{||W_t||} \\
&M_t = \beta M_{t-1} + (1-\beta)\hat{W}_t \\
&W_t = \mathrm{Project}(M_t) \quad \text{onto} \quad C_A, C_P, C_R
\end{aligned}
\tag{8.2}
$$

Where W_t and $\mathrm{g_t}$ are the input parameters and gradients at time step t, η is the learning rate and β is the discounting factor. We repeat the update steps shown above until convergence or the maximum number of steps (T), whichever conditions happens earlier. To alleviate cases where optimization might get stuck in local minima, we repeat the steps above from S different starting points and return the best result. Empirically, we found that $T = 100$ and $S = 10$ are sufficient to find a near optimal solution.

8.8 Validation

We validate our performance prediction model against execution time measured on real systems (Nvidia P4 with 1 GPU and an NVIDIA DGX-1 system with 8 V100 GPU cards), running the most important kernels (distributed GEMM) and end-to-end applications (large-scale language models).

In particular, we study language models (LM) for validation and case study as it is deemed to be one of the most challenging applications to scale [309], and is very costly to train [310]. All applications are implemented in Tensorflow 2.0. We use CrossFlow to predict the runtime, which can take anywhere from milliseconds to 20 seconds.

Validation Space For GEMM validation, we look at a space of more than 4000 GEMM kernels of different shapes and parallelism strategies, where input (`m`), output (`n`) and inner dimensions (`k`) varying from 4K to 32K in steps of 4K, and parallelized across 1, 2, 4, or 8 GPUs, using both Row-Column and Column-Row distributed parallelism strategies. For LM validation (2-layer LSTM model), we look into a space of 125 configurations, where `Batch Size`, `Hidden Dimension` and `Vocab Size` varying from 2K to 6K in steps of 1K. We report the correlation (corr), and also the mean relative error (err) to quantify the quality of our predictions.

Results Figure 8.6 shows the validation results on Nvidia P4 GPU card. On the X-axis, we show

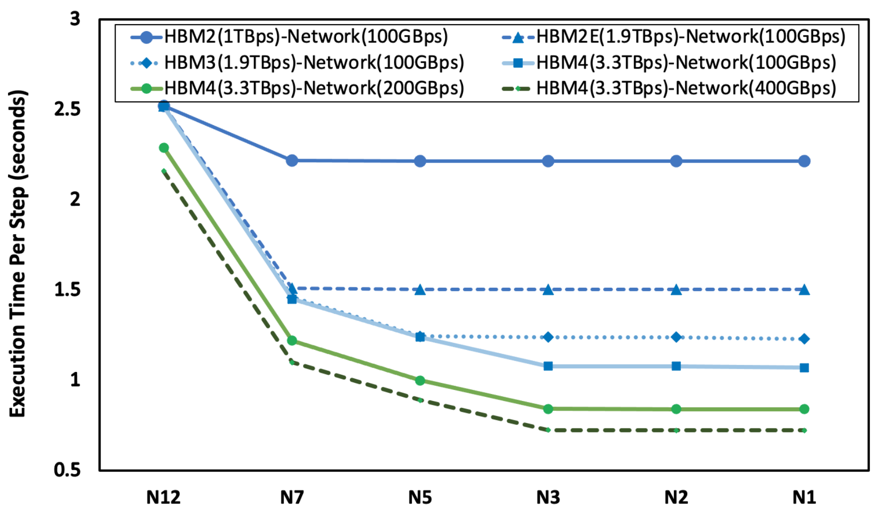


Figure 8.9: Technology Scaling: Effect of scaling different logic, memory and network technologies is shown.

the measured time (in log-scale), and on the Y-axis, we show the predicted time (in log-scale). As shown, predictions and measurements are highly correlated (0.996) and the average of the absolute relative error is small (8.9%). Figure 8.7 shows that CrossFlow predictions on a DGX-1 system across 1, 2, 4 and 8 V100 GPU cards are well correlated (0.972-0.998) and low error (10%-13%). Figure 8.8 shows the performance of LM on V100 GPU card. Similarly, we can predict performance with high correlation (0.996), and low error (16%). A constant pattern visible across all results is the performance prediction deviation from measurement on real hardware for small kernels. This is expected as Tensorflow 2.0 time measurement hooks include all the software stack latency; while this overhead is negligible for large kernels, it accounts for a large portion of total run-time if the kernel is very small. This indicates the tool outcome would be more reliable for large kernels and large models.

8.9 Case Studies

DeepFlow is a pathfinding framework with studies and use cases spanning semiconductor technology development, micro-architecture, neural network models, and algorithmic parallelization techniques. In this section, we give few example case studies for a large-scale language model (hidden dim:

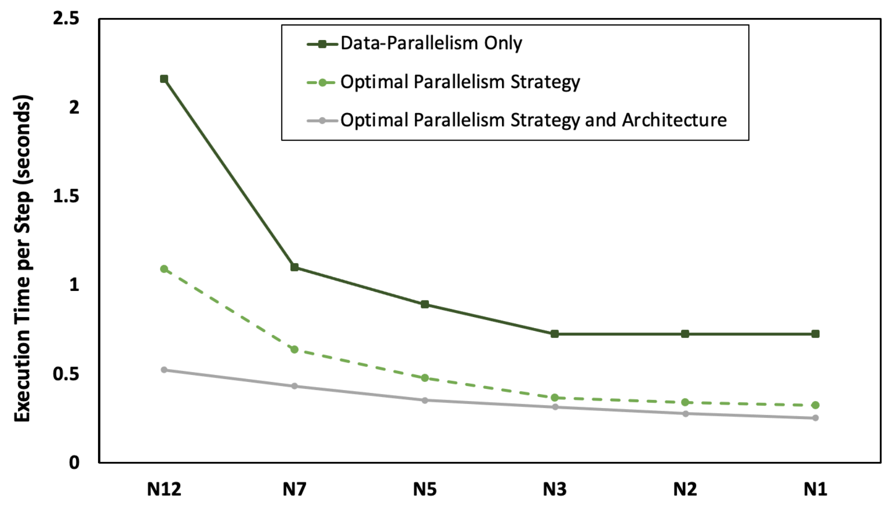


Figure 8.10: Co-optimizing parallelism strategy and hardware architecture design.

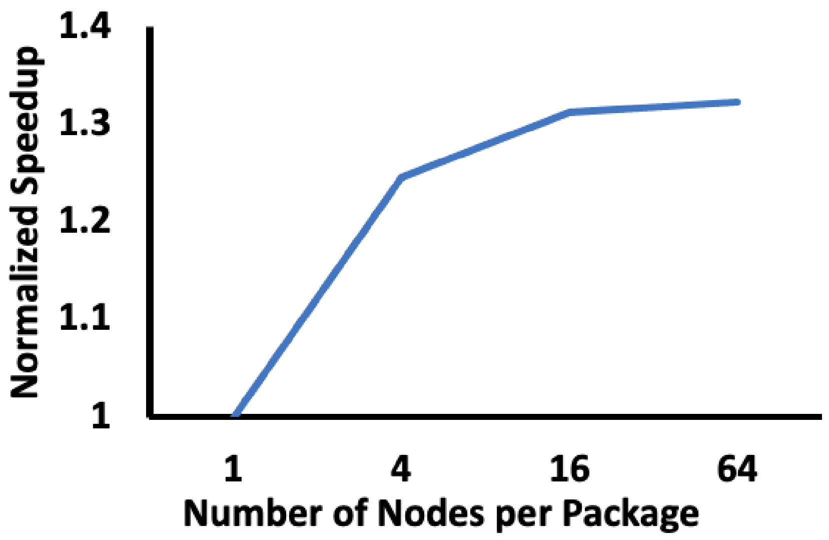


Figure 8.11: Performance improvement from multi-node package

16384, global batch size: 16K, vocab size: 800K, number of layers: 2, sequence length: 20) distributed across 512 nodes. For future technology exploration, we study 7 consecutive **logic** technology nodes (from 12nm (N12) to 1nm (N1). Based on the recent scaling trends for logic technologies [311,312], we assume area and power scale by 1.8x and 1.3x from one node to the next for iso-performance), 4 different **memory** technologies (HBM2 (1 TB/s), HBM2e (2 TB/s), HBM3 (projected 2.6 TB/s [313]), and HBM4 (projected 3.3 TB/s)) and 3 different **network** technologies (Infiniband-NDR (100 GB/s), XDR (200GB/s) and GDR (3.3 TB/s)). [2]

8.9.1 Impact of Technology Scaling

The first question we seek to answer is where the performance bottlenecks are across the stack and which technology could provide the maximum end-to-end performance benefit? Semiconductor technology development decisions are increasingly driven by machine learning as the workload. Many of these decisions trigger multi-billion dollar, multi-year investments.

Figure 8.9 shows the impact of scaling logic, memory, and network technology for a very large-scale language model. For these experiments, we assume that the thermal power budget (TDP) per node is fixed at 300W and the area budget is fixed at 850 mm^2 for the chip, and the fraction of power and area devoted to each component on the chip is constant. Also, we fix the parallelization strategy at data parallelism.

Logic scaling improves compute throughput, and also caching capacity and bandwidth, but only to a smaller extent. Going from N12 to N7, we observe a jump in performance irrespective of memory technology. This is because at N12, the performance of a significant number of kernels is L2 bandwidth bound. At N7, the L2 bandwidth and capacity improve enough for HBM bandwidth to become the new bottleneck. Therefore, with improvement in HBM bandwidth, the balance can shift back again to caches and saturation point can be further improved with logic scaling, hence saturation point shifts further to the right. This trend continues up to N3. Beyond N3, even at very high memory bandwidth (3.3 TB/s) and network bandwidth (400 GB/s) as cache capacity and

[2]The caveat to these results (as with any pathfinding study with DeepFlow) is that if the system architecture or dataflow or neural network is radically different (e.g., this study assumes that same node is homogeneously replicated within the package), the conclusions may change.

bandwidth are the main bottlenecks. Since the on-chip network connecting MCUs to cache and the cache controller overhead scale along with number of cache banks and the number of MCUs (which scale at $\sim 1.8\times$ per technology node), the cache capacity as well as bandwidth increase only marginally at N2 and N1.

8.9.2 Co-optimizing Technology, Parallelism Strategy and Hardware Architecture Design

We argue that technology and architecture exploration should be co-optimized along with parallelism strategy. Figure 8.10 shows the importance of co-optimizing technology with parallelism and hardware design in an incremental fashion.

Three key takeaways from these results are: (1) Parallelism strategy optimization alone can offer $\sim 2\times$ performance improvement. (2) Co-optimizing architecture and parallelism strategy offers meaningful benefits for mature technology (12nm ad 7nm) nodes. But for more advanced technology nodes, only marginal benefits (20%-30%) can be gained on top of parallelism strategy optimization. (3) For current and near-future technology nodes, co-optimizing for model architecture can provide as much benefit as scaling technology nodes (by almost two generations).

8.9.3 Effect of Multi-Node Package

Next, we evaluate the performance improvement that multi-node packaged systems (e.g., MCM-GPU [314], waferscale-GPU [315], Tesla Dojo [316]) can provide in a distributed training setup (see Fig. 8.11). We assumed 2TB/s link bandwidth for the intra-package links and performed both parallelism and architecture search for each case.

Couple of key takeaways from these experiments were: (1) Increasing the number of nodes in a package improves overall performance by roughly 32% at best. (2) Beyond 4-nodes per package, performance improvement is marginal. Since ultra-large packages or waferscale integration dramatically worsens cost, we believe that such technologies may not be worthy investments for scaling large language model training. These conclusions hold across multiple different batch sizes, hidden dimension sizes and intra-node link bandwidths.

8.10 Conclusion

This work is the first effort to explore the cross-stack impact of technology scaling, model scaling and architecture innovations from a holistic perspective, and at the same time considering real-world design constraints like area and power budget for deep learning training. We proposed DeepFlow, a performance modeling framework that enables a cross-stack analysis for hardware-software-technology co-design at-scale. We envision DeepFlow to be used by *ML practitioners* (to decide what hardware to use to maximize their utilization, or simply predict their hypothetical model architecture performance which might not be realizable in today's hardware for many reasons including capacity limitation), by *system designers* (to decide what hardware accelerators they need to acquire or build from scratch to meet their application needs, what new technologies to invest in, etc.), and finally by *technology experts* (to guide future technology development by assessing its impact all the way across the stack, at scale).

CHAPTER 9

Conclusion

This chapter reviews the key contributions of this dissertation and outlines directions for future work.

9.1 Overview of Contributions

A radically different approach - packageless and scale-out processing, to build efficient and high performance processor systems has been proposed and developed in this dissertation. Moreover, the set of pathfinding tools developed here enable rapid exploration of technology, architecture and software for chiplet assemblies and scale-out systems.

9.1.1 Packageless Processors

Processor packages can significantly impact the bandwidth, allowable TDP, and area taken up by a processor. We proposed packageless processors - processors where the packages are removed and PCB-based integration is replaced by a Silicon Interconnection Fabric, a novel interconnection technology that involves mounting dies directly on a silicon wafer using copper pillar-based I/O pins. We showed that packageless processors can have one to two orders of magnitude higher memory bandwidth, up to 70% higher allowable TDP, and 5X-18X lower area than conventional packaged processors. These benefits can be exploited to increase processor performance. For a set of NAS and PARSEC benchmarks, we showed performance improvements up to 58% (16% average), 136% (103% average), and 295% (80% average) resulting from improved memory bandwidth, processor TDP and processor footprint respectively. For the same performance, packageless processing reduces compute subsystem footprint by up to 76% or equivalently increases TDP by up to 2X. The benefits from packageless processing should only increase with increasing I/O and performance demands of emerging applications and processors.

9.1.2 Pathfinding for Chiplet Integration Technologies

As conventional technology scaling becomes challenging, chiplet integration provides a viable pathway to compose larger systems using smaller, high yielding dies. Therefore recently, there has been a proliferation of different chiplet integration technologies. However, the success of this approach depends upon optimizing the performance benefits and cost for different use case scenarios. In this work, we develop a pathfinding methodology for interconnect technologies and use it to study inter-chiplet interconnect performance and energy as a function of dimensional and technology parameters. We demonstrate that a holistic approach considering features of the integration technology, chiplet technology and processing techniques. Our analysis indicates that beyond certain point, dimensional scaling (wire and bump pitch) provides marginal benefit in terms of energy-per-bit and bandwidth density; while other factors such as ESD and chip dicing technologies may provide additional levers for further interconnect scaling.

9.1.3 Chiplet Selection for Application-specific System

Chiplet integration technologies enable cost-effective system customization. In this work, we have developed the first multi-chiplet processor design space exploration framework. Our results clearly show benefits of using multiple chiplet assembly based processors to service diverse workloads (35% improvement in EDP over a single best average system), and advantages of cognizance of chiplets during optimization (over 2x reduction in chiplets required for a heterogeneous CMP example). We also identify the different factors that affect the cost of building multiple systems using chiplets as well as the characteristics of the systems/chiplets chosen under different use contexts. Using the framework we show that emerging chiplet assembly approaches are very promising when total cost of design and manufacturing is considered (up to 72% benefit in cost over SoC) while satisfying energy and performance requirements of every workload. This cost benefit strongly depends on system size and technology maturity.

9.1.4 Waferscale GPU Architecture

Silicon-Interconnection Fabric (Si-IF)-based integration, where pre-manufactured dies are directly bonded on to a silicon wafer, may enable waferscale processors without the yield issues of building a large waferscale chip. Therefore, time is ripe to revisit waferscale architectures. In this work, we showed that it is feasible and useful to architect a modern day waferscale system. Using a waferscale GPU as a case study, we showed that while a 300 mm wafer can house about 100 GPU modules (GPM), only a much scaled down GPU architecture with about 40 GPMs can be built when physical concerns are considered. We also studied the performance and energy implications of waferscale architectures. We showed that waferscale GPUs can provide significant performance and energy efficiency advantages (up to 18.9x speedup and 143x EDP benefit compared against equivalent MCM-GPU based implementation on PCB) without any change in the programming model.

9.1.5 Design Methodologies for Chiplet-based Waferscale Systems

In this work, we attempted to build the largest ever (2048-die) chiplet assembly based waferscale processor (at least 10X larger than largest known commercial chiplet-based systems) and develop a design methodology for the same. We highlight challenges and potential solutions in power delivery, clock distribution, network design, design for testability and fault-tolerance for waferscale systems. Our ongoing work aims at characterizing the waferscale prototype and developing design methods for higher-power waferscale systems.

9.1.6 Multi-GPU Deep Learning Parallelization Strategies

This work demonstrates the benefits of choosing the correct parallelization strategy by combining model-parallelism (MP) with data-parallelism (DP) to overcome the inherent scaling and statistical efficiency losses that data-parallel training has at scale. We analyze the end-to-end training time of DP to understand how scaling and statistical efficiency loss impacts training scalability, and show that the MP speedup achieved for a given DL model is critical to the overall scalability of a hybrid parallelization strategy. We demonstrate that when the global batch size in DP grows to a point where DP-only training speedup drops off significantly, MP can be used in conjunction with DP to continue improving training times beyond what DP can achieve alone.

9.1.7 DeepFlow

DeepFlow is a rapid-yet-accurate cross-stack pathfinding framework for scale-out deep learning training. We envision DeepFlow to be used by *ML practitioners* (to decide what hardware to use to maximize their utilization, or simply predict their hypothetical model architecture performance which might not be realizable in today's hardware for many reasons including capacity limitation), by *system designers* (to decide what hardware accelerators they need to acquire or build from scratch to meet their application needs, what new technologies to invest in, etc.), and finally by *technology experts* (to guide future technology development by assessing its impact all the way across the stack, at scale).

9.2 Directions for Future Work

While this dissertation lays the ground work for chiplet-assembly based processing and waferscale systems and shows the potential benefits of adopting these technologies, many avenues of research and development still exist. Also, there is a large set of problems which can be targeted using novel architectures based on these technologies. Some of the potential future work directions are summarized below:

9.2.1 Waferscale Memory and Networking Systems

The next-generation AI, big-data and scientific applications which would require processing massive volumes of irregular data (graphs, databases, large embedding tables etc.), would require scale-out systems with thousands of cores and 10s of TBs of main memory. To efficiently process the data, the system should support very high random-access memory bandwidth at very low latency. Similar requirements are arising in the cloud computing infrastructure as well. Recent studies have pointed towards a worrisome trend of increasing hardware resource utilization in cloud computing systems.

As a solution, industry has started looking into ‘dis-aggregation’, where the compute, memory, and accelerator resources would be disaggregated into their respective pools and connected using data center scale networks. These networks need to support orders of magnitude more data volume and shuttle data between different pools at extremely low latency. Waferscale architectures can meet the aggressive performance targets of these applications and use cases. Couple of candidate architecture includes building (i) a waferscale memory pool, and (ii) waferscale network switch. Many commodity processors can share TBs of memory on a waferscale memory pool. Unlike today’s systems, where remote memory access is very costly, a waferscale memory could provide very high performance for large shared memory systems. Similarly, for disaggregated data centers, waferscale network switches is a solution which can provide 10-100s of TB/s bandwidth at sub-100 ns latency (versus 5-10s of us latency today). To this end, we need to systematically study the landscape of network design in disaggregated datacenters and develop novel network architectures using waferscale networks coupled with other interconnect technologies such as emerging highly efficient multi-wavelength optical interconnects.

9.2.2 Fault Modelling, Design-for-Test and Physical Design Infrastructure for Chiplet Assemblies

Though chiplet based integration platforms promise to alleviate the yield issues that monolithic integration faces, getting known-good-dies as well as a known-good-substrate is difficult and challenging. Also, faults can occur during die bonding or dies can develop faults at runtime which can hamper reliability of waferscale systems. Even though this problem looks similar to interposer-based systems, the scale of the system can often make this a unique and a challenging problem. E.g., waferscale systems need to tolerate a few faulty dies or links because the cost of a full system is too high to discard under a few faults. We need to investigate the yield loss mechanisms when dies from different heterogeneous processes are assembled on these substrates and develop fault models for chiplet integration technologies. Based on the findings, we could develop both physical redundancy mechanisms as well as architectural fault mitigation mechanisms for such systems. Future work would also focus on developing runtime test and low-overhead fault-aware routing mechanisms for such architectures. On the EDA side, today’s tools don’t scale to waferscale sizes.Though we developed a simple place and route tool for designing prototype waferscale systems, automated place and route solutions for chiplet integration substrate and waferscale substrate design is needed. One could integrate the fault models and automate the redundancy generation schemes in such a tool. This will help us move towards a SoC-like design flow for systems-in-package (SiP) and systems-on-wafer (SoW).

REFERENCES

[1] D. Edwards and H. Nguyen, "Semiconductor and IC Package Thermal Metrics." Application Report, Texas Instruments, 2016, `http://www.ti.com/lit/an/spra953c/spra953c.pdf`, (accessed August 1, 2017).

[2] Intel Corp., *Intel Pentium 4 Processor in the 423-pin Package Thermal Design Guidelines*, November 2000.

[3] H. R. Shanks, P. D. Maycock, P. H. Sidles, and G. C. Danielson, "Thermal Conductivity of Silicon from 300 to 1400°K," *Phys. Rev.*, vol. 130, pp. 1743–1748, Jun 1963.

[4] A. Bajwa, S. Jangam, S. Pal, N. Marathe, T. Bai, T. Fukushima, M. Goorsky, and S. S. Iyer, "Heterogeneous Integration at Fine Pitch ($\leq 10\mu$m) Using Thermal Compression Bonding," in *IEEE 67th Electronic Components and Technology Conference (ECTC)*, pp. 1276–1284, May 2017.

[5] M. Matsuo, N. Hayasaka, K. Okumura, E. Hosomi, and C. Takubo, "Silicon interposer technology for high-density package," in *2000 Proceedings. 50th Electronic Components and Technology Conference (Cat. No.00CH37070)*, pp. 1455–1459, 2000.

[6] Y. Chuang, C. Yuan, J. Chen, C. Chen, C. Yang, W. Changchien, C. C. C. Liu, and F. Lee, "Unified methodology for heterogeneous integration with cowos technology," in *2013 IEEE 63rd Electronic Components and Technology Conference*, pp. 852–859, 2013.

[7] R. Mahajan, R. Sankman, N. Patel, D. Kim, K. Aygun, Z. Qian, Y. Mekonnen, I. Salama, S. Sharan, D. Iyengar, and D. Mallik, "Embedded multi-die interconnect bridge (emib) – a high density, high bandwidth packaging interconnect," in *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, pp. 557–565, 2016.

[8] "R-Tools 3-D Heat Sink thermal Modelling." `http://www.r-tools.com/`.

[9] S. Pal, D. Petrisko, A. A. Bajwa, P. Gupta, S. S. Iyer, and R. Kumar, "A case for packageless processors," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 466–479, 2018.

[10] S. Pal, D. Petrisko, R. Kumar, and P. Gupta, "Design space exploration for chiplet-assembly-based processors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 4, pp. 1062–1073, 2020.

[11] S. Pal, D. Petrisko, M. Tomei, P. Gupta, S. S. Iyer, and R. Kumar, "Architecting waferscale processors - a gpu case study," in *IEEE International Symposium on High Performance Computer Architecture*, pp. 250–263, 2019.

[12] S. Pal, J. Liu, I. Alam, N. Cebry, H. Suhail, S. Bu, S. S. Iyer, S. Pamarti, R. Kumar, and P. Gupta, "Designing a 2048-chiplet, 14336-core waferscale processor," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 1183–1188, IEEE, 2021.

[13] S. Pal, E. Ebrahimi, A. Zulfiqar, Y. Fu, V. Zhang, S. Migacz, D. Nellans, and P. Gupta, "Optimizing multi-gpu parallelization strategies for deep learning training," *IEEE Micro*, vol. 39, no. 5, pp. 91–101, 2019.

[14] M. Bohr, "A 30 Year Retrospective on Dennard's MOSFET Scaling Paper," *IEEE Solid-State Circuits Society Newsletter*, vol. 12, pp. 11–13, Winter 2007.

[15] K. Atasu, L. Pozzi, and P. Ienne, "Automatic Application-specific Instruction-set Extensions Under Microarchitectural Constraints," in *40th Annual Design Automation Conference*, (New York, NY, USA), pp. 256–261, ACM, 2003.

[16] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate Computing and the Quest for Computing Efficiency," in *Proceedings of the 52d Annual Design Automation Conference*, DAC '15, (New York, NY, USA), pp. 120:1–120:6, ACM, 2015.

[17] G. H. Loh, "3D-Stacked Memory Architectures for Multi-core Processors," in *35th Annual International Symposium on Computer Architecture (ISCA)*, (Washington, DC, USA), pp. 453–464, 2008.

[18] N. Z. Haron, S. Hamdioui, and S. Cotofana, "Emerging non-CMOS nanoelectronic devices - What are they?," in *4th IEEE International Conference on Nano/Micro Engineered and Molecular Systems*, pp. 63–68, Jan 2009.

[19] S. S. Iyer, "Heterogeneous Integration for Performance and Scaling," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 6, pp. 973–982, July 2016.

[20] J. H. Lau, *Flip Chip Technologies*. New York, NY, USA: McGraw-Hill, 1996.

[21] UG1099, *Recommended Design Rules and Strategies for BGA Devices*. Xilinx Inc, 1 ed., March 2016.

[22] TE Connectivity Corporation, *LGA 3647 SOCKET AND HARDWARE*, 2017.

[23] Intel Corp., *Land Grid Array (LGA) Socket and Package Technology*, Sept 2009.

[24] Intel Corp., *Ball Grid Array (BGA) Packaging*, 2000.

[25] "Land grid array." `https://en.wikipedia.org/wiki/Land_grid_array`, (accessed July 29, 2017).

[26] W. R. Mann, F. L. Taber, P. W. Seitzer, and J. J. Broz, "The leading edge of production wafer probe test technology," in *2004 International Conference on Test*, pp. 1168–1195, Oct 2004.

[27] Y. Liu, S. L. Wright, B. Dang, P. Andry, R. Polastre, and J. Knickerbocker, "Transferrable fine pitch probe technology," in *2014 IEEE 64th Electronic Components and Technology Conference (ECTC)*, pp. 1880–1884, May 2014.

[28] M. Luthra, "Process challenges and solutions for embedding Chip-On-Board into mainstream SMT assembly," in *Proceedings of the 4th International Symposium on Electronic Materials and Packaging*, pp. 426–433, Dec 2002.

[29] J. H. Lau, *Chip On Board*. Springer USA, 1994.

[30] G. V. Clatterbaugh, P. Vichot, and J. Harry K. Charles, “Some Key Issues in Microelectronic Packaging,” in *Johns Hopkins APL Technical Digest*, vol. 20, pp. 34–49, Oct 1999.

[31] Intel, “Intel Xeon Phi,” (2014).

[32] “Broadwell - Microarchitectures - Intel.” `https://en.wikichip.org/wiki/intel/microarchitectures/broadwell`.

[33] “Atom - Intel.” `https://en.wikichip.org/wiki/intel/atom`.

[34] “Micron DDR4 SDRAM Part Catalog.” `https://www.micron.com/products/dram/ddr4-sdram/8Gb#/`.

[35] “Intel Xeon Processor E5 Family.” `http://ark.intel.com/products/series/59138/Intel-Xeon-Processor-E5-Family`.

[36] J. F. McDonald, E. H. Rogers, K. Rose, and A. J. Steckl, “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*, vol. 21, pp. 32–39, Oct 1984.

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

[38] A. Sodani, R. Gramunt, J. Corbal, H. S. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y. C. Liu, “Knights Landing: Second-Generation Intel Xeon Phi Product,” *IEEE Micro*, vol. 36, pp. 34–46, Mar 2016.

[39] N. E. Jerger, A. Kannan, Z. Li, and G. H. Loh, “NoC Architectures for Silicon Interposer Systems: Why Pay for more Wires when you Can Get them (from your interposer) for Free?,” in *47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 458–470, Dec 2014.

[40] Y. Iwata and S. C. Wood, “Effect of fab scale, process diversity and setup on semiconductor wafer processing cost,” in *IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop*, pp. 237–244, 2000.

[41] F. Yazdani, “A novel low cost, high performance and reliable silicon interposer,” in *IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–6, Sept 2015.

[42] S. L. Wright, R. Polastre, H. Gan, L. P. Buchwalter, R. Horton, P. S. Andry, E. Sprogis, C. Patel, C. Tsang, J. Knickerbocker, J. R. Lloyd, A. Sharma, and M. S. Sri-Jayantha, “Characterization of micro-bump C4 interconnects for Si-carrier SOP applications,” in *56th Electronic Components and Technology Conference*, 2006.

[43] B. Dang, S. L. Wright, P. S. Andry, C. K. Tsang, C. Patel, R. Polastre, R. Horton, K. Sakuma, B. C. Webb, E. Sprogis, G. Zhang, A. Sharma, and J. U. Knickerbocker, “Assembly, Characterization, and Reworkability of Pb-free Ultra-Fine Pitch C4s for System-on-Package,” in *2007 Proceedings 57th Electronic Components and Technology Conference*, pp. 42–48, May 2007.

[44] L. D. Cioccio, P. Gueguen, R. Taibi, T. Signamarcheix, L. Bally, L. Vandroux, M. Zussy, S. Verrun, J. Dechamp, P. Leduc, M. Assous, D. Bouchu, F. de Crecy, L. L. Chapelon, and L. Clavelier, “An innovative die to wafer 3D integration scheme: Die to wafer oxide or copper direct bonding with planarised oxide inter-die filling,” in *2009 IEEE International Conference on 3D System Integration*, pp. 1–4, Sept 2009.

[45] M. Ohyama, M. Nimura, J. Mizuno, S. Shoji, M. Tamura, T. Enomoto, and A. Shigetou, “Hybrid bonding of Cu/Sn microbump and adhesive with silica filler for 3D interconnection of single micron pitch,” in *IEEE 65th Electronic Components and Technology Conference (ECTC)*, pp. 325–330, May 2015.

[46] V. Balan, O. Oluwole, G. Kodani, C. Zhong, R. Dadi, A. Amin, A. Ragab, and M. J. E. Lee, “A 15-22 Gbps Serial Link in 28 nm CMOS With Direct DFE,” *IEEE Journal of Solid-State Circuits*, vol. 49, pp. 3104–3115, Dec 2014.

[47] K. Kaviani, T. Wu, J. Wei, A. Amirkhany, J. Shen, T. J. Chin, C. Thakkar, W. T. Beyene, N. Chan, C. Chen, B. R. Chuang, D. Dressler, V. P. Gadde, M. Hekmat, E. Ho, C. Huang, P. Le, Mahabaleshwara, C. Madden, N. K. Mishra, L. Raghavan, K. Saito, R. Schmitt, D. Secker, X. Shi, S. Fazeel, G. S. Srinivas, S. Zhang, C. Tran, A. Vaidyanath, K. Vyas, M. Jain, K. Y. K. Chang, and X. Yuan, “A Tri-Modal 20-Gbps/Link Differential/DDR3/GDDR5 Memory Interface,” *IEEE Journal of Solid-State Circuits*, vol. 47, pp. 926–937, April 2012.

[48] M. A. Karim, P. D. Franzon, and A. Kumar, “Power comparison of 2D, 3D and 2.5D interconnect solutions and power optimization of interposer interconnects,” in *2013 IEEE 63rd Electronic Components and Technology Conference*, pp. 860–866, May 2013.

[49] AMD, “AMD Radeon R9,” (2015).

[50] NVIDIA, “NVIDIA Updates GPU Roadmap; Announces Pascal,” (2015).

[51] T. G. Lenihan, L. Matthew, and E. J. Vardaman, “Developments in 2.5D: The role of silicon interposers,” in *2013 IEEE 15th Electronics Packaging Technology Conference (EPTC 2013)*, pp. 53–55, Dec 2013.

[52] D. Malta, E. Vick, S. Goodwin, C. Gregory, M. Lueck, A. Huffman, and D. Temple, “Fabrication of TSV-based silicon interposers,” in *2010 IEEE International 3D Systems Integration Conference (3DIC)*, pp. 1–6, Nov 2010.

[53] J. Keech, S. Chaparala, A. Shorey, G. Piech, and S. Pollard, “Fabrication of 3D-IC interposers,” in *2013 IEEE 63rd Electronic Components and Technology Conference*, pp. 1829–1833, May 2013.

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

[55] L. Li, P. Chia, P. Ton, M. Nagar, S. Patil, J. Xue, J. Delacruz, M. Voicu, J. Hellings, B. Isaacson, M. Coor, and R. Havens, "3D SiP with Organic Interposer for ASIC and Memory Integration," in *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, pp. 1445–1450, May 2016.

[56] "System Level Co-Optimizations of 2.5D/3D Hybrid Integration for High Performance Computing System." `http://www.semiconwest.org/sites/semiconwest.org/files/data15/docs/3_John%20Hu_nVIDIA.pdf`, (accessed August 1, 2017).

[57] V. Kontorinis, A. Shayan, D. M. Tullsen, and R. Kumar, "Reducing peak power with a table-driven adaptive processor core," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, (New York, NY, USA), pp. 189–200, ACM, 2009.

[58] A. Syed, "Factors affecting electromigration and current carrying capacity of FC and 3D IC interconnects," in *2010 12th Electronics Packaging Technology Conference*, pp. 538–544, Dec 2010.

[59] K. N. Tu, H. G. Xu Gu, and W. J. Choi, *Electromigration in Solder Joints and Lines.* 2011.

[60] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 469–480, IEEE, 2009.

[61] "HPC-oriented Latency Numbers Every Programmer Should Know." `https://goo.gl/ftzz3a`, (accessed July 29, 2017).

[62] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 52:1–52:12, Nov. 2011.

[63] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, *et al.*, "The nas parallel benchmarks," *The International Journal of Supercomputing Applications*, vol. 5, no. 3, pp. 63–73, 1991.

[64] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, PACT '08, (New York, NY, USA), pp. 72–81, ACM, 2008.

[65] J. T. Pawlowski, "Hybrid memory cube (hmc)," in *2011 IEEE Hot Chips 23 Symposium (HCS)*, pp. 1–24, Aug 2011.

[66] M. N. Bojnordi and E. Ipek, "PARDIS: A Programmable Memory Controller for the DDRx Interfacing Standards," in *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, (Washington, DC, USA), pp. 13–24, IEEE Computer Society, 2012.

[67] "Hybrid memory cube." `https://en.wikipedia.org/wiki/Hybrid_Memory_Cube`, (accessed July 31, 2017).

[68] "SK hynix 21 nm DRAM Cell Technology." `http://techinsights.com/about-techinsights/overview/blog/sk-hynix-21-nm-dram-cell-technology-comparison-of-1st-and-2nd-generation/`, Accessed on July 30, 2017.

[69] "Intel Xeon Server Board - Dual Socket." `https://www.supermicro.com/products/motherboard/Xeon/C600/X10DAX.cfm`.

[70] B. Vasquez and S. Lindsey, "The Promise of Known-good-die Technologies," in *International Conference on Multichip Modules*, pp. 1–6, Apr 1994.

[71] R. Arnold, S. M. Menon, B. Brackett, and R. Richmond, "Test methods used to produce highly reliable known good die (KGD)," in *Proceedings. 1998 International Conference on Multichip Modules and High Density Packaging (Cat. No.98EX154)*, pp. 374–382, Apr 1998.

[72] R. H. Parker, "Bare die test," in *Proceedings 1992 IEEE Multi-Chip Module Conference MCMC-92*, pp. 24–27, Mar 1992.

[73] D. Chu, C. A. Reber, and D. W. Palmer, "Screening ICs on the bare chip level: temporary packaging," *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, vol. 16, pp. 392–395, Jun 1993.

[74] W. Ballouli, T. McKenzie, and N. Alizy, "Known good die achieved through wafer level burn-in and test," in *IEEE/CPMT International Electronics Manufacturing Technology Symposium*, pp. 153–159, 2000.

[75] D. R. Conti and J. V. Horn, "Wafer level burn-in," in *2000 Proceedings. 50th Electronic Components and Technology Conference*, pp. 815–821, 2000.

[76] `https://www.advantest.com/products/leading-edge-products/ha1000`.

[77] H. H. Chen, "Hierarchical built-in self-test for system-on-chip design," in *Conference, Emerging Information Technology 2005.*, p. 3, Aug 2005.

[78] C. Grecu, P. Pande, A. Ivanov, and R. Saleh, "BIST for network-on-chip interconnect infrastructures," in *24th IEEE VLSI Test Symposium*, pp. 6 pp.–35, April 2006.

[79] J. B. Brinton and J. R. Lineback, *Packaging is becoming biggest cost in assembly, passing capital equipment.* EE Times [Online], 1999.

[80] R. H. Katz, *Cost, Price, and Price for Performance*. UC Berkeley, 1996.

[81] C. A. Palesko and E. J. Vardaman, "Cost comparison for flip chip, gold wire bond, and copper wire bond packaging," in *2010 Proceedings 60th Electronic Components and Technology Conference (ECTC)*, pp. 10–13, June 2010.

[82] "Determining Total Cost of Ownership for Data Center and Network Room Infrastructure ." `http://www.apc.com/salestools/CMRP-5T9PQG/CMRP-5T9PQG_R4_EN.pdf`.

[83] A. E. Papathanasiou and M. L. Scott, "Aggressive prefetching: An idea whose time has come," in *Conference on Hot Topics in Operating Systems - Volume 10*, HOTOS, (Berkeley, CA, USA), pp. 6–6, 2005.

[84] S. Garg, D. Marculescu, R. Marculescu, and U. Ogras, "Technology-driven limits on dvfs controllability of multiple voltage-frequency island designs: A system-level perspective," in *2009 46th ACM/IEEE Design Automation Conference*, pp. 818–821, July 2009.

[85] "Heterogeneous integration roadmap 2019 edition." https://eps.ieee.org/technology/heterogeneous- integration- roadmap/2019- edition.html, Aug. 2019.

[86] N. Beck, S. White, M. Paraschou, and S. Naffziger, "'zeppelin': An soc for multichip architectures," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pp. 40–42, 2018.

[87] L. Shan, Y. Kwark, C. Baks, M. Gaynes, T. Chainer, M. Kapfhammer, H. Saiki, A. Kuhara, G. Aguiar, N. Ban, and Y. Nukaya, "Organic multi-chip module for high performance systems," in *2015 IEEE 65th Electronic Components and Technology Conference (ECTC)*, pp. 1725–1729, 2015.

[88] S. Y. Hou, W. C. Chen, C. Hu, C. Chiu, K. C. Ting, T. S. Lin, W. H. Wei, W. C. Chiou, V. J. C. Lin, V. C. Y. Chang, C. T. Wang, C. H. Wu, and D. Yu, "Wafer-level integration of an advanced logic-memory system through the second-generation cowos technology," *IEEE Transactions on Electron Devices*, vol. 64, no. 10, pp. 4071–4077, 2017.

[89] C. C. Liu, S. Chen, F. Kuo, H. Chen, E. Yeh, C. Hsieh, L. Huang, M. Chiu, J. Yeh, T. Lin, T. Yeh, S. Hou, J. Hung, J. Lin, C. Jou, C. Wang, S. Jeng, and D. C. H. Yu, "High-performance integrated fan-out wafer level packaging (info-wlp): Technology and system integration," in *2012 International Electron Devices Meeting*, pp. 14.1.1–14.1.4, 2012.

[90] "2.5d interposer(i-cube™) development." samsungfoundry.com, Oct 2020.

[91] J. Lee and M. Kelly, "Amkor's 2.5d package and hdfo – advanced heterogeneous packaging solutions." China Integrated Circuits, December 2018.

[92] A. A. Bajwa, S. Jangam, S. Pal, N. Marathe, T. Bai, T. Fukushima, M. Goorsky, and S. S. Iyer, "Heterogeneous integration at fine pitch (10 µm) using thermal compression bonding," in *2017 IEEE 67th Electronic Components and Technology Conference (ECTC)*, pp. 1276–1284, 2017.

[93] A. A. Bajwa, S. Jangam, S. Pal, B. Vaisband, R. Irwin, M. Goorsky, and S. S. Iyer, "Demonstration of a heterogeneously integrated system-on-wafer (sow) assembly," in *2018 IEEE 68th Electronic Components and Technology Conference (ECTC)*, pp. 1926–1930, 2018.

[94] N. Kim, D. Wu, D. Kim, A. Rahman, and P. Wu, "Interposer design optimization for high frequency signal transmission in passive and active interposer using through silicon via (tsv)," in *2011 IEEE 61st Electronic Components and Technology Conference (ECTC)*, pp. 1160–1167, 2011.

[95] N. Pantano, C. R. Neve, G. Van der Plas, M. Detalle, M. Verhelst, M. Heyns, and E. Beyne, "Technology optimization for high bandwidth density applications on 3d interposer," in *2016 6th Electronic System-Integration Technology Conference (ESTC)*, pp. 1–6, 2016.

[96] "Hspice." online, 2020.

[97] D. R. Stauffer, J. T. Mechler, M. A. Sorna, K. Dramstad, C. R. Ogilvie, A. Mohammad, and J. D. Rockrohr, *High speed serdes devices and applications*. Springer Science & Business Media, 2008.

[98] Shyh-Chyi Wong, Gwo-Yann Lee, and Dye-Jyun Ma, "Modeling of interconnect capacitance, delay, and crosstalk in vlsi," *IEEE Transactions on Semiconductor Manufacturing*, vol. 13, no. 1, pp. 108–111, 2000.

[99] S. Jangam, A. A. Bajwa, K. K. Thankkappan, P. Kittur, and S. S. Iyer, "Electrical characterization of high performance fine pitch interconnects in silicon-interconnect fabric," in *2018 IEEE 68th Electronic Components and Technology Conference (ECTC)*, pp. 1283–1288, 2018.

[100] S. Ramalingam, "Hbm package integration: Technology trends, challenges and applications," in *2016 IEEE Hot Chips 28 Symposium (HCS)*, pp. 1–17, 2016.

[101] P. Ehrett, V. Goyal, O. Matthews, R. Das, T. Austin, and V. Bertacco, "Analysis of microbump overheads for 2.5 d disintegrated design,"

[102] A. Amerasekera, W. van den Abeelen, L. van Roozendaal, M. Hannemann, and P. Schofield, "Esd failure modes: characteristics mechanisms, and process influences," *IEEE Transactions on Electron Devices*, vol. 39, no. 2, pp. 430–436, 1992.

[103] Y. Sa, S. Yoo, Y. Shin, M. Han, and C. Lee, "Joint properties of solder capped copper pillars for 3d packaging," in *2010 Proceedings 60th Electronic Components and Technology Conference (ECTC)*, pp. 2019–2024, 2010.

[104] W. J. Turner, J. W. Poulton, J. M. Wilson, X. Chen, S. G. Tell, M. Fojtik, T. H. Greer, B. Zimmer, S. Song, N. Nedovic, S. S. Kudva, S. R. Sudhakaran, R. Bashirullah, W. Zhao, W. J. Dally, and C. T. Gray, "Ground-referenced signaling for intra-chip and short-reach chip-to-chip interconnects," in *2018 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–8, 2018.

[105] E. Association *et al.*, "Esd association standard for the development of an electrostatic discharge control program for–protection of electrical and electronic parts, assemblies and equipment (excluding electrically initiated explosive devices). ansi/esd s20: 20-2007," 2007.

[106] W.-S. Lei, A. Kumar, and R. Yalamanchili, "Die singulation technologies for advanced packaging: A critical review," *Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena*, vol. 30, no. 4, p. 040801, 2012.

[107] S.-p. Jeng, H.-W. Chen, S.-Y. Hou, H.-Y. Tsai, A. N. Wu, and Y.-W. Liu, "Protective seal ring for preventing die-saw induced stress," Dec. 18 2012. US Patent 8,334,582.

[108] T. Lazerand and D. Lishan, "Wafer dicing using dry etching on standard tapes and frames," 10 2014.

[109] M. O'Connor, N. Chatterjee, D. Lee, J. Wilson, A. Agrawal, S. W. Keckler, and W. J. Dally, "Fine-grained dram: Energy-efficient dram for extreme bandwidth systems," in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 41–54, 2017.

[110] N. Platform, "Building bigger, faster gpu clusters using nvswitches," 2018.

[111] DARPA, "Common heterogeneous integration and ip reuse strategies (chips)," 2020.

[112] O. C. Project, "Ocp open domain-specific architecture sub-project," 2020.

[113] Intel, "Accelerating innovation through a standard chiplet interface: The advanced interface bus (aib)."

[114] "Intel® xeon® processors."

[115] "Intel® xeon® e7 processors."

[116] "Intel® xeon® e5 processors."

[117] "Intel® xeon® e3 processors."

[118] "Mckinsey on semiconductors, 2014."

[119] "Semiwiki - all things semiconductor."

[120] S. Y. Hou, W. C. Chen, C. Hu, C. Chiu, K. C. Ting, T. S. Lin, W. H. Wei, W. C. Chiou, V. J. C. Lin, V. C. Y. Chang, C. T. Wang, C. H. Wu, and D. Yu, "Wafer-level integration of an advanced logic-memory system through the second-generation cowos technology," *IEEE Transactions on Electron Devices*, vol. 64, pp. 4071–4077, Oct 2017.

[121] M. M. Kim, M. Mehrara, M. Oskin, and T. Austin, "Architectural implications of brick and mortar silicon manufacturing," in *ACM SIGARCH Computer Architecture News*, vol. 35, pp. 244–253, ACM, 2007.

[122] T. Vijayaraghavan, Y. Eckert, G. H. Loh, M. J. Schulte, M. Ignatowski, B. M. Beckmann, W. C. Brantley, J. L. Greathouse, W. Huang, A. Karunanithi, *et al.*, "Design and analysis of an apu for exascale computing,"

[123] A. B. S. P. P. G. S. Jangam, S. Pal and S. S. Iyer, "Latency, bandwidth and power benefits of the superchips integration scheme," in *IEEE Electronic Components and Technology Conference (ECTC), to appear in May 2017*.

[124] D. B. Papworth, "Tuning the pentium pro microarchitecture," *IEEE Micro*, vol. 16, pp. 8–15, Apr 1996.

[125] J. Kin, C. Lee, W. H. Mangione-Smith, and M. Potkonjak, "Power efficient mediaprocessors: Design space exploration," in *Proceedings of the 36th Annual ACM/IEEE Design Automation Conference*, DAC '99, (New York, NY, USA), pp. 321–326, ACM, 1999.

[126] V. Desmet, S. Girbal, and O. Temam, "Archexplorer.org: Joint compiler/hardware exploration for fair comparison of architectures," 2009.

[127] C. Silvano, W. Fornaciari, G. Palermo, V. Zaccaria, F. Castro, M. Martinez, S. Bocchio, R. Zafalon, P. Avasare, G. Vanmeerbeeck, C. Ykman-Couvreur, M. Wouters, C. Kavka, L. Onesti, A. Turco, U. Bondik, G. Mariani, H. Posadas, E. Villar, C. Wu, F. Dongrui, Z. Hao, and T. Shibin, "Multicube: Multi-objective design space exploration of multi-core architectures," in *2010 IEEE Computer Society Annual Symposium on VLSI*, pp. 488–493, July 2010.

[128] S. Kang and R. Kumar, "Magellan: A search and machine learning-based framework for fast multi-core design space exploration and optimization," in *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '08, (New York, NY, USA), pp. 1432–1437, ACM, 2008.

[129] H. Calborean and L. Vinţan, "An automatic design space exploration framework for multicore architecture optimizations," in *9th RoEduNet IEEE International Conference*, pp. 202–207, June 2010.

[130] R. Kumar, D. M. Tullsen, and N. P. Jouppi, "Core architecture optimization for heterogeneous chip multiprocessors," in *Proceedings of the 15th international conference on Parallel architectures and compilation techniques*, pp. 23–32, ACM, 2006.

[131] N. K. Choudhary, S. V. Wadhavkar, T. A. Shah, H. Mayukh, J. Gandhi, B. H. Dwiel, S. Navada, H. H. Najaf-abadi, and E. Rotenberg, "Fabscalar: Composing synthesizable rtl designs of arbitrary cores within a canonical superscalar template," in *ACM SIGARCH Computer Architecture News*, vol. 39, pp. 11–22, ACM, 2011.

[132] Y. Li, B. C. Lee, D. M. Brooks, Z. Hu, and K. Skadron, "Cmp design space exploration subject to physical constraints," in *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.*, 2006.

[133] F. Remond and P. Bricaud, "Set top box soc design methodology at stmicroelectronics," in *Design, Automation and Test in Europe Conference and Exhibition, 2003*, pp. 220–223 suppl., 2003.

[134] R. Saleh, S. Wilton, G. Lemieux, P. P. Pande, and et al., "System-on-chip: Reuse and integration," 2006.

[135] M. Keating and P. Bricaud, "System-on-chip: Reuse and integration," *Proceedings of IEEE*, Jan 2002.

[136] G. Martin, R. Seepold, T. Zhang, L. Benini, and G. De Micheli, "Component selection and matching for ip-based design," in *Proceedings of the conference on Design, automation and test in Europe*, pp. 40–46, IEEE Press, 2001.

[137] T. G. Lenihan, L. Matthew, and E. J. Vardaman, "Developments in 2.5d: The role of silicon interposers," in *2013 IEEE 15th Electronics Packaging Technology Conference (EPTC 2013)*, pp. 53–55, Dec 2013.

[138] C. F. Tseng, C. S. Liu, C. H. Wu, and D. Yu, "Info (wafer level integrated fan-out) technology," in *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, pp. 1–6, May 2016.

[139] M. J. Schulte, M. Ignatowski, G. H. Loh, B. M. Beckmann, W. C. Brantley, S. Gurumurthi, N. Jayasena, I. Paul, S. K. Reinhardt, and G. Rodgers, "Achieving exascale capabilities through heterogeneous computing," *IEEE Micro*, vol. 35, pp. 26–36, July 2015.

[140] A. Kannan, N. E. Jerger, and G. H. Loh, "Enabling interposer-based disintegration of multi-core processors," in *Proceedings of the 48th International Symposium on Microarchitecture*, MICRO-48, (New York, NY, USA), pp. 546–558, ACM, 2015.

[141] "Mochi architecture," `http://www.marvell.com/architecture/mochi/`.

[142] "Standard performance evaluation corporation."

[143] "Eembc - embedded microprocessor benchmarks."

[144] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The splash-2 programs: Characterization and methodological considerations," in *Computer Architecture, 1995. Proceedings., 22nd Annual International Symposium on*, pp. 24–36, IEEE, 1995.

[145] S. Kang and R. Kumar, *Magellan: A search and machine learning-based framework for fast multi-core design space exploration and optimization*, pp. 1432–1437. 2008.

[146]

[147] I. Tirkel, "Yield learning curve models in semiconductor manufacturing," *IEEE Trans. on Semicond. Manuf.*, vol. 26, pp. 564–571, Nov 2013.

[148] A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, (New York, NY, USA), pp. 169–178, ACM, 2000.

[149] A. Phansalkar, A. Joshi, and L. K. John, "Analysis of redundancy and application balance in the spec cpu2006 benchmark suite," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 412–423, 2007.

[150] H. Patil and T. E. Carlson, “Pinballs: Portable and shareable user-level checkpoints for reproducible analysis and simulation,” in *Proceedings of the Workshop on Reproducible Research Methodologies (REPRODUCE)*, 2014.

[151] M. E. Thomadakis, “The architecture of the nehalem processor and nehalem-ep smp platforms,” *Resource*, vol. 3, p. 2, 2011.

[152] E. Sperling.

[153] eSilicon, “How to improve the profitability of fabless semiconductor companies.”

[154] O. N. Corporation, “Wafer price is hiking up.”

[155] D. Stow, I. Akgun, R. Barnes, P. Gu, and Y. Xie, “Cost analysis and cost-driven ip reuse methodology for soc design based on 2.5d/3d integration,” in *Proc. ICCAD*, pp. 1–6, Nov 2016.

[156] ITRS, “Yield enhancement,” 2007 Edition.

[157] “Gartner says worldwide sales of smartphones returned to growth in first quarter of 2018.” `http://http://www.gartner.com/newsroom/id/3876865`.

[158] “Quarterly personal computer (pc) vendor shipments worldwide, from 2009 to 2018, by vendor (in million units).” `https://www.statista.com/statistics/263393/global-pc-shipments-since-1st-quarter-2009-by-vendor/`.

[159] “Gartner newsroom, 3530117 and 3560517.”

[160] “Trilogy Systems.” `https://en.wikipedia.org/wiki/Trilogy_Systems`, (accessed Nov 20, 2017).

[161] R. M. Lea, “WASP: a wafer-scale massively parallel processor,” in *1990 Proceedings. International Conference on Wafer Scale Integration*, pp. 36–42, Jan 1990.

[162] D. Landis and J. Yoder and D. Whittaker and T. Dobbins, “A wafer scale programmable systolic data processor,” in *Proceedings Ninth Biennial University/Government/Industry Microelectronics Symposium*, pp. 252–256, Jun 1991.

[163] S. Pal, D. Petrisko, A. A. Bajwa, P. Gupta, S. S. Iyer, and R. Kumar, “A case for packageless processors,” in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 466–479, Feb 2018.

[164] M. Panahiazar and V. Taslimitehrani and A. Jadhav and J. Pathak, “Empowering personalized medicine with big data and semantic web technology: Promises, challenges, and use cases,” in *2014 IEEE International Conference on Big Data (Big Data)*, pp. 790–795, Oct 2014.

[165] “NVIDIA Powers Virtual Reality.” `http://www.nvidia.com/object/virtual-reality.html`, (accessed Nov 21, 2017).

[166] "NVIDIA Tesla Imaging and Computer Vision." http://www.nvidia.com/object/imaging_comp_vision.html, (accessed Nov 21, 2017).

[167] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR*, vol. abs/1409.1556, 2014.

[168] "Nvidia says its new supercomputer will enable the highest level of automated driving." https://www.theverge.com/2017/10/10/16449416/nvidia-pegasus-self-driving-car-ai-robotaxi, (accessed Nov 21, 2017).

[169] "An Introduction to High Performance Computing on AWS (White paper)." https://d0.awsstatic.com/whitepapers/Intro_to_HPC_on_AWS.pdf, (accessed Nov 21, 2017).

[170] "Microsoft Azure," (accessed Nov 21, 2017).

[171] "TOP500 Shows Growing Momentum for Accelerators." https://insidehpc.com/2015/11/top500-shows-growing-momentum-for-accelerators/, (accessed Nov 21, 2017).

[172] "HPC Application Support for GPU Computing." https://insidehpc.com/2015/11/top500-shows-growing-momentum-for-accelerators/, (accessed Nov 21, 2017).

[173] T. Vijayaraghavan and Y. Eckert and G. H. Loh and M. J. Schulte and M. Ignatowski and B. M. Beckmann and W. C. Brantley and J. L. Greathouse and W. Huang and A. Karunanithi and O. Kayiran and M. Meswani and I. Paul and M. Poremba and S. Raasch and S. K. Reinhardt and G. Sadowski and V. Sridharan, "Design and Analysis of an APU for Exascale Computing," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 85–96, Feb 2017.

[174] "Sandy Bridge (client) - Microarchitectures - Intel (Wikichip)." https://en.wikichip.org/wiki/intel/microarchitectures/sandy_bridge_(client), (accessed Nov 20, 2017).

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

[176] J. W. Poulton and W. J. Dally and X. Chen and J. G. Eyles and T. H. Greer and S. G. Tell and J. M. Wilson and C. T. Gray, "A 0.54 pJ/b 20 Gb/s Ground-Referenced Single-Ended Short-Reach Serial Link in 28 nm CMOS for Advanced Packaging Applications," *IEEE Journal of Solid-State Circuits*, vol. 48, pp. 3206–3218, Dec 2013.

[177] Abts, Dennis and Marty, Michael R. and Wells, Philip M. and Klausler, Peter and Liu, Hong, "Energy Proportional Datacenter Networks," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ISCA '10, (New York, NY, USA), pp. 338–347, ACM, 2010.

[178] Y. Huang and Y. Yesha and M. Halem and Y. Yesha and S. Zhou, "YinMem: A distributed parallel indexed in-memory computation system for large scale data analytics," in *2016 IEEE International Conference on Big Data (Big Data)*, pp. 214–222, Dec 2016.

[179] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A scalable processing-in-memory accelerator for parallel graph processing," in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, pp. 105–117, June 2015.

[180] E. Azarkhish and C. Pfister and D. Rossi and I. Loi and L. Benini, "Logic-Base Interconnect Design for Near Memory Computing in the Smart Memory Cube," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, pp. 210–223, Jan 2017.

[181] K. Murthy and J. Mellor-Crummey, "Communication avoiding algorithms: Analysis and code generation for parallel systems," in *2015 International Conference on Parallel Architecture and Compilation (PACT)*, pp. 150–162, Oct 2015.

[182] V. Kumar and A. Naeemi, "An overview of 3d integrated circuits," in *2017 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization for RF, Microwave, and Terahertz Applications (NEMO)*, pp. 311–313, May 2017.

[183] L. D. Cioccio, P. Gueguen, R. Taibi, T. Signamarcheix, L. Bally, L. Vandroux, M. Zussy, S. Verrun, J. Dechamp, P. Leduc, M. Assous, D. Bouchu, F. de Crecy, L. Chapelon, and L. Clavelier, "An innovative die to wafer 3d integration scheme: Die to wafer oxide or copper direct bonding with planarised oxide inter-die filling," in *2009 IEEE International Conference on 3D System Integration*, pp. 1–4, Sept 2009.

[184] A. Sigl, S. Pargfrieder, C. Pichler, C. Scheiring, and P. Kettner, "Advanced chip to wafer bonding: A flip chip to wafer bonding technology for high volume 3dic production providing lowest cost of ownership," in *2009 International Conference on Electronic Packaging Technology High Density Packaging*, pp. 932–936, Aug 2009.

[185] S. Che, B. M. Beckmann, S. K. Reinhardt, and K. Skadron, "Pannotia: Understanding irregular gpgpu graph applications," in *2013 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 185–195, Sept 2013.

[186] Arunkumar, Akhil and Bolotin, Evgeny and Cho, Benjamin and Milic, Ugljesa and Ebrahimi, Eiman and Villa, Oreste and Jaleel, Aamer and Wu, Carole-Jean and Nellans, David, "MCM-GPU: Multi-Chip-Module GPUs for Continued Performance Scalability," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, (New York, NY, USA), pp. 320–332, ACM, 2017.

[187] Y. L. Chuang and C. S. Yuan and J. J. Chen and C. F. Chen and C. S. Yang and W. P. Changchien and C. C. C. Liu and F. Lee, "Unified methodology for heterogeneous integration with CoWoS technology," in *2013 IEEE 63rd Electronic Components and Technology Conference*, pp. 852–859, May 2013.

[188] C. L. Lai and H. Y. Li and A. Chen and T. Lu, "Silicon Interposer Warpage Study for 2.5D IC without TSV Utilizing Glass Carrier CTE and Passivation Thickness Tuning," in *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, pp. 310–315, May 2016.

[189] Ying-Wen Chou and Po-Yuan Chen and Mincent Lee and C. W. Wu, "Cost modeling and analysis for interposer-based three-dimensional IC," in *2012 IEEE 30th VLSI Test Symposium (VTS)*, pp. 108–113, April 2012.

[190] John Hu, "System level co-cptimizations of 2.5D/3D hybrid integration for high performance computing system," in *Semicon West 2016*, 2016.

[191] H. D. Thacker and M. S. Bakir and D. C. Keezer and K. P. Martin and J. D. Meindl, "Compliant probe substrates for testing high pin-count chip scale packages," in *52nd Electronic Components and Technology Conference 2002. (Cat. No.02CH37345)*, pp. 1188–1193, 2002.

[192] E. Wachter, A. Erichsen, A. Amory, and F. Moraes, "Topology-agnostic fault-tolerant noc routing method," in *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '13, (San Jose, CA, USA), pp. 1595–1600, EDA Consortium, 2013.

[193] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw, "A highly resilient routing algorithm for fault-tolerant nocs," in *2009 Design, Automation Test in Europe Conference Exhibition*, pp. 21–26, April 2009.

[194] J. Wang and S. Yalamanchili, "Characterization and analysis of dynamic parallelism in unstructured gpu applications," in *2014 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 51–60, Oct 2014.

[195] J. Sim, A. Dasgupta, H. Kim, and R. Vuduc, "A performance analysis framework for identifying potential benefits in gpgpu applications," in *Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '12, (New York, NY, USA), pp. 11–22, ACM, 2012.

[196] "Blue Waters." `http://www.ncsa.illinois.edu/enabling/bluewaters`, (accessed July 1, 2018).

[197] "GPU Supercomputing in Blue Waters." `http://developer.download.nvidia.com/compute/academia/whitepapers/Achievement2013_UIUC.pdf`, (accessed July 1, 2018).

[198] "Workload Analysis of Blue Waters." `https://arxiv.org/ftp/arxiv/papers/1703/1703.00924.pdf`, (accessed July 1, 2018).

[199] M. Viñas, Z. Bozkus, and B. B. Fraguela, "Exploiting heterogeneous parallelism with the heterogeneous programming library," *J. Parallel Distrib. Comput.*, vol. 73, pp. 1627–1638, Dec. 2013.

[200] Che, Shuai and Boyer, Michael and Meng, Jiayuan and Tarjan, David and Sheaffer, Jeremy W and Lee, Sang-Ha and Skadron, Kevin, "Rodinia: A benchmark suite for heterogeneous computing," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, pp. 44–54, Ieee, 2009.

[201] J. Power, J. Hestness, M. S. Orr, M. D. Hill, and D. A. Wood, "gem5-gpu: A heterogeneous cpu-gpu simulator," *IEEE Computer Architecture Letters*, vol. 14, pp. 34–36, Jan 2015.

[202] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "Hotspot: a compact thermal modeling methodology for early-stage vlsi design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 501–513, May 2006.

[203] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "Raidr: Retention-aware intelligent dram refresh," in *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, (Washington, DC, USA), pp. 1–12, IEEE Computer Society, 2012.

[204] "NVIDIA GPU maximum operating temperature and overheating." `https://http://nvidia.custhelp.com/app/answers/detail/a_id/2752/~/nvidia-gpu-maximum-operating-temperature-and-overheating`, (accessed Nov 21, 2017).

[205] M. Ahmed and C. Fei and F. C. Lee and Q. Li, "High-efficiency high-power-density 48/1V sigma converter voltage regulator module," in *2017 IEEE Applied Power Electronics Conference and Exposition (APEC)*, pp. 2207–2212, March 2017.

[206] F. C. Lee and Q. Li and Z. Liu and Y. Yang and C. Fei and M. Mu, "Application of GaN devices for 1 kW server power supply with integrated magnetics," *CPSS Transactions on Power Electronics and Applications*, vol. 1, pp. 3–12, Dec 2016.

[207] "Datacenter Power Delivery Architectures:Efficiency and Annual Operating Costs ." `http://www.vicorpower.com/documents/whitepapers/server_efficiency_vichip.pdf.`

[208] P. Gupta and A. B. Kahng, "Efficient design and analysis of robust power distribution meshes," in *19th International Conference on VLSI Design (VLSID'06)*, pp. 6 pp.–, Jan 2006.

[209] M. H. Ahmed and C. Fei and F. C. Lee and Q. Li, "48-V Voltage Regulator Module With PCB Winding Matrix Transformer for Future Data Centers," *IEEE Transactions on Industrial Electronics*, vol. 64, pp. 9302–9310, Dec 2017.

[210] J. Leng and Y. Zu and V. J. Reddi, "GPU voltage noise: Characterization and hierarchical smoothing of spatial and temporal voltage noise interference in GPU architectures," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pp. 161–173, Feb 2015.

[211] S. K. Lee, T. Tong, X. Zhang, D. Brooks, and G. Y. Wei, "A 16-core voltage-stacked system with adaptive clocking and an integrated switched-capacitor dc x2013;dc converter," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, pp. 1271–1284, April 2017.

[212] K. Mazumdar and M. Stan, "Breaking the power delivery wall using voltage stacking," in *Proceedings of the Great Lakes Symposium on VLSI*, GLSVLSI '12, (New York, NY, USA), pp. 51–54, ACM, 2012.

[213] Q. Zhang and L. Lai and M. Gottscho and P. Gupta, "Multi-story power distribution networks for GPUs," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 451–456, March 2016.

[214] E. Alon and M. Horowitz, "Integrated regulation for energy-efficient digital circuits," vol. 43, pp. 1795 – 1807, 09 2008.

[215] E. Papadopoulou and D. T. Lee, "Critical area computation via Voronoi diagrams," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, pp. 463–474, Apr 1999.

[216] N. Chatterjee and M. O'Connor and D. Lee and D. R. Johnson and S. W. Keckler and M. Rhu and W. J. Dally, "Architecting an Energy-Efficient DRAM System for GPUs," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 73–84, Feb 2017.

[217] "JEDEC Publishes HBM2 Specification as Samsung Begins Mass Production of Chips." `https://www.anandtech.com/show/9969/jedec-publishes-hbm2-specification`.

[218] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Design and implementation of the fiduccia-mattheyses heuristic for vlsi netlist partitioning," in *Selected Papers from the International Workshop on Algorithm Engineering and Experimentation*, ALENEX '99, (London, UK, UK), pp. 177–193, Springer-Verlag, 1999.

[219] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, pp. 1–7, Aug. 2011.

[220] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing cuda workloads using a detailed gpu simulator," in *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, pp. 163–174, April 2009.

[221] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, p. 65–76, Apr. 2009.

[222] U. Milic, O. Villa, E. Bolotin, A. Arunkumar, E. Ebrahimi, A. Jaleel, A. Ramirez, and D. Nellans, "Beyond the socket: Numa-aware gpus," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, (New York, NY, USA), pp. 123–135, ACM, 2017.

[223] N. Vijaykumar, E. Ebrahimi, K. Hsieh, P. B. Gibbons, and O. Mutlu, "The locality descriptor: A holistic cross-layer abstraction to express data locality in gpus," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 829–842, June 2018.

[224] L. Wang, Y. Wang, H. Zhang, Z. Zhong, L. Wang, D. Peng, and F. Bai, "Integrated On-Chip Solenoid Inductors With Nanogranular Magnetic Cores," *IEEE Transactions on Magnetics*, vol. 52, pp. 1–4, July 2016.

[225] M. Skach, M. Arora, C. H. Hsu, Q. Li, D. Tullsen, L. Tang, and J. Mars, "Thermal time shifting: Leveraging phase change materials to reduce cooling costs in warehouse-scale computers," in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, pp. 439–449, June 2015.

[226] K. Shirahata, H. Sato, T. Suzumura, and S. Matsuoka, "A scalable implementation of a mapreduce-based graph processing algorithm for large-scale heterogeneous supercomputers," in *13th International Symposium on Cluster, Cloud, and Grid Computing*, pp. 277–284, 2013.

[227] K. Rocki, D. V. Essendelft, I. Sharapov, R. Schreiber, M. Morrison, V. Kibardin, A. Portnoy, J. F. Dietiker, M. Syamlal, and M. James, "Fast stencil-code computation on a wafer-scale processor," 2020.

[228] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *International Symposium on Circuits and Systems*, pp. 1947–1950, 2010.

[229] A. A. Bajwa, S. Jangam, S. Pal, B. Vaisband, R. Irwin, M. Goorsky, and S. S. Iyer, "Demonstration of a heterogeneously integrated system-on-wafer (sow) assembly," in *68th ECTC*, pp. 1926–1930, 2018.

[230] S. Ramalingam, "3d-ics: Advances in the industry," Accessed Nov 23, 2020.

[231] NVIDIA, "A100 tensor core gpu," Accessed Nov 22, 2020.

[232] AMD, "Ryzen™ threadripper™ processors," Accessed Nov 23, 2020.

[233] Y. S. Shao, J. Clemons, R. Venkatesan, B. Zimmer, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S. G. Tell, Y. Zhang, W. J. Dally, J. Emer, C. T. Gray, B. Khailany, and S. 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*, (New York, NY, USA), p. 14–27, 2019.

[234] S. Xie and M. B. Taylor, "The basejump manycore accelerator network," 2018.

[235] M. Liu, B. Vaisband, A. Hanna, Y. Luo, Z. Wan, and S. S. Iyer, "Process development of power delivery through wafer vias for silicon interconnect fabric," in *69th ECTC*, pp. 579–586, 2019.

[236] K. T. Kannan and S. S. Iyer, "Deep trench capacitors in silicon interconnect fabric," in *IEEE 70th Electronic Components and Technology Conference*, pp. 2295–2301, 2020.

[237] I. Kaijian Shi, Synopsys, "Clock distribution and balancing methodology for large and complex asic designs," Accessed Nov 23, 2020.

[238] Yi-Ming Wang and Jinn-Shyan Wang, "An all-digital 50% duty-cycle corrector," in *IEEE International Symposium on Circuits and Systems*, vol. 2, pp. II–925, 2004.

[239] G. S. Alliance, "Electrostatic discharge (esd) in 3d-ic packages," Accessed Nov 21, 2020.

[240] Jie Wu, "A fault-tolerant and deadlock-free routing protocol in 2d meshes based on odd-even turn model," *IEEE Transactions on Computers*, vol. 52, no. 9, pp. 1154–1169, 2003.

[241] A. Zorat, "Construction of a fault-tolerant grid of processors for wafer-scale integration," *Circuits, Systems and Signal Processing*, vol. 6, no. 2, pp. 175–189, 1987.

[242] E. J. Marinissen, T. McLaurin, and Hailong Jiao, "Ieee std p1838: Dft standard-under-development for 2.5d-, 3d-, and 5.5d-sics," in *21st IEEE ETS*, pp. 1–10, 2016.

[243] W. C. Chen, C. Hu, K. C. Ting, V. Wei, T. H. Yu, S. Y. Huang, V. C. Y. Chang, C. T. Wang, S. Y. Hou, C. H. Wu, and D. Yu, "Wafer level integration of an advanced logic-memory system through 2nd generation cowos technology," in *VLSI Technology Symposium*, pp. T54–T55, 2017.

[244] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, pp. 84–90, May 2017.

[245] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. aurelio Ranzato, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1223–1231, Curran Associates, Inc., 2012.

[246] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[247] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016.

[248] PyTorch, "PyTorch: From Research To Production." `https://pytorch.org/`.

[249] A. Sergeev and M. D. Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *CoRR*, vol. abs/1802.05799, 2018.

[250] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," in *Advances in Neural Information Processing Systems*, pp. 1731–1741, 2017.

[251] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "On parallelizability of stochastic gradient descent for speech DNNs," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 235–239, 2014.

[252] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith,

J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *CoRR*, vol. abs/1609.08144, 2016.

[253] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” *CoRR*, vol. abs/1512.00567, 2015.

[254] A. Mirhoseini, H. Pham, Q. V. Le, B. Steiner, R. Larsen, Y. Zhou, N. Kumar, M. Norouzi, S. Bengio, and J. Dean, “Device placement optimization with reinforcement learning,” *CoRR*, vol. abs/1706.04972, 2017.

[255] A. Mirhoseini, A. Goldie, H. Pham, B. Steiner, Q. V. Le, and J. Dean, “A hierarchical model for device placement,” in *International Conference on Learning Representations*, 2018.

[256] W. A. Gardner, “Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique,” *Signal processing*, vol. 6, no. 2, pp. 113–133, 1984.

[257] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, “Parallelized stochastic gradient descent,” in *Advances in Neural Information Processing Systems 23* (J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, eds.), pp. 2595–2603, Curran Associates, Inc., 2010.

[258] Y. Huang, Y. Cheng, D. Chen, H. Lee, J. Ngiam, Q. V. Le, and Z. Chen, “Gpipe: Efficient training of giant neural networks using pipeline parallelism,” *CoRR*, vol. abs/1811.06965, 2018.

[259] J. Chen, R. Monga, S. Bengio, and R. Józefowicz, “Revisiting distributed synchronous SGD,” *CoRR*, vol. abs/1604.00981, 2016.

[260] S. H. Hashemi, S. A. Jyothi, and R. H. Campbell, “Communication scheduling as a first-class citizen in distributed machine learning systems,” *CoRR*, vol. abs/1803.03288, 2018.

[261] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, “Accurate, large minibatch SGD: training imagenet in 1 hour,” *CoRR*, vol. abs/1706.02677, 2017.

[262] S. L. Smith, P. Kindermans, and Q. V. Le, “Don’t decay the learning rate, increase the batch size,” *CoRR*, vol. abs/1711.00489, 2017.

[263] S. Jastrzebski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. J. Storkey, “Three factors influencing minima in SGD,” *CoRR*, vol. abs/1711.04623, 2017.

[264] M. Li, T. Zhang, Y. Chen, and A. J. Smola, “Efficient mini-batch training for stochastic optimization,” in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’14, (New York, NY, USA), pp. 661–670, ACM, 2014.

[265] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” *CoRR*, vol. abs/1609.04836, 2016.

[266] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of collective communication operations in mpich," *Int. J. High Perform. Comput. Appl.*, vol. 19, pp. 49–66, Feb. 2005.

[267] P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," *J. Parallel Distrib. Comput.*, vol. 69, pp. 117–124, Feb. 2009.

[268] S. Shi and X. Chu, "Performance modeling and evaluation of distributed deep learning frameworks on gpus," *CoRR*, vol. abs/1711.05979, 2017.

[269] NCCL, "NVIDIA Collective Communications Library." `https://developer.nvidia.com/nccl`, 2018.

[270] NVIDIA Container, "TensorFlow Release 18.07." `https://docs.nvidia.com/deeplearning/dgx/tensorflow-release-notes/rel_18.07.html`, 2018.

[271] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, June 2009.

[272] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[273] S. Migacz, "GNMT v2: PyTorch Implementation." `https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/Translation/GNMT`, 2018.

[274] L. Guillou, C. Hardmeier, P. Nakov, S. Stymne, J. Tiedemann, Y. Versley, M. Cettolo, B. Webber, and A. Popescu-Belis, "Findings of the 2016 WMT shared task on cross-lingual pronoun prediction," in *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, vol. 2, pp. 525–542, 2016.

[275] R. Józefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *CoRR*, vol. abs/1602.02410, 2016.

[276] NVIDIA, "DGX-1." `https://www.nvidia.com/en-us/data-center/dgx-1/`, 2018.

[277] NVIDIA, "Tesla V100 Tensor Core GPU." `https://www.nvidia.com/en-us/data-center/tesla-v100/`, 2018.

[278] NVIDIA, "NVLink Fabric: Advancing Multi-GPU Processing." `https://www.nvidia.com/en-us/data-center/nvlink/`, 2018.

[279] M. Ott, S. Edunov, D. Grangier, and M. Auli, "Scaling neural machine translation," *CoRR*, vol. abs/1806.00187, 2018.

[280] M. Yamazaki, A. Kasagi, A. Tabuchi, T. Honda, M. Miwa, N. Fukumoto, T. Tabaru, A. Ike, and K. Nakashima, "Yet another accelerated SGD: resnet-50 training on imagenet in 74.7 seconds," *CoRR*, vol. abs/1903.12650, 2019.

[281] T. Nowatzki, M. Sartin-Tarm, L. De Carli, K. Sankaralingam, C. Estan, and B. Robatmili, "A general constraint-centric scheduling framework for spatial architectures," *SIGPLAN Not.*, vol. 48, pp. 495–506, June 2013.

[282] T. Nowatzki, M. Sartin-Tarm, L. De Carli, K. Sankaralingam, C. Estan, and B. Robatmili, "A general constraint-centric scheduling framework for spatial architectures," in *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '13, (New York, NY, USA), pp. 495–506, ACM, 2013.

[283] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *CoRR*, vol. abs/1410.0759, 2014.

[284] D. Das, S. Avancha, D. Mudigere, K. Vaidyanathan, S. Sridharan, D. D. Kalamkar, B. Kaul, and P. Dubey, "Distributed deep learning using synchronous stochastic gradient descent," *CoRR*, vol. abs/1602.06709, 2016.

[285] O. Yadan, K. Adams, Y. Taigman, and M. Ranzato, "Multi-gpu training of convnets," *CoRR*, vol. abs/1312.5853, 2013.

[286] S. Sridharan, K. Vaidyanathan, D. D. Kalamkar, D. Das, M. E. Smorkalov, M. Shiryaev, D. Mudigere, N. Mellempudi, S. Avancha, B. Kaul, and P. Dubey, "On scale-out deep learning training for cloud and HPC," *CoRR*, vol. abs/1801.08030, 2018.

[287] A. Gholami, A. Azad, K. Keutzer, and A. Buluç, "Integrated model and data parallelism in training neural networks," *CoRR*, vol. abs/1712.04432, 2017.

[288] Z. Jia, S. Lin, C. R. Qi, and A. Aiken, "Exploring hidden dimensions in parallelizing convolutional neural networks," *CoRR*, vol. abs/1802.04924, 2018.

[289] M. Bauer, S. Treichler, E. Slaughter, and A. Aiken, "Legion: Expressing locality and independence with logical regions," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, (Los Alamitos, CA, USA), pp. 66:1–66:11, IEEE Computer Society Press, 2012.

[290] A. Harlap, D. Narayanan, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, and P. B. Gibbons, "Pipedream: Fast and efficient pipeline parallel DNN training," *CoRR*, vol. abs/1806.03377, 2018.

[291] S. Chaturapruek, J. C. Duchi, and C. Ré, "Asynchronous stochastic convex optimization: the noise is in the noise and sgd don't care," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 1531–1539, Curran Associates, Inc., 2015.

[292] S. Zhang, C. Zhang, Z. You, R. Zheng, and B. Xu, "Asynchronous stochastic gradient descent for DNN training," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6660–6663, May 2013.

[293] T. Paine, H. Jin, J. Yang, Z. Lin, and T. S. Huang, "GPU asynchronous stochastic gradient descent to speed up neural network training," *CoRR*, vol. abs/1312.6186, 2013.

[294] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in neural information processing systems*, pp. 693–701, 2011.

[295] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *CoRR*, vol. abs/1404.5997, 2014.

[296] Y. You, I. Gitman, and B. Ginsburg, "Scaling sgd batch size to 32k for imagenet training," 08 2017.

[297] J. Zhang and I. Mitliagkas, "Yellowfin and the art of momentum tuning," *arXiv preprint arXiv:1706.03471*, 2017.

[298] C. J. Shallue, J. Lee, J. M. Antognini, J. Sohl-Dickstein, R. Frostig, and G. E. Dahl, "Measuring the effects of data parallelism on neural network training," *CoRR*, vol. abs/1811.03600, 2018.

[299] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM J. Control Optim.*, vol. 30, pp. 838–855, July 1992.

[300] A. Koliousis, P. Watcharapichat, M. Weidlich, L. Mai, P. Costa, and P. Pietzuch, "Crossbow: Scaling deep learning with small batch sizes on multi-gpu servers," *arXiv preprint arXiv:1901.02244*, 2019.

[301] W. Xu, "Towards optimal one pass large scale learning with averaged stochastic gradient descent," *CoRR*, vol. abs/1107.2490, 2011.

[302] dsstne, "Amazon DSSTNE: Deep Scalable Sparse Tensor Network Engine." `https://github.com/amzn/amazon-dsstne`.

[303] OpenAI, "AI and Compute." `https://openai.com/blog/ai-and-compute/`.

[304] K. Olukotun, "Accelerating software 2.0," *ScaledML*, 2020.

[305] Z. Jia, M. Zaharia, and A. Aiken, "Beyond data and model parallelism for deep neural networks," *arXiv preprint arXiv:1807.05358*, 2018.

[306] A. A. Inferentia, "Achieve 12x higher throughput and lowest latency for PyTorch Natural Language Processing applications out-of-the-box on AWS Inferentia." `https://tinyurl.com/3mbuetmr`, (accessed Sep 10, 2021).

[307] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[308] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A systematic approach to dnn accelerator evaluation," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 304–315, 2019.

[309] J. Hestness, N. Ardalani, and G. Diamos, “Beyond human-level accuracy: Computational challenges in deep learning,” in *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming*, pp. 1–14, 2019.

[310] “Deep learning’s diminishing returns.” `https://spectrum.ieee.org/deep-learning-computational-cost`. Accessed: 2021-10-15.

[311] A. Stillmaker and B. Baas, “Scaling equations for the accurate prediction of cmos device performance from 180nm to 7nm,” *Integration*, vol. 58, pp. 74–81, 2017.

[312] “Wikichip: Technology node.” `https://en.wikichip.org/wiki/technology_node`. Accessed: 2021-10-15.

[313] “Hbm3: Big impact on chip design.” `https://semiengineering.com/hbm3s-impact-on-chip-design/`. Accessed: 2021-10-15.

[314] A. Arunkumar, E. Bolotin, B. Cho, U. Milic, E. Ebrahimi, O. Villa, A. Jaleel, C.-J. Wu, and D. Nellans, “Mcm-gpu: Multi-chip-module gpus for continued performance scalability,” in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 320–332, 2017.

[315] S. Pal, D. Petrisko, M. Tomei, P. Gupta, S. S. Iyer, and R. Kumar, “Architecting waferscale processors - a gpu case study,” in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 250–263, 2019.

[316] “Tesla dojo.” `https://en.wikipedia.org/wiki/Tesla_Dojo`. Accessed: 2021-10-15.