Scale-Out Processors

THÈSE N^{O} 5906 (2013)

PRÉSENTÉE LE 24 SEPTEMBRE 2013

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE D'ARCHITECTURE DE SYSTÈMES PARALLÈLES
PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Pejman LOTFI KAMRAN

acceptée sur proposition du jury:

Prof. W. Zwaenepoel, président du jury
Prof. B. Falsafi, directeur de thèse
Prof. G. De Micheli, rapporteur
Dr P. Ranganathan, rapporteur
Prof. P. Stenstrom, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2013

I am deathless, I am the eternal Lord
For I have spread the seed of the Word.
— Ferdowsi

To my parents...

Acknowledgments

I dedicate this thesis to my parents. From the time that I can remember, they were the source of inspiration, motivation, and encouragement for me. Living far from them over the past five years has shown me the impact that they had on all of the success of my life. Thanks for all the trust, encouragement, and love!

I would like to thank my advisor, Prof. Babak Falsafi. He showed me the true meaning of academic research. Babak put me in an interesting and challenging research direction, which fulfilled all of my academic desires. If it were not for the joy of working on such an extraordinary research project, I would have left the Ph.D. program years ago when I faced, for the first time, the difficulties and pains of being a Ph.D. student. I also need to thank him for providing me with five years of financial support, which enabled me to focus on research.

I want to take this opportunity to thank the committee members of my thesis defense, Giovanni De Micheli, Partha Ranganathan, Per Stenstrom, and Willy Zwaenepoel. Their feedback and criticisms have improved this dissertation significantly.

My Ph.D. research was partially supported by EuroCloud, Project No 247779 of the European Commission 7th RTD Framework Programme – Information and Communication Technologies: Computing Systems. In addition to the financial support, as part of the EuroCloud project, I had the opportunity to work with wonderful colleagues from ARM, IMEC, Nokia, and University of Cyprus. In the last year of my Ph.D., I also benefited from an Intel Ph.D. fellowship award. I would like to thank the European Commission 7th RTD Framework Programme and Intel for partially supporting my Ph.D. research.

There are many people who contributed to this research; without them, I could never have achieved what I did. First and foremost, I need to thank Boris Grot. Since he joined PARSA (our laboratory) two years ago, my life as a Ph.D. student became easier. He is a great person to work with, and I was lucky to have him on my side. I want to acknowledge Mike Ferdman and thank him for being my role model in the first four years of my Ph.D. (when he was at PARSA). It is worthwhile to mention that I did send more emails to Mike than I sent to anybody else, and surprisingly, I always received a response quickly! Almutaz Adileh, Onur Kocberber, Djordje Jevdjic, Javier Picorel, and Stavros Volos are PARSA members who significantly contributed to the success of this thesis. Moreover, I need to mention EuroCloud members, Damien Hardy, Sachin Idgunji, Chrysostomos Nicopoulos, Emre Ozer, and Yanos Sazeides, who contributed

to this research. I also owe a debt of gratitude to Nikos Hardavellas, Tom Wenisch, Jason Zebchuk, and the many who contributed to the development of the Flexus infrastructure. Tom developed the core of the Flexus infrastructure, Jason coded the cache hierarchy, and Nikos built the analytic model that I used for most of my research work. Thank you all!

I am grateful to my friends and colleagues, Mohammad Alisafaee, Sotiria Fytraki, Effi Georgala, Cansu Kaynak, Jennifer Sartor, and Evangelos Vlachos, who listened to my talks, proofread my papers, and gave me invaluable feedback on various stages of my research. I also need to thank EuroCloud members, Bushra Ahsan, Tony Gore, Dragomir Milojevic, Andreas Panteli, Andreas Prodromou, and Isidoros Sideris, who accompanied me in various EuroCloud physical meetings, socialized after the meetings, and made physical meetings fun.

I would like to thank Nancy Chong, Ved Gund, Dongha Jung, Samira Manabi, Mehdi Modarressi, and Mahmood Naderan for all of the good times that I had when they were at EPFL. Specifically, the time when Dongha, Samira, and Mehdi were at EPFL was the best period of my Ph.D. life. I need to give special thanks to Hadi Esmaeilzadeh, my friend of 15 years, who was there for me every time I needed to talk to somebody. I also owe a debt of gratitude to Stéphanie Baillargues, Sylvie Fiaux, Valérie Locca, and Rodolphe Buret for handling the administrative tasks with which I needed help over the years.

There are many more who contributed to this success. If I wanted to mention every name, thousands of pages would not be enough. Please accept my apologies, and let me thank you all here. Thank you!

Lausanne, September 7, 2013 Pejman Lotfi-Kamran

Abstract

Global-scale online services, such as Google's Web search and Facebook's social networking, run in large-scale datacenters. Due to their massive scale, these services are designed to scale out (or distribute) their respective loads and datasets across thousands of servers in datacenters. The growing demand for online services forced service providers to build networks of datacenters, which require an enormous capital outlay for infrastructure, hardware, and power consumption. Consequently, efficiency has become a major concern in the design and operation of such datacenters, with processor efficiency being of, utmost importance, due to the significant contribution of processors to the overall datacenter performance and cost.

Scale-out workloads, which are behind today's online services, serve independent requests, and have large instruction footprints and little data locality. As such, they benefit from processor designs that feature many cores and a modestly sized Last-Level Cache (LLC), a fast access path to the LLC, and high-bandwidth interfaces to memory. Existing server-class processors with large LLCs and a handful of aggressive out-of-order cores are inefficient in executing scale-out workloads. Moreover, the scaling trajectory for these processors leads to even lower efficiency in future technology nodes.

This thesis presents a family of throughput-optimal processors, called *Scale-Out Processors*, for the efficient execution of scale-out workloads. A unique feature of Scale-Out Processors is that they consist of multiple stand-alone modules, called pods, wherein each module is a server running an operating system and a full software stack. To design a throughput-optimal processor, we developed a methodology based on performance density, defined as throughput per unit area, to quantify how effectively an architecture uses the silicon real estate. The proposed methodology derives a performance-density optimal processor building block (i.e., pod), which tightly couples a number of cores to a small LLC via a fast interconnect. Scale-Out Processors simply consist of multiple pods with no inter-pod connectivity or coherence. Moreover, they deliver the highest throughput in today's technology and afford near-ideal scalability as process technology advances. We demonstrate that Scale-Out Processors improve datacenters' efficiency by 4.4x-7.1x over datacenters designed using existing server-class processors.

Keywords: Scale-Out Workloads, Datacenters, Efficient Processors, Performance Density, NOC-Out, Scale-Out Processors

Résumé

A l'échelle mondiale, les services en lignes comme la recherche Web de Google et le réseau social Facebook, fonctionnent grâce à des centres de stockage de données de grande envergure. En raison de leur importance, ces services sont conçus pour redistribuer leur charge de travail et l'ensemble des données sur les milliers de serveurs des centres de stockage. La demande croissante des services en ligne a forcé les fournisseurs de services à construire des réseaux de données, qui nécessitent une énorme mise de fonds pour couvrir les frais liés à l'infrastructure, au matériel et à la consommation d'énergie. Par conséquent, l'efficacité est devenue une préoccupation majeure pour la conception et le fonctionnement de ces centres de stockage, et en particulier l'efficacité des processeurs, d'une extrême importance, en raison de leur contribution significative à l'augmentation des performances des centres de données et à la diminution des coûts d'exploitation.

Les applications à déploiement horizontal, qui sont à la base des services en ligne aujourd'hui, envoient des requêtes indépendantes, et ont de larges gammes d'instructions et peu de données locales. Par conséquent, elles bénéficient de conceptions de processeurs qui comportent de nombreux noyaux et un dernier niveau de cache (LLC) de taille modeste, une voie d'accès rapide à la LLC et des interfaces à large bande passante dans la mémoire. Les processeurs des serveurs existants, qui ont de grands LLC et quelques noyaux agressifs de type out-of-order sont inefficaces à l'exécution des applications à déploiement horizontal. Par ailleurs, la trajectoire de mise à l'échelle pour ces processeurs se traduit par une efficacité encore plus faible dans les futurs centres technologiques.

Cette thèse présente une famille de processeurs à performance optimale, appelés les *Processeurs Scale-Out*, pour l'exécution efficace de d'applications à déploiement horizontal. Une caractéristique unique des Processeurs Scale-Out est leur constitution en plusieurs modules autonomes, appelés pods. Chaque pod contient un serveur qui exécute un système d'exploitation et une pile logicielle complète. Pour concevoir un processeur à performance optimale, nous développons une méthodologie basée sur la densité de la performance (PD), qui est définie comme le débit par unité de surface, pour mesure comment une architecture efficace utilise la surface en silicium. La méthodologie proposée utilise un bloc de processeurs à performance optimale (c'est-à-dire un pod), qui associe étroitement plusieurs noyaux à un petit LLC via une interconnexion rapide. Les Processeurs Scale-Out sont simplement composés de plusieurs modules, sans connectivité ni cohérence. De plus, ils offrent les meilleures per-

formances technologiques actuelles et une évolutivité presque idéale avec les avancées de la technologie. Nous démontrons que les Processeurs Scale-Out améliorent l'efficacité des centres de stockage de données de 4.4 à 7.1 fois par rapport aux centres de données conçus pour utiliser des processeurs de classe serveur.

Mots-clefs : Applications à Déploiement Horizontal, Centre de Stockage des Données, Processeurs Efficaces, Densité de la Performance, NOC-Out, Processeurs Scale-Out

Contents

List of Figures

List of Tables

1 Introduction

We are living in an era in which Information Technology (IT) is shaping our society. More than anytime in history, our society is dependent on IT for its day-to-day activities. Education, media, science, social networking, etc. are all affected by IT. The steady growth in processor performance is one of the driving forces behind the success and widespread adoption of IT.

Historically, the improvement in processor performance was driven by two phenomena: Moore's law [69] and Dennard scaling [28]. Technology scaling, which refers to the technology of shrinking transistor dimensions, provided processor designers with twice transistor density every two years (Moore's law). Moreover, the reduction in the supply voltage enabled processor designers to operate twice the number of transistors that technology offers without an increase in power consumption (Dennard scaling). Taking advantage of Moore's law and Dennard scaling, computer architects improved the processing power by constantly increasing the complexity of the processor pipeline and the frequency of the processor. Decades of technology scaling allowed powerful processors with deep and aggressive Out-of-Order (OoO) pipelines and high clock frequency to emerge.

Unfortunately, improving the performance of processors with the historical approach is no longer viable. As physical restrictions slow down the reduction of the supply voltage, Dennard scaling has effectively stopped [27]. While Moore's law is still valid and the number of transistors increases by a factor of two every two years, the failure of Dennard scaling makes power and energy the primary constraints of processors. As such, it is no longer desirable to increase the clock frequency of processors or increase the complexity of the processor pipeline to improve performance, as these approaches are not energy efficient [76]. For this reason, we did not see a noticeable increase in the clock frequency of processors for almost a decade.

As the historical approach for improving processor performance no longer works, since 2004, vendors have started to produce multi-core processors using relatively aggressive OoO cores [33]. This paradigm shift was motivated by the fact that many workloads have inherent thread-level parallelism and can benefit from multiple cores. With improvement in process technology, processor vendors keep the complexity of the cores constant and use the extra

transistors to increase the number of cores and the size of the last-level cache (LLC). As caches consume less energy compared to the cores, processors use almost half of their transistor budget for the LLC with the hope that a larger cache captures a larger fraction of the data working sets and results in faster execution.

In the meanwhile, we are witnessing that workloads are becoming more and more data-centric. We are living in the age of data explosion. The trend for many workloads, such as social networking or online advertising, is to collect and process larger and bigger volumes of data. Unfortunately, there is a mismatch between the workloads trend and the trend in the processor industry. While data-centric workloads are memory-intensive and do not benefit from aggressive OoO cores [40, 39, 30], such cores are offered in the existing products. Moreover, the little data locality in the emerging data-centric workloads makes large LLCs in the existing multi-core processors ineffective [43, 42, 39, 30]. To deliver the computational power necessary for future data-centric workloads, processors need to be redesigned to match the requirements of these workloads.

Global-scale online services represent an important class of data-centric workloads [30, 31]. As scalability is the primary concern for such services, they are designed to scale out (or distribute) the load across a large number of servers in datacenters. Service providers like Google, Microsoft, and Facebook rely on scale-out workloads that run in large-scale datacenters with thousands of servers to deliver media streaming, Web search, and social networking. The vast datasets of scale-out workloads are sharded across servers in datacenters. As data access latency is crucial, shards of the datasets are kept in large-capacity DRAM memories.

Scale-out workloads have common characteristics that can be leveraged to design high-performance processors, namely (a) massive parallelism; (b) little communication; (c) large instruction footprint; and (d) little data locality. As datacenters serve many independent requests coming from users across the world, scale-out workloads have abundant request-level parallelism. Due to the fact that these requests are mostly independent and the datasets are extremely large, there is rarely a need for communication when two requests are being processed. Moreover, these workloads serve a variety of complex requests, and as a result, their instruction footprints are large (hundreds of kilobytes to megabytes). Finally, the vast datasets combined with the independent nature of requests results in little data locality in these workloads.

A processor optimized for the execution of scale-out workloads needs to have many cores to benefit from the abundant parallelism in these workloads. Moreover, the little data locality calls for a medium-sized cache, which also leaves more die area for the cores. The large instruction footprints of scale-out workloads cannot be captured in an L1 instruction cache, and as such, they reside in the L2 cache. Due to the fact that instructions are shared across all the cores, the optimized processor needs to have a shared L2 cache that we refer to as the last-level cache (LLC). Moreover, because instructions are resident in the LLC, the optimized processor needs to provide a fast access path from the individual cores to the LLC for the

instruction delivery. The minimal communication in scale-out workloads calls for minimal connectivity between the cores and makes designing a fast access path from the cores to the LLC trivial. Finally, as the datasets are resident in memory, the optimized processor needs to have high-bandwidth interfaces to frequently access memory.

Scale-out workloads have multi-tier architecture with complex software stacks. Even though there is little communication in these workloads due to the independence of requests and the vast data working set sizes, the shared-memory programming model is valued in the scale-out domain, as it simplifies software development and facilitates the use of existing software stacks. This fact, combined with the common characteristics of scale-out workloads, calls for a shared-memory processor with many cores, a modestly sized LLC, minimal connectivity, fast access to the LLC, and high-bandwidth memory channels.

1.1 Why Not Existing Processors?

Today's volume servers are designed with processors that are essentially general-purpose. These conventional processors combine a handful of aggressively speculative and high clock-frequency cores supplemented by a large shared on-chip cache. As manufacturing technology provides higher transistor density, conventional processors use the additional transistors to scale up the core count, cache capacity, coherence mechanism, and interconnect.

Recently, tiled processors have emerged as competition to volume processors in the scale-out server space [90]. Recognizing the importance of per-server throughput, these processors use a large number of relatively simple cores, each with a slice of the shared LLC, interconnected via a packet-based mesh interconnect. Lower-complexity cores are more efficient than those in conventional designs [59]. Despite the differences in the chip-level organization, the technology scaling trends of tiled processors are similar to conventional designs; each technology generation affords more tiles, which increases the core count, cache capacity, and interconnect resources.

In the context of processors for scale-out workloads, both architectures make suboptimal use of the die area and cannot maximize throughput. One of the inefficiencies of these processors is their large last-level caches. Maximizing throughput necessitates a careful choice in the size of the cache. Smaller caches that can capture the dynamic instruction footprints of scale-out workloads afford more die area for the cores without penalizing per-core performance. Moreover, we demonstrate that while the simpler cores found in tiled designs are more effective than conventional server cores for scale-out workloads, the latency incurred by the on-chip interconnect in tiled organizations lowers performance and limits the benefits of integration, as additional tiles result in more network hops and longer delays.

1.2 Future Projections

As process technology advances and more transistors become available, existing processors use the higher transistor density to increase the number of cores and size of the last-level cache. While scale-out workloads do not benefit from a large last-level cache and existing processors already have large last-level caches beyond what is required for these workloads, future projections indicate that processor vendors plan to increase the size of the LLC even further. Consequently, not only existing processors suffer from large last-level caches when they execute scale-out workloads, but also, this problem exacerbates as manufacturing technology improves.

Moreover, scale-out workloads require fast access to the last-level cache at minimum because their instruction footprints are large; hence, they reside in the LLC. As technology scales and manufacturing capability increases, more cores will be added to processors. In existing processor organizations, there is a direct relationship between the access latency to the LLC and the number of cores that share it. As the number of cores in existing processor organizations increases, the access latency to the LLC also increases.

While existing processors suffer from large last-level caches and/or slow access to the LLC, the projections indicate that both shortcomings will exacerbate with improvement in transistor scaling. The shortcomings of existing processors for the execution of scale-out workloads necessitate a processor design methodology, which would eliminate the shortcomings and enable seamless scalability with improvement in transistor scaling.

1.3 Scale-Out Design Methodology

Scale-out datacenters require processors with many cores and a fast access path to the last-level cache to maximize throughput. As scale-out workloads do not benefit from a large last-level cache (due to little data locality in these workloads), processors optimized for scale-out workloads should dedicate most of their die area to cores and maximize the number of cores. While it is easy to increase the core count, we demonstrate that existing processor organizations fundamentally cannot fulfill both requirements – i.e., many cores and fast access from the cores to the LLC – simultaneously.

As more cores are added to a processor with a fixed-size LLC, both the die area of the processor and the physical distance between the individual cores and the last-level cache increase. Because the time it takes for a core to access the last-level cache is proportional to the physical distance between the core and the LLC, the longer distance between the core and the LLC, which is an artifact of integrating more cores, results in slower access to the LLC. Existing processor organizations are fundamentally incapable of providing fast access to the LLC while many cores are integrated in the processor. We need a new processor organization to break the relationship between core count and LLC access latency. The new processor organization should enable integrating many cores while providing low LLC access latency.

To address the limitation of existing processor organizations, we propose the scale-out design methodology. The scale-out design methodology calls for many-core processors based on the notion of pods. A pod is a module that tightly couples several cores to a modestly sized LLC through a low-latency interconnect. The proposed methodology breaks the relationship between more cores and higher LLC access latency by partitioning the die area of a processor and integrating a pod in each partition. Each pod is a self-contained server-on-a-chip running a full software stack. We formulate a methodology to determine the optimal number of cores and LLC capacity to integrate in a pod for peak throughput. The proposed design, called the Scale-Out Processor, delivers peak throughput in today's process technology and affords near-ideal scalability as the technology scales.

1.4 Organization of Scale-Out Processors

As Scale-Out Processors are composed of many pods, the microarchitecture of pods is essentially the main factor that determines the performance of Scale-Out Processors. While partitioning the die area through integrating many pods enables Scale-Out Processors to break the relationship between core count and LLC access latency across pods, providing fast access from cores to the LLC within a pod is necessary for the success of Scale-Out Processors.

Unfortunately, the existing many-core organizations force a compromise between performance and area cost [64]. While mesh-based tile organizations have a modest area and wire cost, they incur latency overheads through a many-hop organization. In contrast, many-core organizations based on the richly connected topologies, such as a flattened butterfly [55], offer low LLC access latency at high area and wire cost. Pods in Scale-Out Processors require organizations that have low area overhead and provide fast access to the last-level cache.

To provide fast access to the LLC with low area overhead, we propose a novel organization using a simple and critical observation: there is almost no core-to-core communication in scale-out workloads [64, 71, 63]. Based on this observation, the proposed organization (a) decouples cores and the last-level cache; (b) eliminates all unneeded core-to-core links; and (c) uses specialized core-to-LLC networks to connect cores to the last-level cache and vice versa. The bottom line is that the proposed organization delivers the performance of the state-of-the-art with $1/10^{th}$ of the area.

1.5 3D Scale-Out Processors

As scaling down transistor dimensions becomes more complicated and challenging [10], the validity of Moore's law, which is the primary driving force behind the growth of the semiconductor industry, is expected to end [66]. Three-dimensional integration of multiple logic dies is a propitious mechanism that can extend the validity of Moore's law. In a 3D integration, multiple logic dies are stacked on top of each other and interconnected by through-silicon vias (TSVs).

In a conventional 2D integration, the scaling of transistor dimensions implies more transistors and also larger average distance between the transistors. As technology scales and more transistors become available, the average distance between the transistors increases because it is impossible to scale global wire length with technology [12]. On the contrary, in a 3D integration, more transistors become available by stacking more logic dies on top of each other. As the vertical distance is much shorter (i.e., in the order of μm) than the horizontal distance (i.e., in the order of mm), more transistors in a 3D integration come without an increase in the average distance.

Because of the negligible vertical distance in a 3D integration, as well as to benefit from the 3D integration, pods should span vertically across all of the logic dies that are stacked on top of each other. This feature enables two possible directions for 3D integrated pods: (1) Pods grow the number of cores and LLC capacity as more logic dies are added while benefiting from the 3D integration to keep the on-chip distance constant; and (2) Pods keep their core count and LLC capacity constant and use 3D integration to reduce the on-chip distance. Negligible vertical distance enables 3D pods to have either more cores and larger LLC or shorter on-chip distance as compared to 2D pods. Consequently, 3D integration increases the throughput of Scale-Out Processors.

1.6 Dissertation Contributions

In this dissertation, we investigate the design of throughput-optimal processors for scale-out workloads. We begin by demonstrating that existing processor organizations fall short of efficiency in executing scale-out workloads. We then motivate and justify Scale-Out Processors and provide an organization for them. We show that Scale-Out Processors can significantly increase performance per total cost of ownership in datacenters. We conclude by exploring the design space of Scale-Out Processors manufactured with 3D logic-on-logic technology as the likely enabler of Moore's law when transistor scaling stops.

Through a combination of analytic modeling, trace-driven analysis, and cycle-accurate, full-system simulation of various multi-core processors running scale-out workloads, we demonstrate:

Inefficiency of existing processors. We demonstrate that existing processors are incapable of running scale-out workloads efficiently. We identify misallocation of on-chip resources and long LLC access latency as the two main bottlenecks of existing processors.

Fundamental limitation of existing processor organizations. We identify the relationship between the increase in core count and the increase in LLC access latency as the fundamental limitation of existing processor organizations. This fundamental limitation prevents existing processor organizations from maximizing throughput of scale-out workloads.

Scale-Out Processors. We demonstrate the scale-out design methodology that enables avoid-

ing the fundamental limitation of existing processor organizations – namely, the relationship between the increase in core count and the increase in LLC access latency. Moreover, we demonstrate the effectiveness of the resulting processors, i.e., Scale-Out Processors, for efficient execution of scale-out workloads.

Microarchitecture of Scale-Out Processors. We show that existing multi-core microarchitectures are either area-inefficient or performance limiters in the context of Scale-Out Processors. Taking advantage of the characteristics of scale-out workloads, we demonstrate a novel microarchitecture for Scale-Out Processors to achieve both area efficiency and high performance.

Benefits of Scale-Out Processors in datacenters. We demonstrate the effectiveness of Scale-Out Processors in the context of datacenter efficiency. We use *Performance per Total Cost of Ownership* as the efficiency metric at datacenters to show the superiority of Scale-Out Processors as compared to existing processors.

Multi-pod Scale-Out Processors. We demonstrate that Scale-Out Processors with large dies are more efficient than Scale-Out Processors with small dies. We compare single- and multi-pod Scale-Out Processors at datacenters and show that multi-pod Scale-Out Processors are more efficient.

Scale-Out Processors in the post-Moore era. We study the impact of 3D logic-on-logic technology on Scale-Out Processors. Three-dimensional logic-on-logic technology is the likely successor of transistor scaling for extending the validity of Moore's law. We demonstrate that Scale-Out Processors can leverage features specific to 3D logic-on-logic technology to improve performance beyond what is possible in a standard 2D process.

1.7 Dissertation Organization

The rest of this dissertation is organized as follows. In Chapter 2, we study the behavior of scale-out workloads and demonstrate the mismatch between existing processor organizations and the requirements of these workloads. In Chapter 3, we introduce the scale-out design methodology and quantify its benefits. In Chapter 4, we microarchitect a Scale-Out Processor to minimize the area cost and maximize the performance. In Chapter 5, we compare various processor organizations in the context of datacenter total cost of ownership to show the effectiveness of multi-pod Scale-Out Processors. In Chapter 6, we extend the organization of Scale-Out Processors for implementation in 3D logic-on-logic technology as the likely enabler of the post-Moore era. Finally, we comment on related research in Chapter 7 and conclude in Chapter 8.

2 A Case For Scale-Out Processors

Datacenters are the computing platforms for delivering scalable online services. Google, Microsoft, and Facebook rely on networks of datacenters to deliver search capabilities, social networking, and a growing number of other offerings. The scale-out software architecture at the core of the online service model effectively accommodates dataset and demand growth by simply distributing the load on many servers, as servers handle independent requests that do not share any state.

With typical scale-out workloads distributed across thousands of servers inside a datacenter, performance characteristics of each server dictate the datacenter's throughput. In TCO-conscious datacenters, performance per TCO dollar is maximized by increasing the throughput of each server processor, which enables better memory utilization and affords higher per-server performance without a commensurate increase in cost [92].

To maximize the throughput of scale-out workloads, server processors need to have many cores to serve independent requests in parallel and also need to get high throughput from each individual core. For individual cores to deliver high throughput, server processors need to provide a fast path to the LLC that holds the instructions and data secondary working sets.

We motivate the need for Scale-Out Processors based on the observation that the two requirements (i.e., many cores and a fast path to the LLC) cannot be satisfied simultaneously with existing many-core processor organizations. In existing organizations, as the number of cores increases, the distance from the cores to the LLC increases, which results in longer LLC access latency. In this chapter, we show that existing processors are not capable of meeting the two requirements at the same time and, as such, are not suitable for the execution of scale-out workloads.

2.1 What Do Scale-Out Workloads Want?

In this section, we examine a representative set of scale-out workloads in order to understand the demands they place on server processors. Research analyzing the scale-out workload

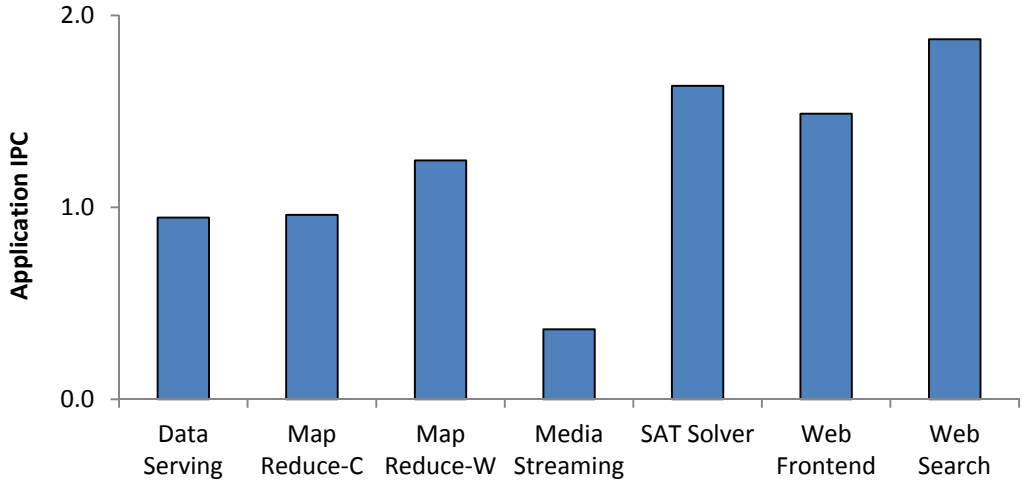


Figure 2.1: Application instructions executed per cycle for an aggressive OoO core (out of maximum IPC of 4).

domain has shown that a key set of traits holds across a wide range of workloads, including Web search, media streaming, and Web serving. These traits can be summarized as request independence, little communication, large instruction footprint, and vast dataset [30]. These traits favor server processors with (a) many simple OoO cores; (b) modestly sized LLC; (c) a fast access path to the LLC; (d) minimal connectivity between cores; and (e) high-bandwidth memory interfaces. Next, we examine each of these requirements in more detail.

2.1.1 Simple OoO Cores

Scale-out workloads have vast datasets that are kept in memory to minimize the response latency [30]. On one hand, scale-out workloads are memory-intensive and spend most of their time accessing data in memory; as such, their benefit from aggressive OoO cores is limited [30]. On the other hand, the instruction-level parallelism (ILP) in these workloads is not as low as conventional memory-intensive workloads (e.g., database workloads) [79].

Figure 2.1 shows the application instructions per cycle (IPC) for an aggressive OoO core with commit width of four and 2GHz frequency. Out of seven scale-out workloads, only one of them (i.e., Media Streaming) has an IPC of less than one. Two of the workloads (i.e., Data Serving and MapReduce-C) have an IPC of around one, and four of the workloads have an IPC of greater than one and less than two. While the core is aggressive OoO and can commit up to four instructions per cycle, on average, at most two instructions are committed every cycle. These results corroborate prior work [30] and show that aggressive OoO cores are not suitable for scale-out workloads. Moreover, four out of seven workloads have an IPC of greater than one, which suggests that simple OoO cores can be useful for these workloads, corroborating prior work [79].

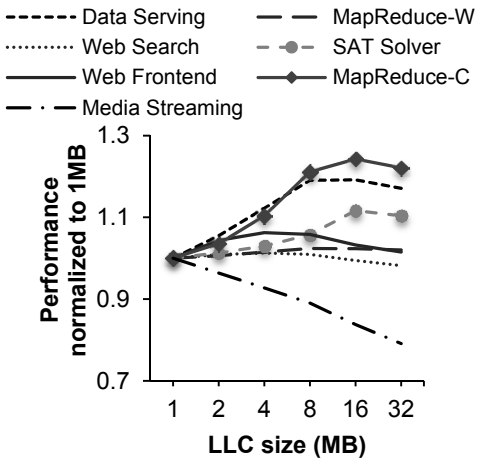


Figure 2.2: Performance of 4-core workloads varying the LLC size.

2.1.2 Many Cores

As mentioned in the previous section, scale-out workloads are memory-intensive and spend a considerable fraction of their execution time waiting for data to arrive from memory. As such, aggressive OoO cores are not well-suited for the execution of these workloads. Fortunately, scale-out workloads, like media streaming, social networking, or Web search, handle a stream of requests that are, to an overwhelming extent, mutually independent [30]. Fundamentally, request independence is the feature that makes scale-out workloads inherently parallel. The abundant request-level parallelism argues for processor designs with a large number of cores to maximize throughput.

2.1.3 Modestly Sized LLC

We seek to establish the range of last-level cache sizes appropriate for scale-out workloads. As it has been shown that a Non-Uniform Cache Architecture (NUCA) L2 cache matches the performance of a multi-level cache hierarchy [53], we evaluate a two-level cache hierarchy with a NUCA LLC in this dissertation.

We analyze the cache requirements of scale-out workloads by sweeping the size of the LLC from 1 to 32MB. The results are presented for a quad-core processor, but note that the general trends are independent of the core count. Details of the methodology can be found in Section 2.4.3.

Figure 2.2 plots the performance (i.e., the aggregate number of application instructions committed per cycle [89]) of individual applications normalized to a design with a 1MB LLC. For most of the workloads, LLC capacities of 2-8MB are sufficient to capture the instruction footprint and secondary working set. Beyond this range, larger cache configurations provide limited benefit because the enormous data working sets of the workloads exhibit little reuse in the LLC. Two of the workloads (MapReduce-C and SAT Solver) exhibit a different behavior,

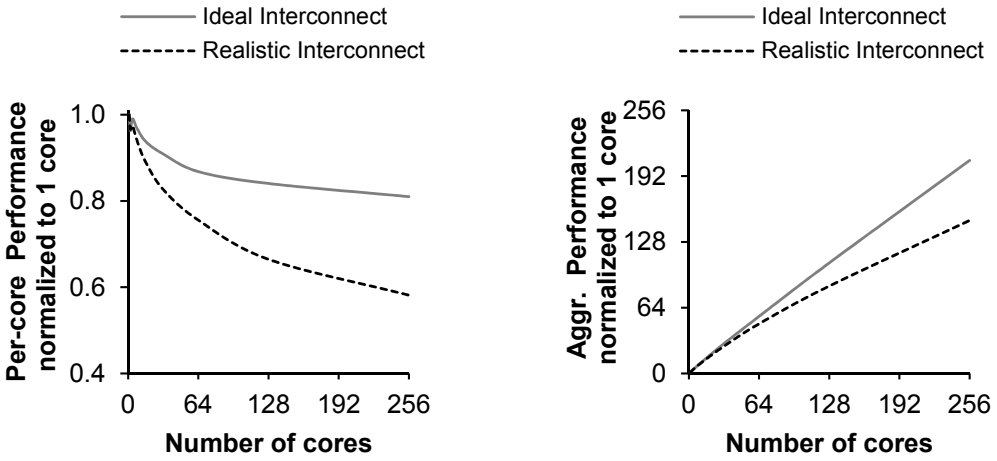


Figure 2.3: Per-core performance of 4-core workloads with a 4MB LLC varying the number of cores (a) and chip-level performance with a 4MB LLC varying the number of cores (b).

as larger caches do help in capturing the secondary working set. However, even for these workloads, a 16-fold increase in cache size from 1 to 16MB translates into a performance gain of just 12-24%. Cache capacity beyond 16MB is strictly detrimental to performance, as the reduction in miss rate is offset by the increased access latency. These results corroborate prior characterizations of scale-out and traditional server workloads executing on chip multiprocessors [30, 43].

2.1.4 Fast Access to LLC

We analyze the sensitivity of scale-out workloads to the on-chip distance and the sharing degree. We fix the LLC size at 4MB and examine the performance as the number of cores varies from 1 to 256. Figure 2.3 plots per-core performance (a) and throughput per chip (b) averaged across workloads and normalized to a single-core baseline. The two lines in the figures correspond to an ideal organization with a fixed-latency interconnect between each core and the LLC (solid gray line), as well as a realistic mesh-based interconnect where the physical distance between cores and cache banks affects the LLC access latency (dashed black line).

In the case of an ideal interconnect, Figure 2.3a shows that the degradation in per-core performance associated with having many cores share the LLC is small (e.g., 16% for a 128x increase in core count from 2 to 256 cores). As a result, Figure 2.3b demonstrates that aggregate performance can be improved by a factor of 210 by sharing a 4MB LLC among 256 cores.

In the case of a design subject to physical constraints, in which the distance to the LLC

grows with core count, the negative slope of the performance curve in Figure 2.3a is much steeper. The distance to the LLC has a direct effect on performance due to a combination of primary working set sizes greatly exceeding the L1 capacity and the memory-intensive nature of scale-out workloads, which makes these workloads particularly sensitive to the average memory access time. As a result, Figure 2.3b shows that a design based on a realistic interconnect reduces performance by 28% when compared to an ideal network at 256 cores, demonstrating how distance effects threaten the ability of server processors to reach their throughput potential.

Overall, scale-out workloads show limited benefit from LLC capacities beyond 8MB. Furthermore, a moderately sized cache can be effectively shared among a large number of cores. However, maximizing the performance in a system with a heavily shared LLC requires mitigating interconnect delays.

2.1.5 Minimal Connectivity

Scale-out workloads serve (mostly) independent requests that originate from many users around the world. To serve these requests, cores that execute scale-out workloads process massive-scale datasets. As datasets of these workloads are extremely large, the datasets are usually sharded, and a core only works on a shard of the datasets.

Two cores that work on two different shards either do not communicate, or the communication is so rare that existing multi-core processors use Ethernet (which is slow) as the standard medium for the communication [7, 9, 6]. Even for cores that process the same shard of data, independent nature of requests, combined with the fact that shards of large-scale datasets are themselves large, results in minimal communication [71, 63, 64]. In a multi-core processor executing scale-out workloads, when a core serves a request, it rarely needs to communicate with other cores.

As communication in scale-out workloads is rare, minimal connectivity is sufficient in processors that execute (or are optimized to execute) these workloads. The minimal connectivity requirement of scale-out workloads suggests that processors optimized for these workloads have a simple organization.

2.1.6 Adequate Number of Memory Interfaces

A feature of scale-out workloads is that they work on massive-scale datasets. As many of the scale-out workloads have strict quality-of-service requirements, the datasets (or a significant fraction of the datasets) are placed in memory to minimize the access latency. As such, these workloads frequently access memory to obtain various pieces of the datasets.

As cores that execute scale-out workloads are inside the processor chip and the DRAM memory is outside of the chip, memory requests need to be sent off-chip (through pins) to go to the

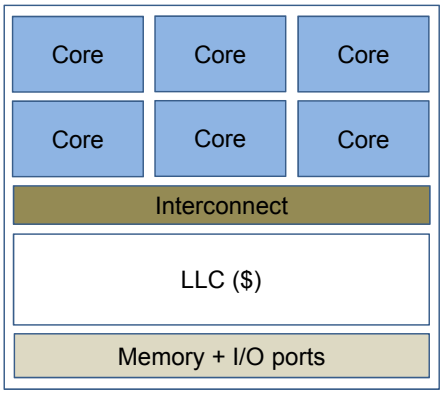


Figure 2.4: A conventional processor.

DRAM memory. Memory interfaces are the modules that connect a processor chip to the off-chip DRAM memory. As a processor might have multiple memory interfaces, a memory access needs to be sent to one of the memory interfaces (determined by a policy like static interleaving) to be passed to the DRAM memory.

A memory interface can sustain a constant amount of off-chip traffic, which is a function of the memory interface protocol and the number of pins dedicated to the memory interface. If the off-chip traffic exceeds the capacity of a memory interface, more than one memory interface is necessary to provide connectivity between the cores and the off-chip DRAM memory. The off-chip traffic of a workload that is running on a multi-core processor can be estimated using simulation or empirical analysis. The number of memory interfaces must be chosen based on the worst-case off-chip traffic of the workloads.

2.2 What Do Existing Processor Organizations Offer?

In this section, we examine the suitability of various server processors (i.e., conventional, tiled, and optimized tiled processors) for efficient execution of scale-out workloads.

2.2.1 Conventional Processors

Conventional processors combine a handful of aggressively speculative and high clock-frequency cores supplemented by a large shared on-chip cache in a dancehall architecture where cores and caches are connected using a crossbar. Examples of commercial conventional processors include Intel Xeon 5670 [46] or AMD Opteron 6320 [2]. As manufacturing technology provides higher transistor density, conventional processors use the additional transistors to scale up the core count, cache capacity, coherence, and interconnect layer. Figure 2.4 shows a typical conventional processor with six cores.

The first limitation of conventional processors is their large last-level caches. While existing

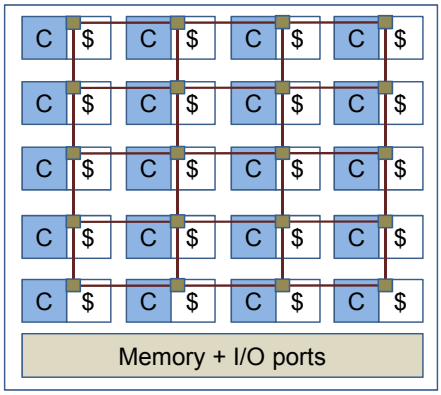


Figure 2.5: A tiled processor.

conventional processors have 12 to 24MB of LLC, and chip manufacturers plan to increase the size of the LLC in the future technologies, scale-out workloads do not benefit from large LLCs. Not only do large LLCs waste silicon real estate that could have been better used for cores, but also, the long access latency of large LLCs deteriorates the performance of scale-out workloads, as shown in Figure 2.2.

Moreover, conventional processors have large, aggressive OoO cores that are not effective for memory-intensive workloads. The large die area of the aggressive OoO cores used in the conventional processors limit the number of cores to just a few. Consequently, conventional processors cannot leverage the abundant request-level parallelism in scale-out workloads to maximize throughput.

Finally, the reliance of the dancehall architectures on crossbar interconnects makes such organizations unattractive for many-core processors due to the poor scalability of crossbars.

2.2.2 Tiled Processors

To overcome the scalability limitations of crossbar-based designs, many-core server processors can employ a tiled organization with a fully distributed last-level cache. Figure 2.5 shows an overview of a many-core processor based on a tiled design. Each tile consists of a core, a slice of the distributed last-level cache, and a router. The tiles are linked via a routed, packet-based, multi-hop interconnect in a mesh topology. An example of commercial tiled processors includes Tilera's TILE-Gx 3036 [83], which has 36 cores and 12MB of cache.

The tiled organization and a structured interconnect fabric allow mesh-based designs to scale to large core counts. Unfortunately, the regularity of the mesh topology works to its disadvantage when it comes to performance scalability. Each hop in a mesh network involves the traversal of a multi-ported router, which adds delay due to the need to access the packet buffers, arbitrate for resources, and navigate the switch. As Figure 2.3 shows, these delays diminish the performance of a mesh-based processor as the number of cores and, consequently,

on-chip distance increases.

Moreover, just like conventional processors, existing tiled processors have large last-level caches beyond what is required by scale-out workloads. Large LLCs in tiled processors waste silicon real estate and prevent tiled processors from maximizing core count and, as a result, throughput.

2.2.3 Optimized Tiled Processors

In this section, we discuss various optimization proposals, which are suggested in the computer architecture research community, for eliminating the shortcomings of tiled processors.

LLC-Optimal Tiled Processors

Tiled processors usually divide the die area equally between cores and caches. As a result, a significant fraction of the die area in tiled processors is devoted to last-level caches in spite of the fact that large LLCs are not beneficial for scale-out workloads. Given this observation, *LLC-optimal tiled* processors [39] use only a small fraction of their die area to last-level caches (aggregate LLC capacity of 4-8MB as motivated in Figure 2.2) and devote a significant fraction of their die area to cores. This optimization enables tiled processors to maximize the number of cores. The abundant request-level parallelism in scale-out workloads enables *LLC-optimal tiled* processors to increase throughput with every extra core.

Reducing the size of the last-level cache, large interconnect-induced LLC access latency is the primary shortcoming of *LLC-optimal tiled* processors. As the latency of the mesh interconnect grows with core count, and because *LLC-optimal tiled* processors have more cores as compared to tiled processors, the negative impact of the interconnect latency on the performance of *LLC-optimal tiled* processors is even more drastic.

LLC-Optimal Tiled Processors with Instruction Replication

Instruction footprints of scale-out workloads do not fit in L1 instruction caches and are stored in the last-level cache. One challenge with large, LLC-resident instruction footprint is that the on-die distance between the cores and the LLC adds delay to the cache access time. Because L1-I misses stall the processor, scale-out workloads are particularly sensitive to the on-die communication delays due to frequent instruction fetches from the LLC.

Researchers proposed instruction replication in the last-level cache as a way to reduce the distance between the cores and the LLC slice that serves an instruction miss. The state-of-the-art instruction replication in LLC slices of a tiled processor is R-NUCA [41]. R-NUCA replicates instructions in LLC slices in a way that instruction blocks are, at most, one network hop away from the requesting core. For this goal, R-NUCA divides tiles into clusters of size four and

replicates instructions in each cluster.

While instruction replication is effective for standard tiled processors, which have large LLCs, *LLC-optimal tiled* processors have a small LLC, and instruction replication increases the pressure on the LLC. Instruction replication in the small LLC of *LLC-optimal tiled* processors increases the number of LLC misses and reduces the benefit of instruction replication. Moreover, instruction replication can only reduce the on-chip interconnect delay of instruction accesses – data accesses do not benefit from this optimization.

2.3 Metric for the Design-Space Evaluation

Scale-out workloads are inherently parallel; consequently, they are best served by substrates that provide a large number of cores to achieve high per-server throughput. However, higher core density leaves less silicon real estate for on-die caches, while simultaneously increasing the physical core-to-cache distance and interconnect delays. The cache should be large enough to capture the dynamic instruction footprint and shared OS data, yet small enough to provide fast access, which is particularly important for instruction fetches that lie on the critical path of execution. The physical distance between the cores and cache must also be short in order to minimize the delay due to the interconnect.

To capture these conflicting requirements in a single metric for assessing processor efficiency, we propose performance density (PD), defined as performance per mm^2. Given a core microarchitecture, PD provides a simple means of comparing a range of designs that differ in core count, LLC size, and interconnect parameters.

2.4 Methodology

We compare the performance, area, and energy efficiency of various processors to an ideal processor, which has a small LLC and an ideal 4-cycle interconnect, to highlight the inefficiencies of existing processor organizations for the execution of scale-out workloads.

2.4.1 Design and Technology Parameters

Baseline (40nm). We compare the various chip architectures in 40nm technology with an on-chip supply voltage of 0.9V. The choice of the 40nm as the baseline technology node is to highlight that even with a relatively old technology node, existing processor organizations are inefficient, and there is room for improvement. We model chips with an area of 250-280mm^2, a power budget of 95W, and a maximum of six single-channel DDR3 interfaces; these parameters are representative of existing server processors fabricated in 40 and 45nm process technology.

Design parameters are summarized in Table 2.1 and Table 2.2. We consider three types of cores and a range of cache sizes. Conventional processors feature an aggressive, 4-wide, large

Table 2.1: Area and power estimates for various system components at 40nm.

Component		Area	Power
Cores	**Conventional**	25mm^2	11W
	OoO	4.5mm^2	1W
	In-order	1.3mm^2	0.48W
LLC	**16-way SA**	5mm^2 per MB	1W per MB
Interconnect		0.2 - 4.5 mm^2	<5W
DDR3 interface (PHY+ controller)		(2 + 10) mm^2	5.7W
SoC components		42mm^2	5W

instruction-window core microarchitecture. Tiled and ideal processors are assessed using two types of cores: (1) high-performance, three-way out-of-order core, similar to ARM Cortex-A15 [85]; and (2) dual-issue in-order core, resembling ARM Cortex-A8 [5]. To simplify the comparison, we assume a 2GHz operating frequency for all three core types. Cache parameters are estimated using CACTI 6.5 [72].

We estimate the area of the memory interfaces and other SoC components by scaling the micrograph of a Nehalem processor in 45nm technology [57]. We measure the power consumption of a DDR3-1667 channel to be 5.7W. Assuming effective utilization of 70% [24], a 12.8GB/s channel provides 9GB/s of useful bandwidth. We estimate the power of other SoC components and interfaces to be 5W using McPAT v0.8 [58] configured to model Sun UltraSPARC T2.

Scaling study (20nm). To understand the effect of technology scaling on the different processor configurations, we also model them in 20nm technology. We choose the 20nm technology node to study the effect of technology scaling because it is the technology node currently in use by major processor manufacturers, and it highlights the inefficiency of existing processor organizations in today's technology node. We assume perfect area scaling of cores and caches over two technology generations. Per ITRS estimates, we model a supply voltage of 0.8V and choose not to increase the frequency to minimize power dissipation. We find that the analog circuitry in the PHYs prevents memory interfaces from truly benefiting from technology scaling. We evaluate systems with the emerging DDR4 interface, which is expected to double per-channel memory bandwidth over DDR3 [32].

2.4.2 Scale-Out Workloads

Scale-out workloads that are used for the evaluation and include Data Serving, MapReduce, Media Streaming, SAT Solver, Web Frontend, and Web Search are taken from CloudSuite 1.0 [1, 30]. We present two MapReduce workloads: Text Classification and Word Count (referred to as MapReduce-C and MapReduce-W, respectively). For the Web Frontend workload, we use the banking option from SPECweb2009 in place of its open-source counterpart from

Table 2.2: Specification of various system components.

Processing Cores	*Conventional*: 4-wide dispatch/retirement 128-entry ROB, 32-entry LSQ, 2GHz *Out-of-order*: 3-wide dispatch/retirement 60-entry ROB, 16-entry LSQ, 2GHz *In-order*: 2-wide dispatch/retirement, 2GHz
L1I / D Caches	*Conventional*: 64KB, 4(8)-way I(D) cache 3-cycle load-to-use, 2 ports, 32 MSHRs *Rest*: 32KB, 2-way, 2-cycle load-to-use, 1 port, 32 MSHRs
Last-Level Cache	16-way set-associative, 64B lines, 64 MSHRs, 16-entry victim cache
Interconnect	*Ideal Interconnect*: 4 cycles *Mesh*: 3 cycles/hop (includes both router and channel delay)
Main Memory	45ns access latency

CloudSuite, as SPECweb2009 exhibits better performance scalability at high core counts. All evaluated systems run Solaris 10 operating system.

2.4.3 Performance Evaluation

As evaluating performance of various processors requires evaluating configurations with many cores (more than 64 cores), we use a verified analytic model[1] [39] to predict the performance of various processors. The model extends the classical average memory access time analysis to predict the aggregate number of application instructions committed per cycle (i.e., performance) for a given LLC capacity and core count; the model is parametrized by simulation results, including core performance, cache miss rates, and interconnect delay.

2.5 Results

We now compare various processor organizations to an ideal processor. Conventional chip design is representative of existing products. The tiled in-order organization is similar to the Tilera Tile64 processor [90]. For the conventional processor, we dedicate one memory channel for every four cores, while for various tiled and ideal processors, we measure the bandwidth demand for every core/cache organization and workload using simulation. In addition, we provision the number of memory channels to accommodate the worst-case bandwidth demand across the workload spectrum. For all chip compositions, we model as many cores as can be afforded without exceeding the area, energy, and bandwidth constraints specified in Section 2.4.1.

[1] The accuracy of the analytic model is evaluated in the next chapter.

Table 2.3: Specification of various processor designs at 40nm.

	40nm						
Processor design	PD	Cores	LLC (MB)	MCs	Die (mm^2)	Power (Watt)	Perf/ Watt
Conventional	0.026	6	12	2	276	94	0.08
Tiled (OoO)	0.060	20	20	1	245	50	0.29
LLC-Optimal Tiled (OoO)	0.084	32	8	2	251	56	0.38
LLC-Optimal Tiled with IR (OoO)	0.086	32	8	3	264	62	0.37
Ideal (OoO)	0.101	32	8	2	251	56	0.45
Tiled (IO)	0.099	64	20	2	251	67	0.37
LLC-Optimal Tiled (IO)	0.131	96	6	5	261	86	0.40
LLC-Optimal Tiled with IR (IO)	0.145	96	6	5	261	86	0.44
Ideal (IO)	0.167	96	6	5	261	86	0.51

2.5.1 40nm Technology

Conventional: 2MB of LLC per core. Cores and caches are interconnected via a crossbar. One DDR3 channel is used for every four cores. Six cores can be afforded without exceeding the 95W power budget. The resulting processor reaches a performance density of 0.026.

We begin the study of tiled and ideal processors with out-of-order cores in the 40nm technology.

Tiled with OoO cores: 1MB of LLC per tile. The OoO tiled design is area-limited; a total of 20 cores can be integrated on a $250mm^2$ die while maintaining a regular grid topology with a reasonable aspect ratio. This organization uses a mesh interconnect with 3-cycle per-hop latency. The resulting organization needs one memory channel and achieves a performance density of 0.061.

LLC-optimal tiled with OoO cores: 256KB of LLC per tile. The OoO *LLC-optimal tiled* design is area-limited; a total of 32 cores can be integrated on a $250mm^2$ die. The aggregate LLC capacity in the OoO *LLC-optimal tiled* is 8MB (aggregate across all 32 cores), as larger LLC capacities only deteriorate performance. The resulting organization needs two memory channels and reaches a performance density of 0.084.

LLC-optimal tiled with IR and OoO cores: Same core count and LLC capacity as *LLC-optimal tiled* with OoO cores but uses the R-NUCA instruction replication (IR) to minimize instruction access latencies. This processor needs three memory channels and reaches a performance density of 0.086.

Ideal processor with OoO cores: 32 cores and 8MB of LLC (same as *LLC-optimal tiled* with OoO cores) interconnected using an ideal interconnect with a 4-cycle latency. The ideal processor requires two memory channels and gets a performance density of 0.101.

Compared to the conventional design (see Table 2.3), the ideal processor with out-of-order

cores achieves nearly 3.9x higher performance density, thanks to its greater execution resources, resulting from lower-complexity cores and a smaller LLC. Performance density of the ideal processor is 1.7x higher when compared to the tiled processor. The latter is hampered by over-provisioned cache capacities and excessive LLC access delays stemming from the multi-hop topology. A tiled design, even with the same LLC capacity as the ideal processor (i.e., *LLC-optimal tiled*), shows a 17% lower PD at 32 cores as compared to the ideal processor due to its larger interconnect latency. *LLC-optimal tiled*, even with an optimization like instruction replication, is 15% behind the ideal processor at 40nm technology, while both processors have the same number of cores and LLC capacity, highlighting the importance of low LLC access latency.

In the past 10 years, there has emerged a trend toward simpler cores in server processors. Companies like Tilera target scale-out datacenters with chips based on simple in-order cores, validating prior research showing that such designs are well-suited for certain scale-out workloads [59]. Although server processors based on simple cores may sacrifice latency, for services whose primary concern is throughput (e.g., data analysis), high-throughput designs that integrate many simple cores may be preferred to organizations with fewer cores of higher performance. Following this trend, we continue the study with in-order cores in 40nm technology.

Tiled with in-order cores: The tiled processor with in-order cores maintains the same core-to-cache area ratio of the OoO design. The resulting configuration affords 64 cores and 20MB of LLC, with area as the limiting factor. This processor requires two memory channels and achieves a performance density of 0.099.

LLC-optimal tiled with in-order cores: 64KB of LLC per tile. The in-order *LLC-optimal tiled* design is area-limited; a total of 96 cores can be integrated on a 260mm^2 die while maintaining a regular grid topology with a reasonable aspect ratio. The LLC capacity in the in-order *LLC-optimal tiled* is 6MB (aggregate across 96 cores), as larger LLC capacities only deteriorate performance. The resulting processor requires five memory channels and reaches a performance density of 0.131.

LLC-optimal tiled with IR and in-order cores: Same core count and LLC capacity as *LLC-optimal tiled* with in-order cores but uses the R-NUCA instruction replication (IR) to minimize instruction access latencies. This processor needs five memory channels and achieves a performance density of 0.145.

Ideal processor with in-order cores: 96 cores and 6MB of LLC (the same as *LLC-optimal tiled* with in-order cores) interconnected using an ideal interconnect with a 4-cycle latency. This processor needs five memory channels and obtains a performance density of 0.167.

The ideal processor achieves a 1.7x improvement in performance density over a tiled design with in-order cores, and a 6.4x improvement over a conventional processor. Compared to a tiled design with the same LLC capacity as the ideal processor (i.e., *LLC-optimal tiled*

Table 2.4: Specification of various processor designs at 20nm.

	20nm						
Processor design	PD	Cores	LLC (MB)	MCs	Die (mm^2)	Power (Watt)	Perf/ Watt
Conventional	0.067	12	48	3	213	93	0.15
Tiled (OoO)	0.206	80	80	2	256	80	0.66
LLC-Optimal Tiled (OoO)	0.258	112	28	4	251	83	0.78
LLC-Optimal Tiled with IR (OoO)	0.294	112	28	4	251	83	0.89
Ideal (OoO)	0.366	112	28	4	251	83	1.11
Tiled (IO)	0.227	180	80	4	249	94	0.60
LLC-Optimal Tiled (IO)	0.360	224	12	6	202	86	0.84
LLC-Optimal Tiled with IR (IO)	0.362	192	12	6	192	80	0.87
Ideal (IO)	0.518	224	12	6	202	86	1.21

with in-order cores), the ideal processor obtains a 1.3x higher performance density. While an optimization like instruction replication improves the performance density of the *LLC-optimal tiled* processor, the resulting processor is 13% behind the ideal processor.

2.5.2 Projection to 20nm Technology

To understand the effect of technology scaling on various processor organizations, we project all of the processors to the 20nm technology node. In the tiled design, per-hop delays remain the same as in the 40nm baseline, as greater wire RC delays are offset by a reduction in tile dimensions.

For all processor configurations, we keep the core-to-cache area ratio the same as that of 40nm. For the conventional processor, we dedicate one memory channel for every four cores, while for various tiled and ideal processors, we provision the number of memory channels based on the bandwidth demands. For all configurations, we populate the die area with as many cores as we can afford without exceeding the area, energy, or bandwidth constraints specified in Section 2.4.1. The specifications of the resulting processors is shown in Table 2.4.

Ideal processor based on out-of-order cores is area-limited, occupying 251mm^2; additionally, it dissipates 83W of power at peak load and achieves a performance density of 0.366, an improvement of 3.6x over the 40nm baseline. While ideal scaling predicts PD to improve by a factor of 4, the growth in area dedicated to the memory interfaces that do not benefit from technology scaling reduces the fraction of the die available to compute and dampens the gains in PD.

Compared to the ideal processor, a technology-scaled conventional design is power-limited at 20nm. It integrates 12 cores on a die, improving performance density by a factor of 2.6 from the 40nm design. The tiled architecture with out-of-order cores enjoys ideal scaling in core count, reaching 80 cores in an area-limited configuration, with performance density growing by 3.4x over the 40nm baseline. Performance improvements in the tiled organization

is constrained by the growth in the network diameter, which increases LLC access delays. The *LLC-optimal tiled* with (IR and) OoO cores is area-limited at 20nm. The performance density of the *LLC-optimal tiled* (with IR) shows 3.1x (3.4x) improvement over the design at 40nm. The ideal processor shows the strongest scalability, improving performance density by 5.5x, 1.8x, 1.4x, and 1.2x over conventional, tiled, *LLC-optimal tiled* and *LLC-optimal tiled with IR*, respectively, when implemented in 20nm technology.

In an implementation based on in-order cores, the ideal processor is bandwidth-limited, assuming a constraint of six DDR4 memory controllers on a die. Technology scaling improves performance density by a factor of 3.1, instead of the expected factor of 4, due to the large area overheads introduced by the on-die memory interfaces necessary to feed the processor.

Unlike the ideal processor, the tiled processor is power-limited at 20nm. Compared to the 40nm design, the tiled processor has 2.8x more cores (180 vs. 64) and 2.3x higher performance density. Moreover, the *LLC-optimal tiled* with (IR and) in-order cores is bandwidth-limited at 20nm. The number of cores in the *LLC-optimal tiled with IR* is 14% less than that of *LLC-optimal tiled* because both processors are bandwidth-limited, and the instruction replication increases the off-chip bandwidth demands. The performance density of the *LLC-optimal tiled* with (IR and) in-order cores shows 2.7x (2.5x) improvement over the design at 40nm. In absolute terms, the ideal processor improves performance density by 7.7x, 2.3x, 1.4x, and 1.4x over conventional, tiled, *LLC-optimal tiled*, and *LLC-optimal tiled with IR*, respectively, when all processors are engineered with in-order cores in the 20nm technology.

2.5.3 Summary

The conventional architecture achieves the lowest performance density among the evaluated designs at both 40 and 20nm technology nodes. Conventional designs have low performance density because (a) the caches are overprovisioned, allowing less area for compute; and (b) the compute area is misallocated, as aggressive cores provide only a small performance improvement over less aggressive out-of-order cores, yet they consume considerably more area.

The tiled processors using a mesh-based interconnect and out-of-order cores achieve 2.2x higher performance density than the conventional processor in 40nm technology (3.1x in 20nm). The use of lower-complexity cores improves performance density and, as a result, throughput; however, the large LLC and the delays associated with a multi-hop interconnect limit the performance gains. The *LLC-optimal tiled* processors improve performance density over tiled processors by 1.4x and 1.3x in 40nm and 20nm technology, respectively. To limit the performance loss of the multi-hop interconnect, *LLC-optimal tiled* processors can use the instruction replication [41] in the LLC to optimize for the instruction accesses. In 40nm (20nm) technology, the *LLC-optimal tiled* processors with the instruction replication achieve 2% (16%) higher performance density over *LLC-optimal tiled* organization.

The highest performance density is achieved with the ideal processor, which has a small LLC and an ideal 4-cycle interconnect. The ideal processor with out-of-order cores improves performance density by 3.9x, 1.7x, 1.2x, and 1.2x over conventional, tiled, *LLC-optimal tiled*, and *LLC-optimal tiled with IR*, respectively, in 40nm technology (5.5x, 1.8x, 1.4x, and 1.2x over the respective designs in 20nm).

On workloads with laxer QoS requirements, higher performance density (and, consequently, higher throughput) can be achieved through the use of in-order cores. In such cases, the ideal processor improves performance density by 6.4x (7.7x) and 1.7x (2.3x) over conventional and tiled processors, respectively, in 40nm (20nm) technology. Moreover, the ideal processor achieves higher performance density over *LLC-optimal tiled* and *LLC-optimal tiled* processors that are optimized with the instruction replication by 1.8x (1.4x), and 1.2x (1.4x), respectively, in 40nm (20nm) technology. The results corroborate prior work, which shows that low-complexity cores are well-suited for throughput workloads [25, 43].

The results presented in this chapter show a significant gap between the performance density of existing processor organizations and that of ideal processors for the execution of scale-out workloads. This dissertation aims at proposing Scale-Out Processors for efficient execution of scale-out workloads. Scale-Out Processors approach the performance density of ideal processors.

3 A Methodology to Design Scale-Out Processors

Scale-out workloads, which are behind many of today's online services, as a class, have a set of common characteristics that differentiate them from desktop, media processing, and scientific domains [18]. A typical scale-out workload, be it a streaming service or Web search, handles a stream of mostly independent client requests that require accessing pieces of data from a vast dataset. Processing a diversity of requests, scale-out workloads have large instruction footprints.

The presence of common traits – namely, (a) request independence; (b) large instruction footprints; and (c) vast dataset sizes – indicates that processors can be optimized for this workload class. The abundant request-level parallelism argues for processor designs with a large number of cores to maximize throughput. The independent nature of requests virtually eliminates inter-thread communication activity; however, large instruction footprints require a fast path between the individual cores and the last-level cache (LLC) containing the applications' instructions and data. Finally, the vast dataset dwarfs on-die storage capacities and offers few opportunities for caching due to limited reuse. (See Section 2.1 for more details.)

As discussed in the previous chapter, existing processor organizations are not capable of offering all of the features necessary for efficient execution of scale-out workloads. While existing processor organizations can offer many cores and a small LLC, in existing organizations, with the increase in core count, the access latency of individual cores to the last-level cache increases. As the number of cores increases, the distance from the cores to the LLC also increases, which results in a longer LLC access latency. This fundamental shortcoming makes existing processor organizations inherently unsuitable for the execution of scale-out workloads. Unfortunately, technology scaling exacerbates the problem: as technology scales and more cores are offered, the distance from the cores to the LLC and, as a result, the LLC access latency in existing processor organizations, increases.

In this chapter, we seek to develop a methodology to design technology-scalable server chips for scale-out workloads that simultaneously offer all of the features necessary for the efficient execution of scale-out workloads. In this methodology, performance density (PD), defined as

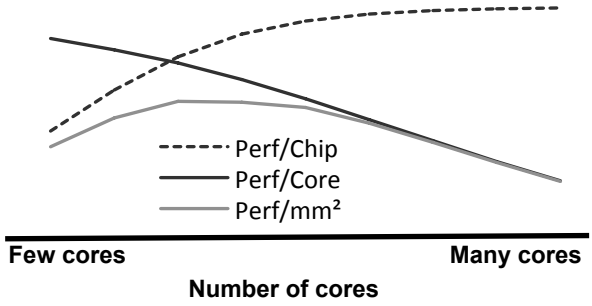


Figure 3.1: Performance per core, performance per chip, and performance density for a hypothetical workload.

throughput per unit area, is used to quantify how effectively an architecture uses the silicon real estate. The proposed design methodology derives a performance-density optimal processor building block called a pod, which tightly couples a number of cores to a small LLC via a fast interconnect. A key aspect of the proposed architecture is that pods are stand-alone servers, with no inter-pod connectivity or coherence. The pod methodology enables processors to scale seamlessly with technology, side-stepping the challenges of scaling both software and hardware to large core counts, while at the same time guaranteeing maximum throughput and optimally-efficient use of the on-chip real estate.

3.1 Motivation

We propose performance density (PD), defined as performance per mm^2 for assessing processor efficiency. Given a core microarchitecture, PD provides a simple means of comparing a range of designs that differ in core count, LLC size, and interconnect parameters.

Figure 3.1 provides the intuition behind the performance density metric using a hypothetical workload whose behavior is representative of scale-out workloads. The x-axis plots the number of cores for a fixed-size cache. The number of cores increases to the right of the graph, resulting in a higher core-to-cache ratio. The black solid line plots per-core performance, which diminishes as the number of cores grows due to the combination of distance and sharing at the LLC. The dashed line shows the aggregate throughput, which scales with the additional core resources, but the growth is sub-linear in the core count due to the eroding per-core throughput. Finally, the gray line plots performance density, whose peak represents an optimal configuration that maximizes performance per unit area by balancing core count, LLC capacity, sharing, and distance factors.

3.2 Scale-Out Design Methodology

Today's server chips, such as the conventional processors and the emerging tiled designs, scale performance through the addition of cores, cache capacity, interconnect and coherence

resources, and miscellaneous glue logic. This scaling strategy is a characteristic of the scale-up model. This dissertation argues that the model of increasing processor complexity is counter-productive for scale-out workloads because additional resources do not yield a commensurate improvement in chip-level performance.

To overcome the limitations of the conventional design methodology, we have developed a technology-scalable approach for maximizing the performance of server processors targeting scale-out workloads. This approach uses performance density as an optimization metric and builds on a simple observation that, given a configuration that is PD-optimal, the most profitable way of scaling aggregate performance is to grow the number of PD-optimal units on a chip. This strategy maintains the optimal performance density while increasing the aggregate throughput. In contrast, an approach that expands a PD-optimal configuration through additional core and cache resources lowers performance density and leads to a chip organization whose peak throughput is suboptimal for a given area budget.

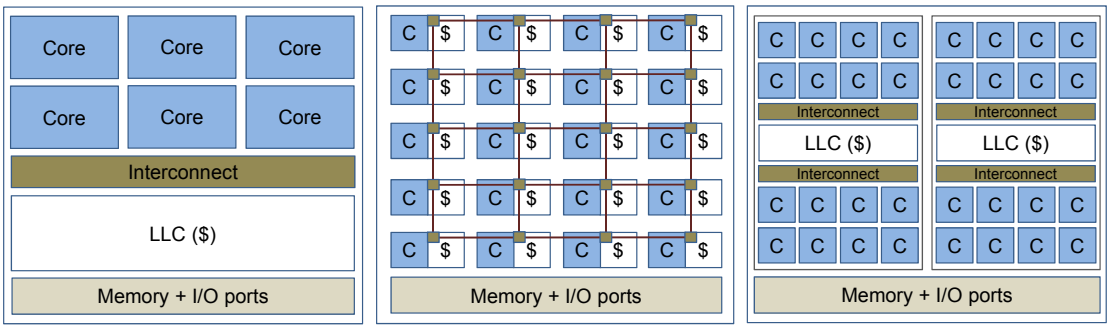
3.2.1 Pod as a Building Block

The notion at the heart of a Scale-Out Processor is the pod, a PD-optimal organization of core, cache, and interconnect resources. A pod is a complete server that runs its own copy of the operating system. Depending on the characteristics of the underlying process technology and component microarchitecture, a single pod may require only a fraction of the available die area, power, and bandwidth budget. To fully leverage the benefits of integration, multiple pods can be placed on a die. In effect, a pod acts as the tiling unit in a Scale-Out Processor. Adding more pods does not affect the optimality of each individual pod, allowing performance to scale linearly with the pod count. Because each pod is a complete server-on-a-die, direct inter-pod connectivity is not required. Thus, perfect performance scalability comes at a negligible integration expense that side-steps the challenge of scaling up the global interconnect and coherence infrastructure. The lack of inter-dependence among pods is a feature that fundamentally sets Scale-Out Processors apart from existing chip organizations and enables optimality-preserving scaling across technology generations.

Figure 3.2 captures the spirit of our approach and highlights the differences from existing designs by comparing the chip-level organization of conventional, tiled, and Scale-Out designs. In the rest of this section, we explain the features of a PD-optimal pod and describe the implications of a pod-based design at the chip level.

3.2.2 Pod Features

Scale-out workloads have large instruction footprints [30], which are not well-accommodated by private caches [41]. A shared LLC is, thus, a better choice, as it can capture the working set of application and OS instructions, along with OS and thread-private data, without the performance and area expense of per-core private cache hierarchies. However, once these



(a) Conventional (b) Tiled (c) Scale-Out

Figure 3.2: Comparison of conventional, tiled, and Scale-Out architectures. The Scale-Out design features two pods.

elements are captured, larger LLC configurations do not benefit scale-out workloads whose datasets greatly exceed capacities of practical on-die caches.

Because much of the useful capacity of the shared LLC comes from the common instruction and OS working set, the cache is naturally amenable to high degrees of sharing, a trend shown in Figure 2.3. However, the high incidence of misses at the L1-I mandates an LLC organization with low access latency and a fast core-to-cache interconnect. Thus, the performance density of a pod is maximized by balancing throughput gains arising from having many cores share an LLC against the reduction in per-core performance stemming from longer cache access latencies.

The core microarchitecture, cache parameters, and interconnect characteristics all play their respective roles in determining a PD-optimal organization by influencing factors that include the cache bank access latency, core-to-cache distance, wire delays, and the pressure placed by each core on the cache. Across the spectrum of scale-out workloads, modest cache capacities in the range of 2-8MB are sufficient. Meanwhile, the number of cores required to maximize performance density for the range of parameters considered in our studies varies from 16 (for out-of-order cores) to up to 32 (for in-order cores).

3.2.3 Chip-Level Considerations

A Scale-Out chip is a simple composition of one or more pods and a set of memory and I/O interfaces. Multi-pod designs reduce the number of chips for a given throughput target or power budget. Having fewer chips on a motherboard is beneficial in high-density datacenter racks, where motherboard space is limited, and reducing the number of sockets per board may be an effective cost-reduction strategy.

Integrating multiple pods on a die improves efficiency by sharing the on-die DRAM and I/O ports, increasing bandwidth utilization, and reducing pin requirement. Scale-Out chips that

Table 3.1: System parameters for cycle-accurate, full-system simulations.

CMP Size	1-64 cores (Data Serving, MapReduce, SAT Solver) 1-32 cores (Web Frontend, Web Search) 4-16 cores (Media Streaming)
Processing Cores	*Conventional*: 4-wide dispatch/retirement 128-entry ROB, 32-entry LSQ, 2GHz *Out-of-order*: 3-wide dispatch/retirement 60-entry ROB, 16-entry LSQ, 2GHz *In-order*: 2-wide dispatch/retirement, 2GHz
L1I / D Caches	*Conventional*: 64KB, 4(8)-way I(D) cache 3-cycle load-to-use, 2 ports, 32 MSHRs *Rest*: 32KB, 2-way, 2-cycle load-to-use, 1 port, 32 MSHRs
Last-Level Cache	16-way set-associative, 64B lines, 64 MSHRs, 16-entry victim cache *UCA*: 1 bank per 4 cores; *NUCA*: 1 bank per tile
Interconnect	*Ideal Crossbar*: 4 cycles *Crossbar*: 1-8 cores: 4 cycles; 16, 32, 64 cores: 5, 7, and 11 cycles respectively *Mesh*: 3 cycles/hop (includes both router and channel delay)
Main Memory	45ns access latency

share pins among pods necessitate a global interconnect layer to connect the individual pods to the memory and I/O ports. Fortunately, such a layer can be kept trivial due to the limited connectivity that it must provide, because pod-to-pod communication is not needed, and the number of external interfaces is low.

The number of pods on a die is dictated by physical constraints, such as area, power, and pin bandwidth. The lack of interdependence among pods is a valuable feature that eliminates the need for pod-to-pod communication infrastructure and chip-wide coherence support. The absence of these mechanisms reduces the design complexity of Scale-Out Processors and boosts their scalability.

3.3 Methodology

We use Flexus [93, 89] for cycle-accurate, full-system simulation of various chip multi-processor (CMP) configurations. Flexus extends the Virtutech Simics functional simulator with timing models of in-order and out-of-order cores, caches, on-chip protocol controllers, and interconnect. We model CMPs with 1 to 64 cores, various cache sizes, and three different on-chip interconnects. The details of the simulated architecture are listed in Table 3.1. Flexus models the SPARC v9 ISA and is able to run unmodified operating systems and applications.

We use the SimFlex multiprocessor sampling methodology [89]. The samples are drawn over an interval of 10 seconds (30 seconds for Media Streaming) of simulated time. For each measurement, we launch simulations from checkpoints with warmed caches and branch

predictors and run 100K cycles (2M cycles for Data Serving) to achieve a steady state of detailed cycle-accurate simulation before collecting measurements for the subsequent 50K cycles. We use the ratio of the number of application instructions committed per cycle to the total number of cycles (including the cycles spent executing operating system code) to measure performance; this metric has been shown to accurately reflect overall system throughput [89]. Performance measurements are computed with 95% confidence with an average error of less than 4%.

Because finding optimal pod configurations for Scale-Out chips requires evaluating a large design space, we augment the simulation-based studies with an analytic model [39] to limit the extent of time-intensive simulation. The model extends the classical average memory access time analysis to predict per-core performance for a given LLC capacity; the model is parametrized by simulation results, including core performance, cache miss rates, and interconnect delay.

3.4 Results

We now compare Scale-Out Processor designs to conventional and tiled organizations.[1] For each Scale-Out design, we first find a performance-density-optimal pod organization; then, we integrate pods up to the area, energy, and bandwidth limits per Section 2.4.1. We start by validating the analytic model against the results obtained via cycle-accurate simulation and then use the analytic model for the rest of the study.

3.4.1 Model Validation

Figure 3.3 illustrates the performance density results for designs with out-of-order cores, a 4MB LLC, and different interconnect types across our scale-out workloads. We model three different interconnects: a mesh, an ideal crossbar with a constant delay that is independent of the number of interconnected components, and a realistic crossbar (labeled crossbar) whose delay is a function of the core count. The markers in the graph show cycle-accurate simulation results, whereas the lines correspond to the analytic model.

In general, the analytic model predicts performance with excellent accuracy up to 16 cores. At 32 and 64 cores, the actual performance diminishes on three of the workloads (Data Serving, Web Search, and SAT Solver) due to poor software scalability, an effect not captured by the model. Performance scales well with core count for the remaining three workloads, and our model shows good accuracy, even at high core counts.

[1] The specification of conventional and tiled processors is presented in Chapter 2.

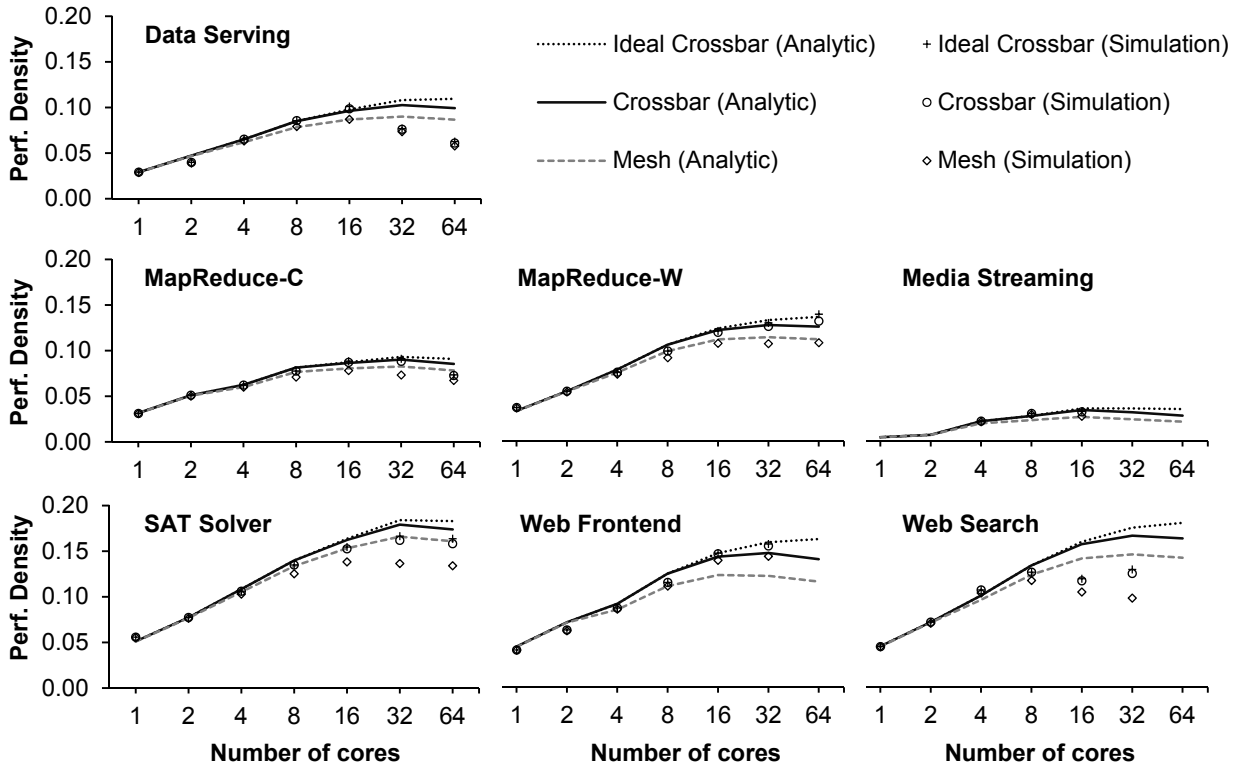


Figure 3.3: Cycle-accurate simulation and analytic results for designs with out-of-order cores and a 4MB LLC.

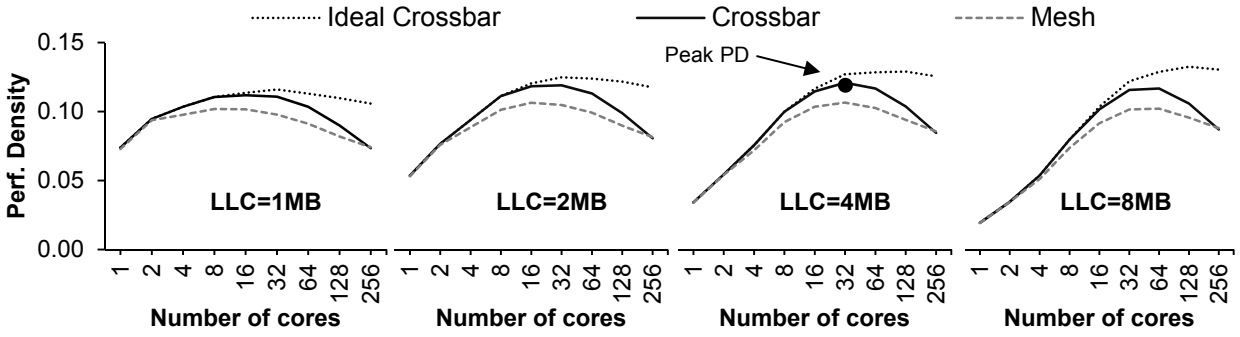


Figure 3.4: Performance density for a system with out-of-order cores and a range of last-level cache sizes.

3.4.2 Scale-Out Processors with Out-of-Order Cores

We begin the study with out-of-order cores in the 40nm technology. Figure 3.4 plots performance density, averaged across all workloads, for four different LLC sizes. We do not consider cache sizes above 8MB, as bigger caches do not provide performance improvements (see Section 2.1.3). Each graph consists of three lines, corresponding to one of the three interconnects

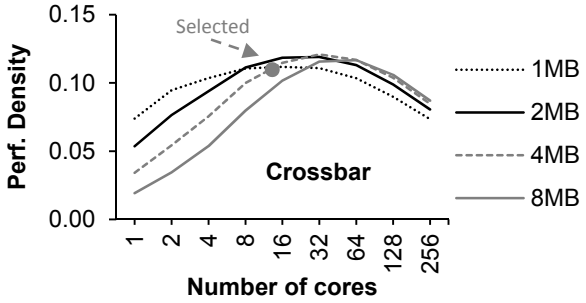


Figure 3.5: Performance density of pods (OoO) based on a crossbar interconnect and various LLC sizes.

described in Section 3.4.1.

In systems with a realistic interconnect (i.e., crossbar or mesh), performance density starts diminishing above 32 cores, regardless of cache capacity, indicating that the physical distance between the cores and the LLC hurts performance when integrating a large number of cores.

Performance density is maximized with 32 cores, a 4MB LLC, and a crossbar interconnect. However, the peak is almost flat. In such cases, software scalability bottlenecks, coherence complexity, and the difficulty of implementing a crossbar interconnect for a large number of cores are likely to shift the design toward a near-to-optimal pod with fewer cores.

To explore this trade-off, Figure 3.5 examines performance density of pods based on a crossbar interconnect across various LLC sizes. Among designs with fewer than 32 cores, a pod that integrates 16 cores and 4MB of LLC is within 5% of the true optimum. We, therefore, adopt the 16-core 4MB LLC design with a crossbar interconnect, as the preferred pod configuration due to its high PD at modest design complexity.

The PD-optimal pod occupies 92mm^2 and draws 20W of power for cores, caches, and the crossbar. Peak bandwidth demand is 9.4GB/s for 16 cores.

Chip-level assessment. Under the constraints specified in Section 2.4.1, a Scale-Out Processor can afford two pods before hitting the area limit. The resulting chip features 32 cores on a 263mm^2 die with a TDP of 62W.

3.4.3 Scale-Out Processors with In-Order Cores

Figure 3.6 illustrates performance density results, averaged across all workloads, for cache sizes ranging from 1 to 8MB and three different interconnects. The general trends are similar to those described in the previous section; however, simpler cores in a throughput-oriented architecture yield an optimal pod design with 32 cores, 2MB of LLC, and a crossbar interconnect. To mitigate the complexity associated with a large crossbar, pairs of cores can share a switch interface. Because the per-core performance is lower than in a design with out-of-order cores,

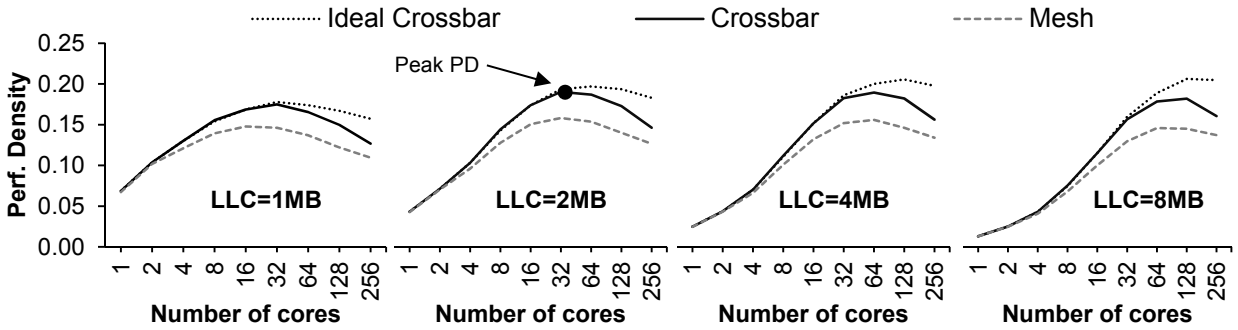


Figure 3.6: Performance density for a system with in-order cores and a range of last-level cache sizes.

we find the impact of switch sharing to be negligible.

The PD-optimal pod occupies 52mm^2 and draws 17W of power for cores, caches, and the crossbar. Peak bandwidth demand is 15GB/s for 32 cores.

Chip-level assessment. A Scale-Out Processor with in-order cores integrates three PD-optimal pods and is area-constrained. With memory interfaces and miscellaneous peripheral circuitry factored in, this configuration requires 270mm^2 of die area and a TDP of 91W.

3.4.4 Projection to 20nm Technology

To understand the effect of technology scaling on the Scale-Out Processor design, we project Scale-Out Processors with out-of-order and in-order cores to the 20nm technology node.

In a Scale-Out Processor based on out-of-order cores, seven pods can be integrated, for a total of 112 cores. The resulting configuration is area-limited, occupies 251mm^2, dissipates 83W of power at peak load, and achieves a performance density of 0.339, an improvement of 3.7x over the 40nm baseline. While ideal scaling predicts PD to improve by a factor of 4, the growth in area dedicated to the memory interfaces that do not benefit from technology scaling reduces the fraction of the die available to compute and dampens the gains in PD.

In an implementation based on in-order cores, the Scale-Out configuration is bandwidth-limited, assuming a constraint of six DDR4 memory controllers on a die. Compared to the 40nm baseline, the number of pods doubles to six on 192mm^2 die. Technology scaling improves performance density by a factor of 2.8, instead of the expected factor of 4, due to the large area overheads introduced by the on-die memory interfaces necessary to feed the Scale-Out chip.

The analysis above assumes the use of DDR4 memory interfaces. If DDR3 interfaces are used instead, Scale-Out designs using both core types, as well as tiled chips with in-order cores, will be bandwidth-limited at 20nm. The results corroborate prior work showing that highly-

Table 3.2: Performance density, area, power, and bandwidth requirements of various processor designs.

	40nm						
Processor design	PD	Cores	LLC (MB)	MCs	Die (mm^2)	Power (Watt)	Perf/ Watt
Conventional	0.026	6	12	2	276	94	0.08
Tiled (OoO)	0.060	20	20	1	245	50	0.29
LLC-Optimal Tiled (OoO)	0.084	32	8	2	251	56	0.38
LLC-Optimal Tiled with IR (OoO)	0.086	32	8	3	264	62	0.37
Scale-Out (OoO)	0.092	32	8	3	263	62	0.39
Tiled (IO)	0.099	64	20	2	251	67	0.37
LLC-Optimal Tiled (IO)	0.131	96	6	5	261	86	0.40
LLC-Optimal Tiled with IR (IO)	0.145	96	6	5	261	86	0.44
Scale-Out (IO)	0.155	96	6	6	270	91	0.46

	20nm						
Processor design	PD	Cores	LLC (MB)	MCs	Die (mm^2)	Power (Watt)	Perf/ Watt
Conventional	0.067	12	48	3	213	93	0.15
Tiled (OoO)	0.206	80	80	2	256	80	0.66
LLC-Optimal Tiled (OoO)	0.258	112	28	4	251	83	0.78
LLC-Optimal Tiled with IR (OoO)	0.294	112	28	4	251	83	0.89
Scale-Out (OoO)	0.339	112	28	4	251	83	1.03
Tiled (IO)	0.227	180	80	4	249	94	0.60
LLC-Optimal Tiled (IO)	0.360	224	12	6	202	86	0.84
LLC-Optimal Tiled with IR (IO)	0.362	192	12	6	192	80	0.87
Scale-Out (IO)	0.441	192	12	6	192	80	1.06

integrated chips based on simple cores will necessitate bandwidth-boosting techniques, such as 3D-stacked DRAM caches, to mitigate the memory bandwidth wall [42].

3.4.5 Summary

Table 3.2 summarizes chip-level features, power and bandwidth requirements, performance per Watt, and performance per mm^2 (i.e., performance density) for the various processor designs. Under an area-normalized comparison, processors with a higher performance density necessarily yield a higher performance. Conversely, for a given performance target, PD-optimized designs need a smaller die area compared to chips with a lower performance density.

The highest performance density is achieved in a Scale-Out Processor, which uses a pod-based organization to limit interconnect delays and maximizes compute area through modestly sized last-level caches. A Scale-Out design with out-of-order cores improves performance density by 3.5x and 1.5x over conventional and tiled chips, respectively, in 40nm technology (5.1x and 1.6x over the respective designs in 20nm). A Scale-Out design also achieves higher performance density over optimized tiled designs, such as *LLC-optimal tiled* and *LLC-optimal*

tiled with IR by 10% (31%) and 7% (15%), respectively, in 40nm (20nm) technology. Scale-Out Processors are 9% (7%) behind the ideal processor with the same LLC capacity and an ideal 4-cycle interconnect in 40nm (20nm) technology. It is noticeable that the performance density of the Scale-Out Processor becomes closer to the performance density of the ideal processor in the 20nm technology (as compared to 40nm technology) because, in this technology node, a larger fraction of the die area is used for memory channels that do not directly contribute to performance.

On workloads with laxer QoS requirements, higher performance density (and, consequently, higher throughput) can be achieved through the use of in-order cores. In such cases, a Scale-Out chip improves performance density by 6x (6.6x) and 1.6x (1.9x) over conventional and tiled designs, respectively, in 40nm (20nm) technology. Moreover, a Scale-Out organization achieves higher performance density over optimized tiled organizations, such as *LLC-optimal tiled* or *LLC-optimal tiled with IR* by 18% (22%) and 7% (22%), respectively, in 40nm (20nm) technology. The performance density of Scale-Out Processors is 7% (15%) behind that of the ideal processor when both processors are designed using in-order cores in 40nm (20nm) technology. The gap between the performance density of the Scale-Out Processor and the ideal processor increases in 20nm technology because in this technology node, both processors are bandwidth-limited, and the ideal processor can make better use of on-die caches. The results also underscore Scale-Out Processors' advantage under technology scaling, as both in-order and out-of-order Scale-Out configurations improve the lead in performance density over conventional and tiled chips as technology is scaled from 40 to 20nm.

Finally, we note that Scale-Out organizations are effective in improving processor energy efficiency, in addition to performance density. Compared to tiled organizations, performance per Watt is improved by 1.3x and 1.2x for out-of-order and in-order designs, respectively, in 40nm technology. The improvements extend to 1.6x and 1.8x at 20nm. Energy efficiency in Scale-Out chips is improved through higher per-core performance and lower energy/op. While core efficiency is the same for Scale-Out and tiled chips with the same core type, Scale-Out chips dissipate less energy in the memory hierarchy through smaller caches (less leakage) and smaller communication distances.

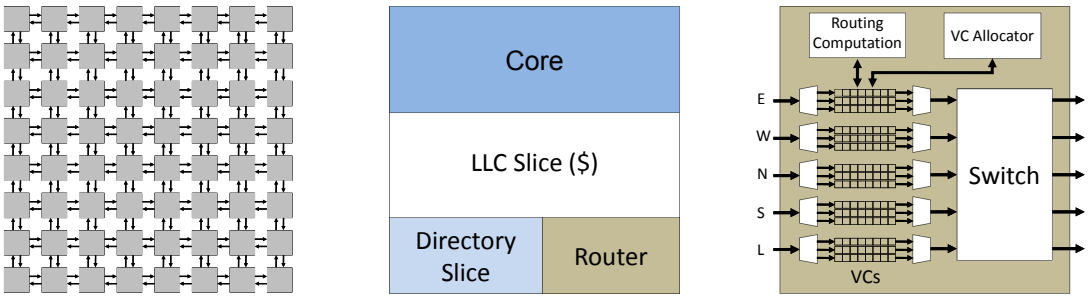
4 Microarchitecture of Scale-Out Processors

Taking advantage of common scale-out workload features, and driven by the need to increase server efficiency, in Chapter 3, we introduced Scale-Out Processors and formalized the Scale-Out Processor (SOP) design methodology. The SOP methodology calls for partitioning a chip into stand-alone modules named pods, wherein each pod is a server running an operating system and a full software stack. Moreover, the SOP methodology provides an optimization framework for deriving optimal core counts and LLC capacities in each pod based on microarchitectural and technology parameters. Additionally, each pod advocates many cores, modestly sized LLCs, and low interconnect delays.

With SOP methodology calling for having many pods in Scale-Out Processors, an open question is the following: how should the cores and the LLC in a pod be arranged and interconnected for maximum efficiency? While fast access to instructions and data is essential for the performance of scale-out workloads, instructions and data are frequently served by the last-level cache due to their large working sets. Consequently, providing a fast access path to the LLC is important when designing a pod for Scale-Out Processors. As the organization of the cores and the LLC and the choice of the interconnection network determine the access path between the individual cores and the LLC, we must optimize them for fast instruction and data delivery. Due to the fact that the organization of a pod and the choice of the interconnect depend on the number of cores in the pod, we discuss these issues for pods with few and many cores separately.

4.1 Pods with Few Cores

When few cores and cache banks need to be connected, a dancehall organization [82] with a crossbar interconnect for communication between the cores and cache banks is the right choice: the access latency of a crossbar interconnect is small, and its area overhead is negligible. Fortunately, designing a pod based on a crossbar is straightforward: there are many processors in the market that connect cores to cache banks via a crossbar in a dancehall fashion [81, 46, 2]. In the previous chapter, we showed that a pod with OoO cores features 16 cores and 4MB of



(a) Mesh-based pod with 64 tiles. (b) Tile organization. (c) Mesh router.

Figure 4.1: Elements of tiled pods.

LLC. For this pod, a crossbar can be used to connect cores and cache banks. The organization of this pod is similar to Niagara III [81], which has 16 cores and 6MB of cache; in addition, it provides connections between cores and cache banks using a crossbar. For the sake of brevity, we do not go into more details; instead, we will focus on pods with many cores.

4.2 Pods with Many Cores

The reliance of the dancehall architectures on crossbar interconnects makes such organizations unattractive for many-core pods due to the poor scalability of crossbars. To overcome the scalability limitations of crossbar-based designs, many-core pods can employ a tiled organization with a fully distributed last-level cache. Figure 4.1a shows an overview of a generic pod based on a tiled design. Each tile, pictured in Figure 4.1b, consists of a core, a slice of the distributed last-level cache, a directory slice, and a router. The tiles are linked via a routed, packet-based, multi-hop interconnect in a mesh topology.

The tiled organization and a structured interconnect fabric allow mesh-based designs to scale to large core counts. Unfortunately, the regularity of the mesh topology works to its disadvantage when it comes to performance scalability. Each hop in a mesh network involves the traversal of a multi-ported router, shown in Figure 4.1c, which adds delay due to the need to access the packet buffers, arbitrate for resources, and navigate the switch.

To overcome the performance drawbacks of mesh-based interconnects, researchers developed low-diameter topologies suitable for on-die implementation. These topologies use rich inter-node connectivity to bypass intermediate routers between a packet's source and destination nodes. A state-of-the-art, low-diameter topology is the flattened butterfly [55], shown in Figure 4.2. The flattened butterfly uses a set of dedicated channels to fully connect a given node to others along the row and column. The resulting network requires, at most, two hops (one in each of the X and Y dimensions) to deliver the packet to the destination. In doing so, the flattened butterfly greatly reduces the contribution of routers to the end-to-end delay, allowing performance to approach that of a crossbar interconnect.

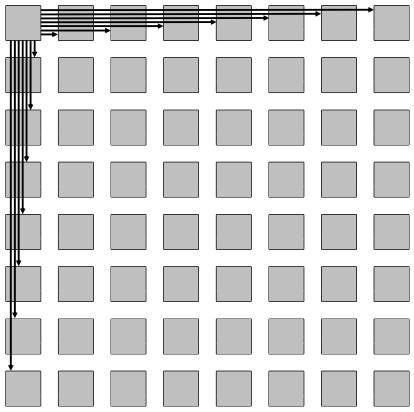


Figure 4.2: Flattened butterfly topology (links from only one node shown for clarity).

Problematically, the performance advantages of the flattened butterfly, or another richly connected Network-on-Chip (NOC), come at considerable area expense, stemming from the use of many-ported routers and a multitude of links. For instance, in the flattened butterfly in Figure 4.2, each router necessitates 14 network ports (7 in each of the two dimensions), plus a local port. The network ports are costly due to the presence of deep packet buffers necessary to cover the flight time of the long-range links. Meanwhile, the routers' internal switch fabric is area-intensive due the need to interconnect a large number of ports. Finally, links consume valuable on-die real estate due to the need for frequent repeater placement[1], even though wires themselves can be routed over tiles.

To summarize, existing NOC architectures require an uneasy choice between performance and area efficiency. Meanwhile, many-core pods in Scale-Out Processors demand both good performance and good area efficiency.

4.2.1 Memory Traffic in Scale-Out Workloads

In order to maximize the efficiency of many-core pods in Scale-Out Processors, we examine the memory traffic in scale-out workloads to identify opportunities for optimizations.

As noted earlier, scale-out workloads have large instruction footprints and vast datasets. Cores executing these workloads frequently access the LLC because neither the instructions nor the datasets fit in L1 caches. The large instruction footprints of scale-out workloads can be readily accommodated in the LLC, while the vast datasets dwarf the LLC capacity and reside in memory. Consequently, the majority of accesses to the instruction blocks hit in the LLC, while many dataset accesses miss and are filled from main memory.

On an L1 miss, the directory controller and the LLC check to determine if the block is available in the pod. If so, and if LLC's copy is the most recent, the LLC will service the miss and send

[1] Repeaters are necessary to overcome poor RC characteristics of wires in current and future technologies.

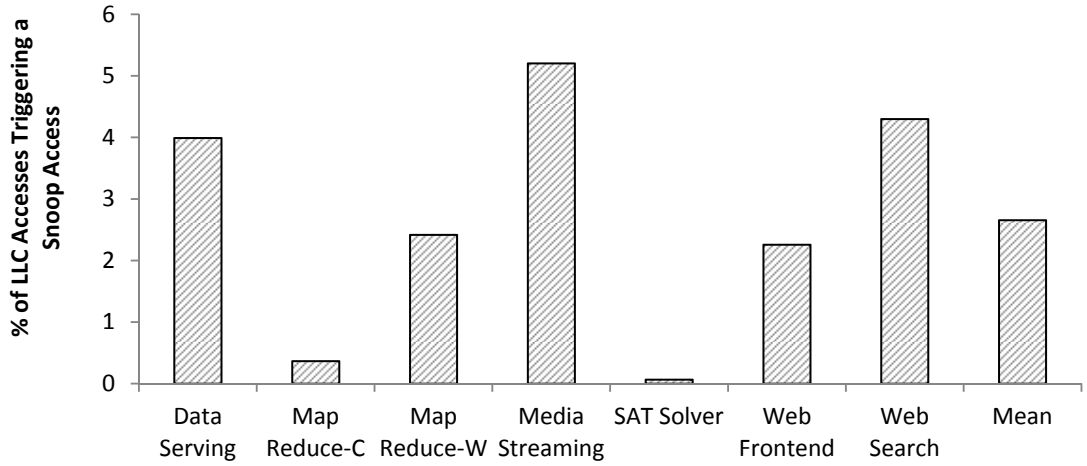


Figure 4.3: Percentage of LLC accesses causing a snoop message to be sent to a core.

the data to the requesting core. If the requesting core signals that it needs to modify the block, the directory will also send *snoop* messages to the set of sharers, instructing them to invalidate their copy. Conversely, if the directory indicates that another core has the block, it will send a snoop message to the appropriate core, instructing it to forward the block to the requester (L1-to-L1 forwarding). Finally, in the case of a miss, the LLC fetches the block from the main memory and returns it to the requesting core.

Importantly, core-to-core communication (i.e., L1-to-L1 forwarding) is triggered only as a result of data sharing at the L1 level. However, due to the high-level behavior of scale-out workloads, this type of data sharing is rare. Instructions are actively shared, but they are read-only and served from the LLC; dataset is vast, and the likelihood of two independent requests sharing a piece of data is low.

Figure 4.3 shows the fraction of accesses to the LLC that cause a snoop message to be sent to an L1 cache across seven scale-out workloads. As expected, coherence activity is negligible in these workloads, with an average of 2.7 out of 100 LLC accesses triggering a snoop. Earlier work made similar observations for both scale-out [30] and server [63] workloads.

The lack of coherence activity in scale-out workloads implies that the dominant traffic flow is from the cores to the LLC and back to the cores. We refer to this phenomenon as a core-to-cache *bilateral* access pattern. In tiled pods, the coupled nature of core and LLC slices means that accesses to the last-level cache from each individual core, over time, target all of the tiles, resulting in an all-to-all traffic pattern at the pod level. Achieving low latency under all-to-all traffic requires a richly connected topology, necessarily resulting in high area and wire cost.

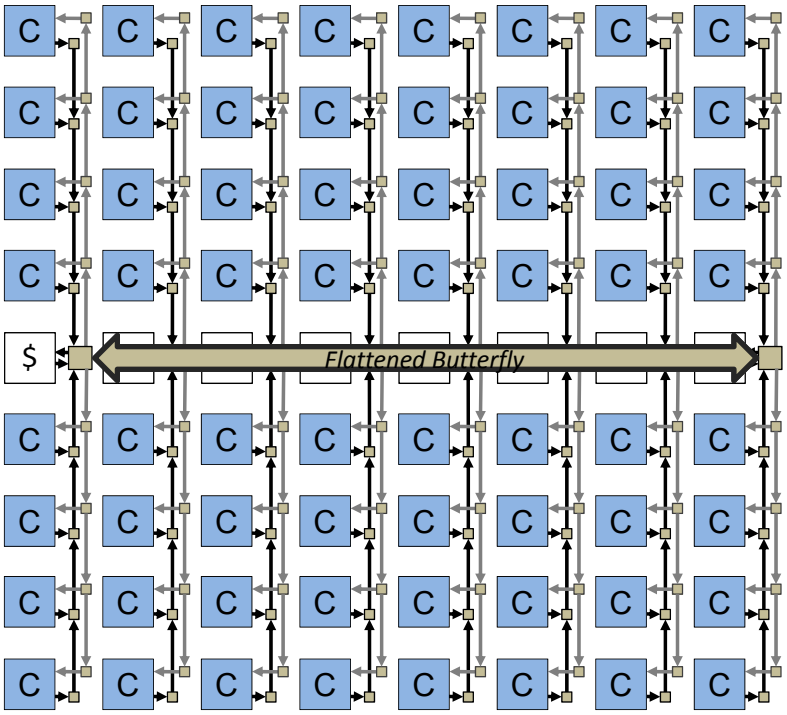
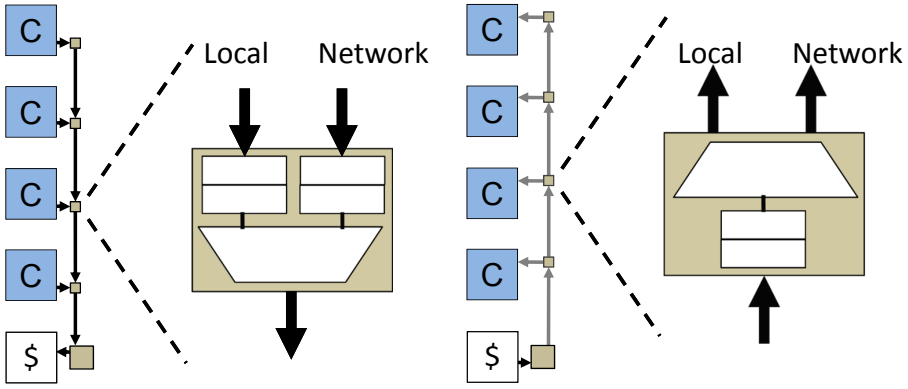


Figure 4.4: NOC-Out organization.

4.2.2 NOC-Out

NOC-Out is a many-core pod organization optimized for the bilateral access pattern dominant in scale-out workloads. NOC-Out leverages two insights to minimize interconnect delays at a small area footprint. First, NOC-Out segregates the LLC slices from the cores into separate cache-only tiles and concentrates the cache tiles in the center of the pod. The segregation of cores and the LLC breaks the all-to-all traffic pattern characteristic of tiled pods and establishes a bilateral traffic flow between core and cache regions. Second, NOC-Out takes advantage of the bilateral traffic to limit network connectivity, enabling a reduction in network cost. Specifically, NOC-Out eliminates the bulk of the core-to-core links and the supporting router structures, preserving a minimum degree of connectivity to enable each core to reach the LLC region.

Figure 4.4 shows a high-level view of the proposed organization, featuring LLC slices in the center of the pod and core tiles on both sides of the LLC. NOC-Out uses simple, routing-free *reduction trees* to guide packets toward the centralized cache banks, as well as *dispersion trees*, which are logical opposites of reduction trees, to propagate response data and snoop traffic out to the cores. Every reduction and dispersion tree connects a small number of cores to exactly one cache bank. The LLC banks are linked in a flattened butterfly topology, forming a low-latency Non-Uniform Cache Access (NUCA) cache. Notably, NOC-Out does not support direct core-to-core connectivity, requiring all traffic to flow through the LLC region.



(a) Reduction tree. (b) Dispersion tree.

Figure 4.5: Details of NOC-Out networks.

In the rest of the section, we detail the organization of the reduction, dispersion, and LLC networks.

Reduction Network

The reduction network is designed for a low-latency delivery of packets from the cores to the centralized cache banks. Figure 4.5a shows the key features of a reduction tree, which spans a column of cores and terminates at the LLC bank at the end of the column. Effectively, a reduction tree is a many-to-one interconnect, with all packets that enter a reduction tree flowing to the same destination cache bank. A node in the tree is a buffered, flow-controlled, two-input multiplexer that merges packets from the local port with those already in the network.

Compared to a conventional packet-based NOC, the reduction network does not require routing, as all packets flow to a common destination. The switch, typically implemented as a crossbar or a mux tree in conventional NOCs, is reduced to a simple two-input mux in a reduction tree. The reduction network is similar to conventional NOCs, in that it benefits from the use of virtual channels (VCs) for protocol deadlock avoidance; as such, it requires a virtual channel allocation mechanism. However, with just two ports (local and network), the VC allocator is trivially simple. In fact, given the low memory-level parallelism (MLP) of scale-out workloads [30], static-priority arbitration policies that always prioritize the network over the local port (or vice versa) tend to work well and afford further simplification of the arbitration logic.

NOC-Out distinguishes three message classes – data requests, snoop requests, and responses (both data and snoop) – to guarantee network-level deadlock freedom for its coherence protocol. Of these, only data requests and responses travel through the reduction trees, as snoop requests can only originate at the directory nodes at the LLC. As a result, each port in a

reduction tree has two virtual channels (one per message class).

Upon arrival at a router in a reduction tree, a packet is buffered in the appropriate VC (determined by the packet's message class). With a total of four VCs in a router (two ports with two VCs per port), a 4:1 arbiter selects a winning VC based on priority and downstream buffer availability. In this dissertation, we assume the following fixed priority ordering of VCs (highest to lowest): network responses, local responses, network requests, and local requests. By prioritizing the network over the local port, we seek to mitigate the latency disadvantage of cores that are more distant from the LLC. Because a reduction tree router has exactly one output port, routing and output port selection logic is unnecessary, and just one arbiter is required per node.

Dispersion Network

The dispersion network carries packets (data responses and snoop requests) from the LLC to the cores. Figure 4.5b shows a logical view of a dispersion tree. A dispersion tree is a logical opposite of the reduction tree, with a single source (a cache bank) and multiple destinations (cores). Each node in a tree is a buffered, flow-controlled demultiplexer that selects a local output port for packets that have reached their destination or propagates them farther up the tree toward the next node.

As is the case with the reduction network, virtual channels are necessary for deadlock avoidance to guarantee that snoop requests do not block data responses from reaching their destination. With two VCs per node (one per message class), on each clock cycle, simple control logic (1) uses message priority and buffer availability to select a winning VC; and (2) sets up demux control to forward a flit from the selected VC to the local or network output. Again, we use a static priority assignment to prioritize reply messages over snoop requests, which is subject to buffer availability.

LLC Network

As described above, NOC-Out segregates core and LLC slices[2] into separate tiles. Because each core connects to just one LLC tile through its reduction and dispersion trees, NOC-Out relies on a richly connected flattened butterfly network to route traffic between LLC tiles. The choice of the network is motivated by the need to minimize delay and reduce contention in the LLC region.

In order to reduce the area and channel expense of the flattened butterfly, NOC-Out takes advantage of the fact that the number of LLC tiles need not match the number of core tiles. The number of LLC tiles can be reduced because low instruction- and memory-level parallelism in scale-out workloads naturally dampen the bandwidth pressure on the LLC. Our empirical

[2] An LLC slice is composed of data, tags, and directory.

data shows that a design with four cores per one LLC bank achieves a level of performance that is within 2% of a system with an equal number of cores and LLC banks. Moreover, each LLC tile can house multiple LLC banks that share the router. A reduction in the number of the LLC tiles diminishes the cost and extent of the richly connected LLC network.

Additional Considerations

Before concluding the description of NOC-Out, we highlight several additional aspects of the proposed design; namely, its flow control architecture and support for shared memory.

Flow control: All three NOC-Out networks (reduction, dispersion, and LLC) rely on conventional virtual channel credit-based flow control. The amount of buffering per port in both reduction and dispersion trees is insignificant (a few flits per VC) thanks to a short round-trip credit time resulting from a trivial pipeline. The flattened butterfly LLC network requires more buffering per port to cover the multi-cycle delays of long-range links and multi-stage routers; however, this cost is restricted to just a fraction of the nodes.

Shared memory: Shared memory is a prominent feature of today's software stacks. Despite being optimized for the bilateral core-to-cache communication, NOC-Out fully supports the shared memory paradigm through conventional hardware coherence mechanisms, preserving full compatibility with existing software. What NOC-Out sacrifices by eliminating direct core-to-core connectivity is the support for locality-optimized communication. Instead, NOC-Out optimizes for cost and performance on scale-out server workloads that do not benefit from locality optimizations.

4.3 Methodology

Table 4.1 summarizes the key elements of our methodology, with the following sections detailing the specifics of the evaluated designs, technology parameters, workloads, and simulation infrastructure.

4.3.1 Pod Parameters

Our target is a many-core pod implemented in 32nm technology. This pod features 64 cores, 8MB of last-level cache, and four DDR3-1667 memory channels. Core microarchitecture is modeled after an ARM Cortex-A15, a three-way out-of-order design with 32KB L1-I and L1-D caches. The cache line size is 64B.

We evaluate various pod organizations with a 64-core configuration and a relatively large core (i.e., Cortex-A15) because this configuration represents a worst-case scenario for pods with regard to the number of cores and the physical distance between the cores and the LLC. The chosen LLC capacity (i.e., 8MB), based on the results presented in Chapter 3, maximizes the

Table 4.1: Evaluation parameters.

Parameter	Value
Technology	32nm, 0.9V, 2GHz
Pod features	64 cores, 8MB NUCA LLC, 4 DDR3-1667 memory channels
Core	ARM Cortex-A15-like: 3-way out-of-order, 60-entry ROB, 16-entry LSQ, 2.9mm^2
Cache	per MB: 3.2mm^2
NOC Organizations:	
Mesh	Router: 5 ports, 3 VCs/port, 5 flits/VC, 2-stage speculative pipeline. Link: 1 cycle
Flattened Butterfly	Router: 15 ports, 3 VCs/port, variable flits/VC, 3 stage pipeline. Link: up to 2 tiles per cycle
NOC-Out	Reduction/Dispersion networks: 2 ports/router, 2 VCs/port, 1 cycle/hop (inc. link) LLC network: flattened butterfly

performance density of the selected 64-core configuration.

We consider three pod organizations, as follows:

Mesh: Our baseline for the evaluation is a mesh-based tiled pod, as shown in Figure 4.1. The 64 tiles are organized as an 8-by-8 grid, with each tile containing a core, a slice of the LLC, and a directory node.

At the network level, a mesh hop consists of a single-cycle link traversal followed by a two-stage router pipeline for a total of three cycles per hop at zero load. The router performs routing, VC allocation, and speculative crossbar (XB) allocation in the first cycle, followed by XB traversal in the next cycle. Each router port has three VCs to guarantee deadlock freedom across three message classes: data requests, snoop requests, and responses. Each VC is five flits deep, which is the minimum necessary to cover the round-trip credit time.

Flattened Butterfly (FBfly): The FBfly-based pod has the same tiled organization as the mesh baseline, but it benefits from the rich connectivity afforded by the flattened butterfly organization, as shown in Figure 4.2. Each FBfly router has 14 network ports (seven per dimension) plus a local port. Due to high arbitration complexity, the router does not employ speculation, resulting in a three-stage pipeline. Each router port has three VCs to guarantee deadlock freedom. The number of flit buffers per VC is optimized based on the location of the router in the network to minimize buffer requirements. Finally, the link delay is proportional to the distance spanned by the link. Given our technology parameters (detailed below) and tile dimensions, a flit in the channel can cover up to two tiles in a single clock cycle.

NOC-Out: Our proposed pod organization, described in Section 4.2.2, segregates core and LLC tiles, as well as localizes the LLC in the center of the pod. To connect cores to the LLC, NOC-Out uses specialized reduction and dispersion networks. Direct inter-core connectivity is not supported, and all traffic must flow through the LLC region.

Both the reduction and dispersion networks require just two VCs per port. In the reduction network, only data requests and responses flow from the cores to the cache, as snoop requests cannot originate at the core tiles. Similarly, the dispersion network only needs to segregate snoop requests and data responses, as data requests cannot originate at the LLC. In the absence of contention, both networks have a single-cycle per-hop delay, which includes traversal of both the link and the arbitrated mux (in the reduction tree) or demux (in the dispersion tree). This delay is derived based on the technology parameters and tile dimensions.

The LLC is organized as a single row of tiles, with each tile containing 1MB of cache and a directory slice. The aspect ratio of the LLC tiles roughly matches that of the core tiles, allowing for a regular layout across the pod, as shown in Figure 4.4. LLC tiles are internally banked to maximize throughput. For the evaluation, we model two banks per tile (16 LLC banks, in total), as our simulations show that this configuration achieves similar throughput at lower area cost as compared to designs with higher degrees of banking. The eight LLC tiles are fully connected via a one-dimensional flattened butterfly. LLC routers feature a 3-stage non-speculative pipeline, with three VCs per input port.

4.3.2 Technology Parameters

We use publicly available tools [49, 72] and data [4, 47] to estimate the area and energy of various pod organizations. Our study targets a 32nm technology node with an on-die voltage of 0.9V and a 2GHz operating frequency.

We use custom wire models, derived from a combination of sources [4, 47], to model links and router switch fabrics. For links, we model semi-global wires with a pitch of 200nm and power-delay-optimized repeaters that yield a link latency of 125ps/mm. On random data, links dissipate 50fJ/bit/mm, with repeaters responsible for 19% of link energy. For area estimates, we assume that link wires are routed over logic or SRAM and do not contribute to network area; however, repeater area is accounted for in the evaluation.

Our buffer models are taken from ORION 2.0 [49]. We model flip-flop-based buffers for mesh and NOC-Out, as both have relatively few buffers per port. For the flattened butterfly, we assume SRAM buffers that are more area- and energy-efficient than flip-flops for large buffer configurations.

Cache area, energy, and delay parameters are derived via CACTI 6.5 [72]. A 1MB slice of the LLC has an area of 3.2mm^2. Finally, parameters for the ARM Cortex-A15 core are borrowed from Microprocessor Report and scaled down from the 40nm technology node to the 32nm target. Core area, including L1 caches, is estimated at 2.9mm^2. Core features include three-way decode/issue/commit, 60-entry ROB, and 16-entry LSQ.

4.3.3 Workloads

We use scale-out workloads from CloudSuite [1]. The workloads include Data Serving, MapReduce, Media Streaming, Web Frontend, SAT Solver, and Web Search. We consider two MapReduce workloads – text classification (MapReduce-C) and word count (MapReduce-W). For the Web Frontend workload, we use the e-banking option from SPECweb2009 in place of its open-source counterpart from CloudSuite, as SPECweb2009 exhibits better performance scalability at high core counts. Two of the workloads – SAT Solver and MapReduce – are batch, while the rest are latency-sensitive and are tuned to meet the response time objectives.

Four out of seven workloads scale to 64 cores. The other three – namely Media Streaming, Web Serving, and Web Search – only scale to 16 cores due to various software bottlenecks. For these three workloads, we choose the 16 tiles in the center of the pod for the mesh and flattened butterfly designs and the 16 core tiles adjacent to the LLC in the NOC-Out design.

4.3.4 Simulation Infrastructure

We estimate the performance of the various pod configurations using Flexus full-system simulation [93, 89]. Flexus extends the Virtutech Simics functional simulator with timing models of cores, caches, on-chip protocol controllers, and interconnect. Flexus models the SPARC v9 ISA and is able to run unmodified operating systems and applications.

We use the SimFlex multiprocessor sampling methodology [89]. Our samples are drawn over an interval of 10 seconds (30 seconds for Media Streaming) of simulated time. For each measurement, we launch simulations from checkpoints with warmed caches and branch predictors, and then we run 100K cycles (2M cycles for Data Serving) to achieve a steady state of detailed cycle-accurate simulation before collecting measurements for the subsequent 50K cycles. We use the ratio of the number of committed application instructions to the total number of cycles (including the cycles spent executing operating system code) to measure performance; this metric has been shown to accurately reflect overall system throughput [89]. Performance measurements are computed with 95% confidence with an average error of less than 4%.

4.4 Evaluation

We first examine performance and area efficiency of mesh, flattened butterfly, and NOC-Out designs, given a fixed 128-bit link bandwidth. We then present an area-normalized performance comparison, followed by a discussion of power trends.

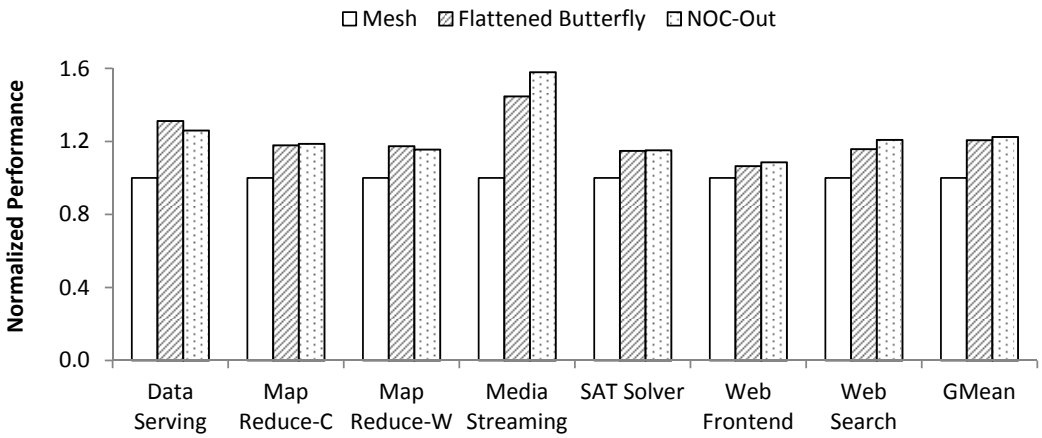


Figure 4.6: System performance, normalized to a mesh-based design.

4.4.1 System Performance

Figure 4.6 shows full-system performance of a single pod, normalized to the mesh, under the various NOC organizations. Compared to the mesh, the richly connected flattened butterfly topology improves performance by 7-45%, with a geomean of 21%. The highest performance gain is registered on the Media Streaming workload, which is characterized by extremely low ILP and MLP, making it particularly sensitive to the LLC access latency.

On average, the proposed NOC-Out design matches the performance of the flattened butterfly. On Data Serving, bank contention is responsible for a small performance degradation in NOC-Out, resulting in lower performance as compared to the flattened butterfly. On the other hand, on Media Streaming and Web Search (both 16-core workloads), NOC-Out benefits from a smaller average communication distance between the cores and the LLC, which translates into higher performance. The bottom line is that NOC-Out improves performance by 22% over the mesh and, on average, matches the performance of the flattened butterfly.

We conclude the performance assessment by noting that while the bisection bandwidths of the various topologies are different, the networks are not congested. Differences in latency, not bandwidth, across the topologies are responsible for the performance variations.

4.4.2 NOC Area

Figure 4.7 breaks down the NOC area of the three organizations by links, buffers, and crossbars. Only repeaters are accounted for in link area, as wires are assumed to be routed over tiles.

At over 23mm^2, the flattened butterfly has the highest NOC area, exceeding that of the mesh by nearly a factor of seven. The large footprint of the flattened butterfly is due to its large link budget and the use of buffer-intensive, many-ported routers.

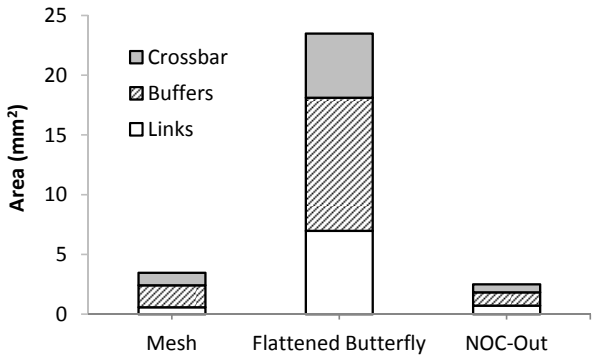


Figure 4.7: NOC area breakdown.

NOC-Out's interconnect footprint of 2.5mm^2 is the lowest among the evaluated pod organizations, requiring 28% less area than a mesh and about 10 times less area than a flattened butterfly. NOC-Out's area advantage stems from minimal connectivity among the majority of the nodes (i.e., cores) and from the use of low-complexity network trees (reduction and dispersion) that minimize router costs. Each of the two tree networks contributes just 18% to the total NOC footprint. In contrast, the flattened butterfly interconnecting NOC-Out's LLC region constitutes 64% of the total network area while linking just 11% of the tiles.

4.4.3 Area-Normalized Comparison

The performance and area analysis in the previous two sections assumed a fixed link width of 128 bits, resulting in vastly different NOC area costs and bisection bandwidths. To better understand how various pod organizations compare given a fixed NOC budget, we assess the performance of the mesh and flattened butterfly using NOC-Out's area of 2.5mm^2 as a limiting constraint. We reduce the width of both mesh and flattened butterfly NOCs until each of their respective areas (links + routers) equals that of NOC-Out and then measure the performance of the resulting designs.

Figure 4.8 summarizes the results of the study, with the performance of the three organizations normalized to that of the mesh. Given a smaller area budget, the performance of both mesh and flattened butterfly degrades. The degradation is small in the mesh network, as the increase in the serialization latency continues to be dwarfed by the header delay. In contrast, the richly connected flattened butterfly sees its link bandwidth shrinks by a factor of seven, significantly impacting end-to-end latency through a spike in the serialization delay. Compared to the flattened butterfly at the same area budget, NOC-Out enjoys a 75% performance advantage. Compared to the mesh, NOC-Out's performance edge is 24%.

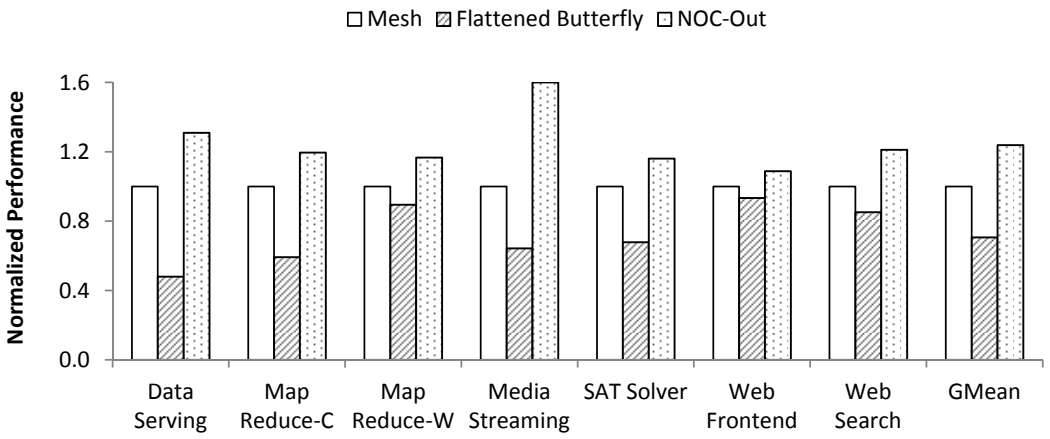


Figure 4.8: System performance, normalized to a mesh-based design, under a fixed NOC area budget.

4.4.4 Power Analysis

Our analysis shows that the NOC is not a significant consumer of power. For all three organizations, NOC power is below 2W. In contrast, cores alone consume in excess of 60W. Low ILP and MLP of scale-out workloads are the main reasons for the low power consumption at the NOC level. Another factor is the near-absence of snoop traffic in these workloads.

NOC-Out results in the most energy-efficient NOC design, dissipating 1.3W of power, on average. Mesh and flattened butterfly average 1.8W and 1.6W, respectively. In all organizations, most of the energy is dissipated in the links. NOC-Out's higher efficiency stems from the lower average distance between the cores and the LLC, resulting in less energy spent in the wires. Meanwhile, the flattened butterfly's rich connectivity gives it an advantage over the mesh (1.6W against 1.8W).

4.4.5 Summary

The evaluation results show that NOC-Out offers the performance of the richly connected flattened butterfly topology at a fraction of the network area. Whereas the flattened butterfly requires a prohibitive 23mm^2 of die real estate, NOC-Out necessitates just 2.5mm^2 for the interconnect. When constrained to NOC-Out's area budget, the performance of the flattened butterfly diminishes, giving NOC-Out a 75% performance advantage. In comparison to a mesh, NOC-Out improves performance by 22% and reduces the network area footprint by 28%.

4.5 Discussion

4.5.1 Scalability of NOC-Out

So far, our description and evaluation of NOC-Out has been in the context of a 64-core pod. NOC-Out can be readily scaled to support larger numbers of cores through the use of concentration and, in configurations featuring hundreds of cores, through judicious use of express channels in reduction and dispersion networks. If necessary, the LLC network can be scaled up by extending its flattened butterfly interconnect from one to two dimensions. We now briefly discuss each of these options.

Concentration: Concentration can be used to reduce the network diameter by aggregating multiple terminals (e.g., cores) at each router node [4]. In the case of reduction and dispersion networks, a factor of two concentration at each node (i.e., two adjacent cores sharing a local port of the mux/demux) could be used to support twice the number of cores of the baseline design at nearly the same network area cost. With four times more nodes in the network and a concentration factor of four, we find that the 16B links in the tree networks are bottlenecked by insufficient bandwidth, necessitating either additional or wider links.

Express links: In pods with hundreds of cores, the height of the reduction and dispersion trees may become a concern from a performance perspective. To mitigate the tree delay, express links can be judiciously inserted into the tree to bypass some number of intermediate nodes, allowing performance to approach that of an "ideal" wire-only network. While express links increase the cost of the network due to greater channel expense, they are compatible with the simple node architectures described in Section 4.2.2 and do not necessitate the use of complex routers.

Flattened butterfly in LLC: When executing scale-out workloads, much of the useful LLC content is the instruction footprint and OS data. Because this content is highly amenable to sharing by all of the cores executing the same binary, a pod with a higher core count does not mandate additional LLC capacity [65]. Should the need arise, however, to expand the LLC beyond a single row of tiles, the flattened butterfly network interconnecting the tiles can be readily scaled from one to two dimensions. While an expanded flattened butterfly increases the cost of NOC-Out, the expense is confined to the fraction of the pod occupied by the LLC.

5 Scale-Out Processors with Large Dies

Scale-Out Processors are designed using Scale-Out Processor (SOP) design methodology and can efficiently execute scale-out workloads. The SOP methodology uses the metric of performance density to form throughput-optimal building blocks (named pods) and replicates them on a chip to the available area, bandwidth, and power budgets. While single- and multi-pod Scale-Out Processors have the same performance density, an open question is why multi-pod Scale-Out Processors are preferable for the execution of scale-out workloads. To answer this question, we study and evaluate the efficiency of various datacenters based on single- and multi-pod Scale-Out Processors.

5.1 Motivation

Datacenters are the workhorses powering the information revolution. Companies leading the transformation to the digital universe – such as Google, Microsoft, and Facebook – rely on networks of datacenters to provide search capabilities, social connectivity, media streaming, and a growing number of other offerings to large, distributed audiences. A scale-out datacenter houses tens of thousands of servers, necessary for high scalability, availability, and resilience [9, 6].

The massive scale of datacenters requires an enormous capital outlay for infrastructure and hardware, often exceeding $100 million per datacenter [80]. Similarly expansive are the power requirements, typically in the range of 5-15MW per datacenter, totaling millions of dollars in annual operating costs. With demand for information services skyrocketing around the globe, efficiency has become a paramount concern in the design and operation of large-scale datacenters.

In order to reduce infrastructure, hardware, and energy costs, datacenter operators target high compute density and energy efficiency. Total Cost of Ownership (TCO) is an optimization metric that considers the costs of real estate, power delivery and cooling infrastructure, hardware acquisition costs, and operating expenses. Because server acquisition and power costs

Table 5.1: Server chip characteristics.

Processor	Cores	LLC size (MB)	DDR3 interfaces	Power (W)	Area (mm^2)	Cost ($)
Conventional	6	12	2	94	276	800
Tiled (OoO)	20	20	1	50	245	370
1Pod (OoO)	16	4	2	36	158	320
Scale-Out (OoO)	32	8	3	62	263	370
Tiled (In-order)	64	20	2	67	251	370
1Pod (In-order)	32	2	2	34	118	320
Scale-Out (In-order)	96	6	6	91	270	370

constitute the two largest TCO components [38], servers present a prime optimization target in the quest for more efficient datacenters. In addition to cost, performance is also of paramount importance in scale-out datacenters designed to serve thousands of concurrent requests with real-time constraints. The ratio of performance to TCO (performance per dollar of ownership expense) is, thus, an appropriate metric for evaluating different datacenter designs.

Table 5.1 summarizes principal characteristics of various server processors. In addition to single- and multi-pod Scale-Out Processors, in this study, we also include server processors representing existing commercial processors (i.e., conventional and tiled) to evaluate the benefits of using Scale-Out Processors in existing datacenters. All processors in this study are designed at the 40nm technology and are taken from Table 3.2.

5.2 Methodology

We now describe the cost models and the experimental setup used in evaluating the various processors at datacenters.

5.2.1 TCO Model

Large-scale datacenters employ high-density server racks to reduce the space footprint and improve cost efficiency. A standard rack can accommodate up to 42 *1U* servers, where each server integrates one or more processors, multiple DRAM DIMMs, disk- or flash-based storage nodes, and a network interface. Servers in a rack share the power distribution infrastructure and network interfaces to the rest of the datacenter. The number of racks in a large-scale datacenter is commonly constrained by the available power budget.

Our TCO analysis, derived using EETCO [44], considers four major expense categories summarized below. Table 5.2 further details key parameters.

Datacenter infrastructure: Includes land, building, power provisioning and cooling equipment with a 15-year depreciation schedule. Datacenter area is primarily determined by the

Table 5.2: TCO parameters.

Parameter	Value
Rack dimensions (42U): width x depth x inter-rack space	0.6m x 1.2m x 1.2m
Infrastructure cost	\$3000/m^2
Cooling and power provisioning equipment cost	\$12.5/Watt
Cooling and power provisioning equipment space overhead	20%
SPUE (fan and power supply efficiency factor)	1.3
PUE	1.3
Personnel cost	\$200 per rack/month
Networking gear	360W, \$10,000 per rack
Motherboard	25W, \$330 per 1U
Disk	10W, \$180, 100-year MTTF
DRAM	1W, \$25, 800-year MTTF per GB
Processor	30-year MTTF

IT (rack) area, with cooling and power provisioning equipment factored in. The cost of this equipment is estimated per Watt of critical power.

Server and networking hardware: Server hardware includes processors, memory, disks, and motherboards. We also account for the networking gear at the edge, aggregation, and core layers of the datacenter, and we assume that the cost scales with the number of racks. The amortization schedule is three years for server hardware and four years for networking equipment.

Power: Predominantly determined by the servers, including fans and power supply, networking gear, and cooling equipment. The electricity cost is \$0.07/KWh.

Maintenance: Includes costs for repairing faulty equipment, determined by its mean-time-to-failure (MTTF), and the salaries of the personnel.

5.2.2 Processor Price Estimation

We evaluate a number of datacenter server processor designs, as summarized in Table 5.1. The price for the conventional processor is estimated by picking the lowest price (\$800) among online vendors for Xeon 5670 processor, which has the same number of cores and cache capacity as our conventional processor. To estimate the price for the rest of the processors, we use Cadence InCyte Chip Estimation tool (enterprise edition) modeling a production volume of 200,000 units (reverse-engineered from a commercial processor, i.e., Tilera Gx-3036, selling price) and a margin of 50%. We use this production volume to estimate the selling price for each processor type, taking into account non-recurring engineering (NRE) costs, mask

and production costs, yield, other expense categories, and a 50% profit margin. We find that despite nearly doubling the die size, the price per chip for tiled and Scale-Out Processors increases by just 15% (around $50) over the *1pod* processor because non-recurring engineering and mask costs dominate. While the above estimates are used for the majority of the studies, we also consider the sensitivity of different designs to processor price in Section 5.3.3.

5.2.3 Experimental Setup

For all experiments, we assume a fixed datacenter power budget of 20MW and a power limit of 17kW per rack. We evaluated lower-density racks rated at 6.6kW, but we found the trends to be identical across the two rack configurations. As such, we present one set of results.

To compare the performance and TCO of different server architectures, we start with a rack power budget and subtract all power costs at both the rack and the board levels, excluding the processors. The per-rack costs include network gear, cooling (fans), and power conversion. At the 1U server level, we account for the motherboard, two disks, and memory (model parameter) power. The remaining power budget is divided by the peak power of each evaluated processor chip to determine the number of processors per server. Datacenter performance is then estimated based on the number of processors in each 1U server (using the per-processor performance data collected in simulation), the number of servers in a rack, and the number of racks in the datacenter. As the performance of a datacenter is estimated when scale-out workloads are running, the conclusions of our studies are applicable to datacenters or parts of datacenters that run such workloads.

Finally, we make no assumptions on what the optimal amount of memory per server is, which in practice varies for different workloads, and model servers with 32, 64, and 128GB of memory per 1U. One simplifying assumption that we do make is that the amount of memory per 1U is independent of the chip design. Underlying this assumption are the observations that (a) the data is predominantly read-only; and (b) the data is partitioned for high parallelism, allowing performance to scale with more cores and sockets until bottlenecked by the bandwidth of the memory interfaces. Bandwidth limitations are accounted for in our studies.

5.3 Evaluation

5.3.1 Performance and TCO

We first compare datacenter performance and TCO for various processor designs assuming 64GB of memory per 1U server. The results are presented in Figures 5.1 and 5.2.

In general, we observe significant disparity in datacenter performance across the processor range stemming from the different capabilities and energy profiles of the various processor architectures. Highly integrated processors based on in-order cores – namely tiled, *1pod* and Scale-Out Processors – deliver the highest performance at the datacenter level. The

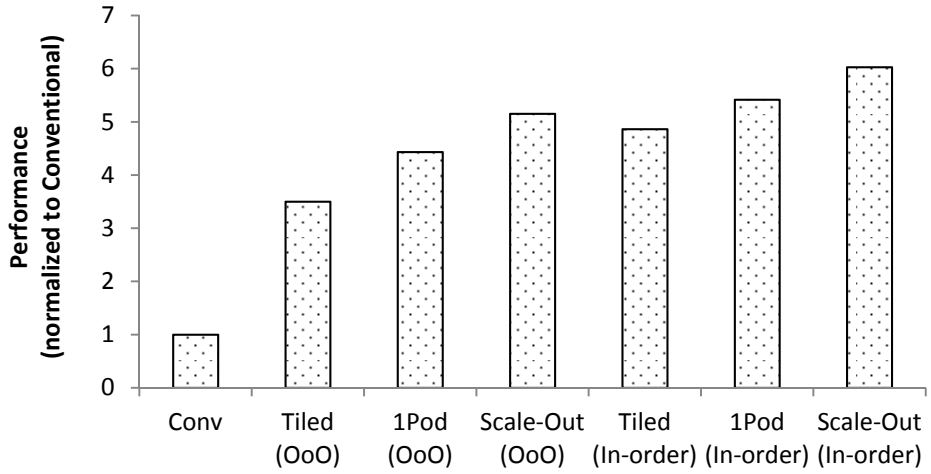


Figure 5.1: Datacenter performance for various server processors normalized to a design based on a conventional processor.

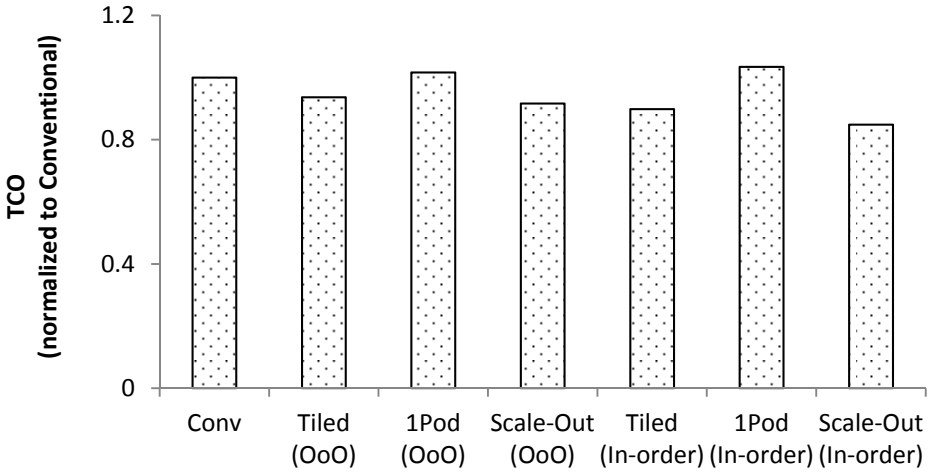


Figure 5.2: Datacenter TCO for various server processors normalized to a design based on a conventional processor.

1pod processor improves aggregate performance by a factor of 4.4 over conventional and 1.3 over tiled processors. The *1pod* processor is superior due to a combination of efficient core microarchitectures and high chip-level integration – attributes that help amortize the power of both on-chip and server-level resources among many cores, affording more power for compute resources.

The highest performance is delivered by the Scale-Out Processor with in-order cores, a design with the highest performance density that improves datacenter performance by an additional 11% over the *1pod* processor. The Scale-Out design effectively translates its performance density advantage into a performance advantage by better amortizing fixed power overheads

at the chip level among its many cores, ultimately affording more power for the execution resources at the rack level.

The Scale-Out Processor design based on out-of-order cores sacrifices 15% of the throughput at the datacenter level, as compared to the in-order design. However, higher core complexity is justified for workloads that demand tight latency guarantees and have a non-trivial computational component. Even with higher-complexity cores, the Scale-Out Processor attains better datacenter performance than either the conventional or tiled alternatives.

The differences in TCO among the different designs are not as pronounced as differences in performance, owing to the fact that processors contribute only a fraction to the overall datacenter acquisition and power budget. Nonetheless, one important trend worth highlighting is that while *1pod* designs based on out-of-order cores are less expensive and more energy efficient (by factors of 2.5 and 4.3, respectively) than conventional processors on a per-unit basis, at the datacenter level, a *1pod* design has a 2% higher TCO. The reason for the apparent paradox is that the low-power consumption of the *1pod* design necessitates as many as five sockets (versus two for conventional) per 1U server in order to saturate the available power budget. The acquisition costs of such a large number of processors negate the differences in unit price and energy efficiency, emphasizing the need to consider total cost of ownership in assessing datacenter efficiency.

5.3.2 Relative Efficiency

We next examine the combined effects of performance, energy efficiency, and TCO by assessing the various designs on datacenter performance/TCO and performance/Watt. Figures 5.3 and 5.4 present the results for performance/TCO and performance/Watt, respectively, as memory capacity is varied from 32 to 128GB per 1U server.

With 64GB of memory per 1U server, the following trends can be observed:

- The tiled processor with out-of-order cores improves performance/Watt by 3.5x over the conventional processor, and its performance/TCO advantage is 3.7x.
- A datacenter based on the *1pod* processor with out-of-order cores improves performance per TCO by a factor of 4.4 over conventional and 1.2 over tiled processors. Energy efficiency is improved by 4.4x and 1.3x, respectively, underscoring the combined benefits of aggressive integration and the use of an efficient core microarchitecture.
- The Scale-Out Processor with out-of-order cores further improves performance/TCO by 29% and performance/Watt by 16% over the *1pod* processor through a more efficient use of the die real estate.
- The Scale-Out Processor with in-order cores achieves 26% higher performance/TCO than the design based on out-of-order cores. The less aggressive in-order microarchitecture is responsible for the higher energy and area efficiencies of each core, resulting in higher

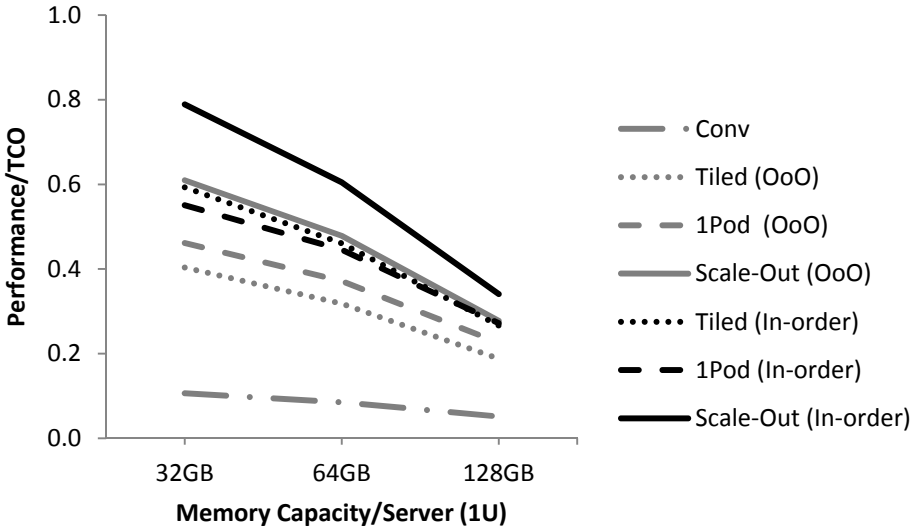


Figure 5.3: Datacenter performance/TCO for different server chip designs. Data not normalized.

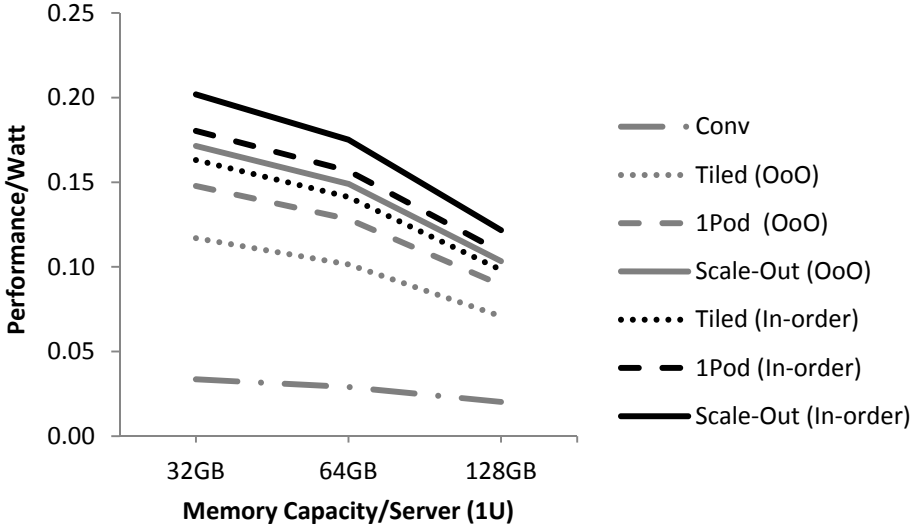


Figure 5.4: Datacenter performance/Watt for different server chip designs. Data not normalized.

throughput at the chip level. When the TCO premium is justified, which may be the case for throughput workloads (e.g., classification), the in-order Scale-Out Processor offers a 7.1x performance/TCO advantage (6x in performance/Watt) over the conventional processor.

While the discussion above focuses on servers with 64GB of memory, the trends are similar with other memory configurations. In general, servers with more memory lower the performance-

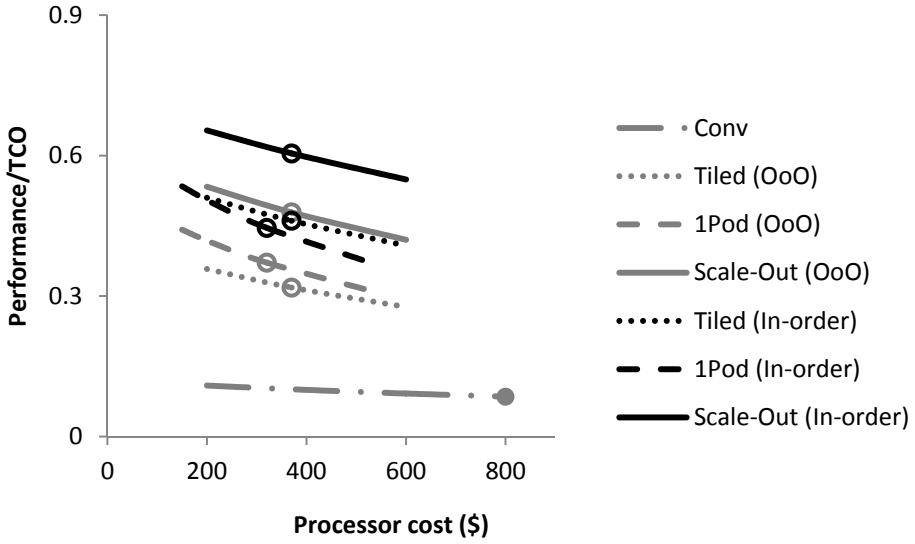


Figure 5.5: Relationship between the price per processor and TCO. Solid circles indicate known market prices; unfilled circles show estimated prices based on a production volume of 200K units.

to-TCO ratio, as memory adds to server cost while diminishing the processor power budget. The opposite is true for servers with less memory, in which the choice of processor has a greater effect on both cost and performance.

5.3.3 Sensitivity to Processor Price

Figure 5.5 shows the effect of varying the processor price on the relative efficiency (performance/TCO) of various processors assuming 64GB of memory per 1U server. For each processor type, we assume an ASIC design in 40nm technology and compute the price as a function of market size, ranging from 40K to 1M units, as described in Section 5.2.

In general, we observe that the price of larger chips has less impact on datacenter TCO as compared to that of smaller chips, since it takes fewer large chips to populate a server due to power constraints. In contrast, the *1pod* design, which has the smallest die area, is more sensitive to unit price due to the sheer volume of chips required per 1U server. For instance, there is a factor of 2.5 difference in the number of chips per server between conventional and *1pod* designs.

A consistent trend in our study is that, from a TCO perspective, processors with larger die area are preferred to smaller ones, as seen in the curves for the various Scale-Out designs. While the additional die area adds expense, the price difference is modest (around 16% or $50 per chip), as NRE and design costs dominate production costs. Furthermore, the increased cost is offset by the reduction in the number of required processors.

6 Scale-Out Processors in the Post-Moore Era

As scaling down transistor dimensions becomes more complicated and challenging [10], the validity of Moore's law, which is the primary driving force behind the growth of the semiconductor industry, is expected to end [66]. Three-dimensional integration of multiple logic dies is a propitious mechanism that can extend the validity of Moore's law. In a 3D integration, multiple logic dies are stacked on top of each other and interconnected by through-silicon vias (TSVs). TSVs are vertical connections passing through the stacked dies to provide connectivity among them. TSVs are high-performance connections because of their high density, as well as their short length.

In a conventional 2D integration, the scaling of transistor dimensions implies more transistors and also larger average distances between the transistors. As technology scales and more transistors become available, the average distance between the transistors increases because it is impossible to scale global wire length with technology [12]. On the contrary, in a 3D integration, more transistors become available by stacking more logic dies on top of each other. As the vertical distance is much shorter (i.e., in the order of μm) than the horizontal distance (i.e., in the order of mm), more transistors in a 3D integration come without an increase in the average distance between transistors.

Three-dimensional integration has its own unique challenges that can prevent 3D chips from becoming commercially attractive. The main challenge of a 3D chip is the increase in the working temperature of the chip. As multiple logic dies are stacked on top of each other, it becomes more difficult to cool the chip. Moreover, for a 3D integration, components need to have placement for the TSVs. The need for the TSV placement means that today's 2D components can rarely be reused in a 3D integration, which, consequently, leads to an increase in the initial investment for the 3D integration. Another cost overhead is due to the fact that 3D technology is not mature yet, and it is possible that some of the TSVs fail. Such failures can significantly reduce the number of fully functional 3D chips.

Despite the 3D integration challenges, there have been several 3D chip prototypes in academia and industry [10, 60, 56]. While these prototypes are expensive, they illustrate the feasibility of

connecting multiple logic dies using TSVs. These prototypes usually have few cores on each logic die to avoid the thermal issues. As the level of integration increases and more cores are fabricated on each logic die, liquid cooling technologies can be used for avoiding the thermal issues in 3D chips [84, 67]. While these technologies are expensive today, the price is expected to drop significantly as 3D technology becomes mainstream and benefits from high volume of the mainstream market.

In this chapter, we assume that 3D integration challenges are addressed. We seek to develop strategies for Scale-Out Processors to take advantage of the features specific to 3D logic-on-logic technology to boost performance beyond what is possible in standard 2D technology.

6.1 3D Logic-on-Logic Technology

As transistors become smaller, transistor scaling is becoming more expensive. The cost per transistor is likely to go up at the 14nm technology node for the first time in history [37]. Moreover, the challenges of producing transistors in the sub-10nm regime are expected to put an end to the transistor scaling [66]. Transistor scaling is the driving force behind the unprecedented growth in the semiconductor industry, and its failure will negatively affect the whole industry.

Three-dimensional logic-on-logic technology can offer some of the advantages of transistor scaling to the semiconductor industry. This technology allows multiple logic dies to be stacked on top of each other. The stacked logic dies are interconnected by dense and short vertical connections called through-silicon vias or TSVs. Taking advantage of the TSVs, 3D logic-on-logic technology has the potential to reduce the interconnect delay, increase the on-chip bandwidth, and decrease the power consumption of wires. Moreover, similar to transistor scaling, 3D technology can offer a reduction in the form factor of integrated circuits.

While both transistor scaling and 3D logic-on-logic technology enable increasing the number of transistors per unit area, there is a major difference between the two technologies. In standard 2D technology, the reduction in transistor dimensions (i.e., transistor scaling) increases the number of transistors per unit area and also increases the delays of wires due to the greater wire RC delays [12]. On the contrary, with the 3D logic-on-logic technology, stacking multiple logic dies on top of each other enables more transistors per unit area but does not change the average wire delay. With 3D logic-on-logic technology, the delays of the vertical connections are negligible (almost zero), and the delays of the horizontal connections remain unchanged. This major difference provides an opportunity for tuning existing architectures for implementation with the 3D logic-on-logic technology.

In this chapter, we use standard commodity components (e.g., cores, caches, and memory channels) and attempt to investigate the impact of 3D logic-on-logic technology on the organization of the Scale-Out Processors.

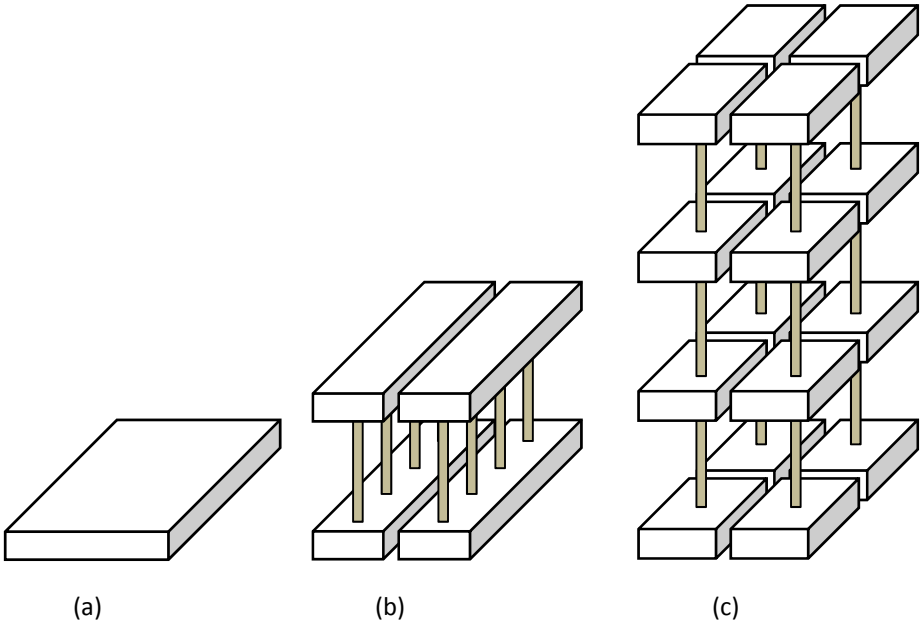


Figure 6.1: Fixed-pod strategy with one logic die (a), two stacked logic dies (b), and four stacked logic dies (c). Every connected vertical piece is a pod (i.e., (a), (b), and (c) have one, two, and four pods, respectively).

6.2 Why 3D Pods?

While 3D integration is a promising approach to extend the validity of Moore's law and enables continual increase in the performance of Scale-Out Processors when transistor scaling stops, a 3D pod design is not necessary for the 3D integration. Independent 2D pods can be constructed on every logic die of a 3D chip. Each 2D pod can function independent of the other 2D pods on upper or lower logic dies, similar to various pods on a large die in a standard 2D chip (see Chapter 3). As 3D pods increase the cost of the design and fabrication process, their existence is only justified if they provide higher throughput over 2D pods.

Three-dimensional pods are attractive because they can leverage features specific to 3D technology to boost performance. As the vertical distance between two stacked dies is much shorter than the horizontal distance in a 3D chip, 3D technology breaks the relationship between core count (or LLC capacity) and the distance between the cores and the last-level cache. This phenomenon enables two possibilities for 3D pods to take advantage of the 3D integration to boost performance. One option is to keep the number of cores and LLC capacity in each pod constant as dies are attacked on top of each other. As core count and LLC capacity in a pod do not change, the 3D pod can leverage the 3D integration to arrange the cores and the LLC in various dies to reduce the distance between the cores and the LLC. The reduction in the distance between the cores and the LLC in each pod results in a performance boost. We refer to this option as the fixed-pod strategy. Figure 6.1 highlights how the fixed-pod strategy works. An extra bonus of the fixed-pod strategy, in addition to the performance boost, is the

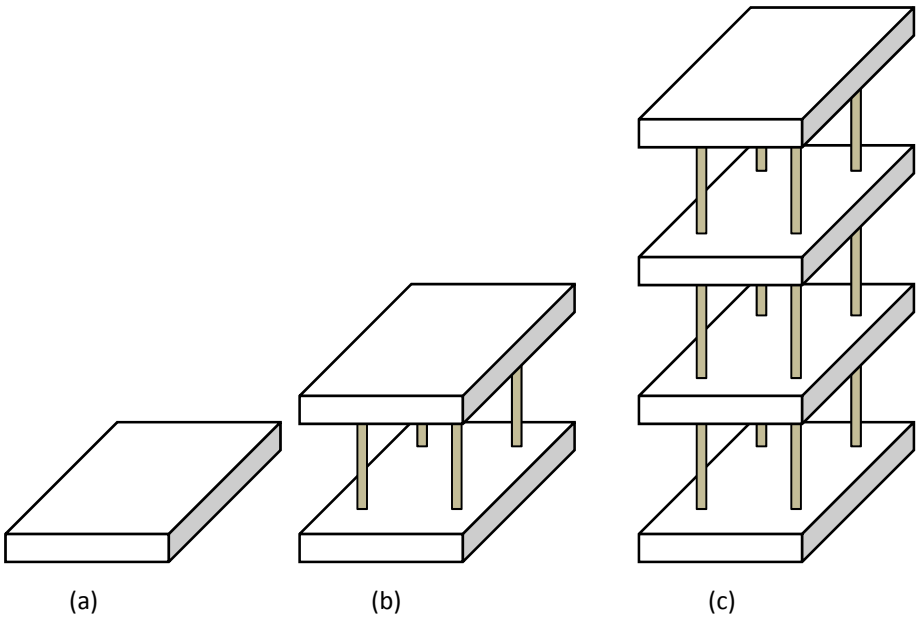


Figure 6.2: Fixed-distance strategy with one logic die (a), two stacked logic dies (b), and four stacked logic dies (c). Every connected vertical piece is a pod (i.e., (a), (b), and (c) all have one pod).

lack of need for software scalability. As the number of cores in each pod remains constant, this strategy does not require software scalability to benefit from the 3D integration.

The other alternative for 3D pods to take advantage of the 3D integration is to keep the number of cores and LLC capacity in each logic die constant and scale them up in a pod with the number of dies. In this strategy, 3D integration enables a pod to have more cores and a larger LLC without producing any change in the core-to-cache area ratio and the on-chip distance (i.e., a virtue of the vertical integration). The larger LLC in a 3D pod has a higher filtering rate, which boosts performance and also reduces the number of power-hungry and area-expensive memory channels (i.e., frees energy and area for more cores). The combination of more effective LLC and fewer memory channels ensures higher performance density for 3D pods and higher throughput for 3D Scale-Out Processors. We refer to the option of scaling core count and LLC capacity of a pod with the number of logic dies as the fixed-distance strategy. Figure 6.2 highlights how the fixed-distance strategy works. While this strategy requires software scalability to benefit from the 3D integration, it takes advantage of the fact that in such a 3D pod, the number of cores and the size of the LLC at each logic die are identical to those of a 2D pod. If a piece of software is not scalable, the 3D pod can function as multiple independent 2D pods (as as extra bonus).

The fact that 3D pods can deliver higher throughput as compared to 2D pods makes them attractive. The organization of a 3D pod has significant impacts on its performance and design complexity (just like the organization of a 2D pod). In the next section, we first introduce a

metric for 3D design-space evaluation; then, we propose an organization for a 3D pod. We will show that the organization of a 2D pod can be extended to become suitable for a 3D pod.

6.3 Metric for the 3D Design-Space Evaluation

To assess various organizations of a 3D pod, there is a need for a metric that captures conflicting requirements, such as core count, LLC capacity, area, and the number of stacked logic dies in a single representative number. This metric can make 3D design evaluation simple, just like what performance density did for the 2D design evaluation. Performance density, as defined in Chapter 2 (i.e., performance per mm^2), is not useful for 3D design evaluation because it does not consider the number of stacked logic dies. The definition for performance density needs to be extended to be useful for 3D design evaluation. While performance per unit of area is the right metric for a planar design (i.e., 2D), intuitively, the right metric for a 3D design is performance per unit of volume. Moreover, because the distance between all two adjacent logic dies in a 3D multi-die integration is the same, performance per unit of volume is proportional to performance per unit of area divided by the number of logic dies. The extended definition for the performance density (PD) makes it applicable to 3D design evaluation and also is equivalent to the definition presented in Chapter 2 when the number of stacked logic dies is one.

6.4 3D Pod Organization

The right organization for a 3D pod is an organization that maximizes performance density. A 3D pod organization, similar to that of a 2D pod, should support the following: (a) many cores; (b) modestly sized LLC; and (c) decoupled core-LLC floorplanning to turn the all-to-all traffic pattern into the bilateral traffic pattern. In this section, we propose an organization for 3D pods that supports all of the mentioned features. For this study, we only consider homogeneous organizations where the floorplanning of a 3D pod at each logic die is identical to those of other logic dies. In a homogeneous organization, the LLC is distributed across all logic dies in the 3D organization. Distributing the LLC across all logic dies has two benefits: (1) while cores occupy a significant fraction of the die area, LLC area is not large enough to occupy a complete die; and (2) stacking LLCs on top of each other enables reducing the latency of the LLC. Moreover, having a homogeneous organization reduces the complexity of the fabrication process by enabling the reuse of masks for all of the logic dies [10].

Due to the fact that the basic principles of 2D and 3D pods are the same (i.e., many cores, modestly sized LLC, and decoupled cores and LLC), the organization of a 3D pod at each logic die is identical to that of a 2D pod. Figure 6.3 shows a high-level view of the proposed organization, featuring the LLC in the center of the dies, while cores are on both sides of the LLC. At each logic die, a 3D pod uses simple, routing-free reduction trees to guide packets toward the centralized LLC banks, as well as dispersion trees, which are logical opposites of

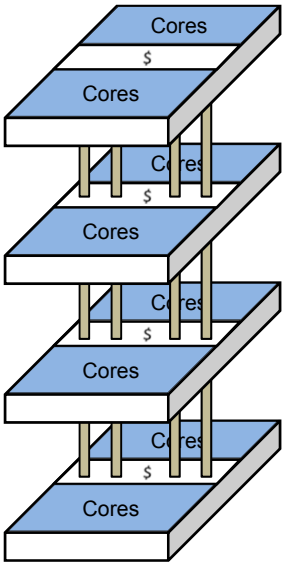


Figure 6.3: Organization of a 3D pod.

reduction trees, to propagate response data and snoop traffic out to the cores. Every reduction and dispersion tree connects a small number of cores to exactly one LLC bank. The LLC banks are linked in a flattened butterfly topology, forming a low-latency NUCA cache. (For more details, please refer to Chapter 4.)

While in a 2D pod, LLC slices are distributed across a planar surface in the center of the die, in a 3D pod, LLC slices are distributed across a 3D volume in the center of various dies that are stacked on top of each other. To provide connectivity between the cores and all of the LLC slices, and due to the fact that the LLC slices are distributed to all of the logic dies, all of the logic dies need to be connected together. Just like 2D pods, the connectivity within 3D pods are provided in a way that the die section dedicated to cores remains as simple as possible. For this goal, the flattened butterfly routers in the LLC region in the center of the dies are extended to provide connectivity across stacked dies in a 3D pod. Figure 6.3 shows a high-level view of the proposed organization. Routers in the LLC region are connected together using TSVs to form a vertical flattened butterfly network. Identical to a 2D pod, cores send their requests to the LLC region (in the same die) using the reduction network. Then, the flattened butterfly network in the LLC region forwards the request to the destination if it happens to be on the same die. If the LLC slice that holds the requested data happens to be on a different die, the router (in the same die as the requesting core) that is in the same position as the destination receives the request and forwards it to the final destination using the vertical TSV connections.

6.5 Methodology

We compare the performance of 3D Scale-Out Processors to 2D Scale-Out Processors using a combination of cycle-accurate simulation, analytic models, and technology studies.

Table 6.1: Area and power estimates for various system components at 40nm.

Component		Area	Power
Cores	**OoO**	4.5mm^2	1W
	In-order	1.3mm^2	0.48W
LLC	**16-way SA**	5mm^2 per MB	1W per MB
Interconnect		0.2 - 4.5 mm^2	<5W
DDR4 interface (PHY+ controller)		(2 + 10) mm^2	5.7W

6.5.1 Design and Technology Parameters

We compare 2D and 3D Scale-Out Processors in 40nm technology with an on-chip supply voltage of 0.9V. We model processors with an area of 250-280mm^2 (per logic die), a power budget of 250W [84, 67], and a maximum of six single-channel DDR4 interfaces; these parameters are an estimation for the power, area, and bandwidth budgets of 3D integrated chips. Design parameters are summarized in Table 6.1.

We determine the LLC capacity and core count of pods in 2D and 3D Scale-Out Processors by evaluating a broad design space from 1 to 1024 cores and LLC capacities in the range of 2 to 32MB. Results are presented in Section 6.6. We model as many pods as can be afforded without exceeding the area, energy, and bandwidth constraints specified in this section. The number of memory channels is computed to accommodate the worst-case bandwidth demand across the workload spectrum for every core/LLC configuration.

6.6 Results

We now compare 3D Scale-Out Processor designs to 2D Scale-Out Processors. For each Scale-Out design, we first find a performance-density-optimal pod organization. Then, we integrate the pods up to the area, energy, and bandwidth limits per Section 6.5.1.

6.6.1 3D Scale-Out Processors with Out-of-Order Cores

We begin our study with out-of-order cores. Figure 6.4 plots performance density, averaged across all workloads, for five different LLC sizes, while the number of logic dies stacked on top of each other varies from one to four.

In the configuration with a single logic die, performance density maximizes with 32 cores and 2MB of LLC and diminishes with larger core count or LLC capacity, indicating that the physical distance between the cores and the LLC in a 2-dimensional logic die hurts performance when integrating a large number of cores or LLC capacity. In configurations with more than one logic die, pods with 32 cores and 2MB of LLC (i.e., fixed-pod strategy) are competing with pods with 32 cores and 2MB of LLC per logic die (i.e., fixed-distance strategy) for the highest

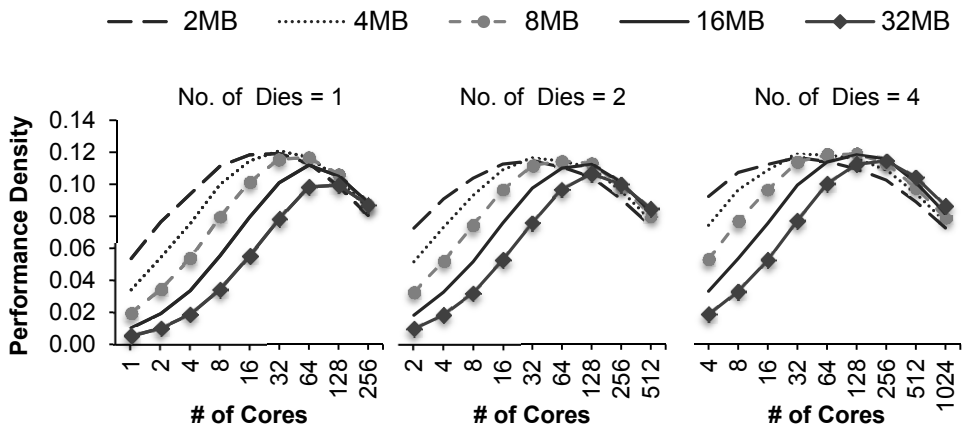


Figure 6.4: Performance density for a system with out-of-order cores, a range of last-level cache sizes, and various numbers of stacked logic dies.

performance density.

On one hand, 3D technology enables pods to place the cores and the LLC across several logic dies while keeping the core count and LLC capacity constant. This strategy (i.e., fixed-pod strategy) reduces the footprint of a pod on a single logic die, decreases the distance from the cores to the LLC, and, as a result, boosts performance. On the other hand, 3D technology allows expanding a pod with more cores and larger LLC capacity without increasing the on-chip distance (i.e., fixed-distance strategy) by placing the cores and the LLC across several logic dies. Not only does such a 3D pod have the same on-chip distance as a 2D pod, but also, because of having a larger LLC capacity and, hence, a lower LLC filtering rate, this 3D pod has a higher performance density.

To explore these two alternatives, Figure 6.5 examines performance density of 3D Scale-Out Processors based on the fixed-pod and the fixed-distance strategies. In the fixed-pod strategy, the number of cores and LLC capacity in the pod are independent of the number of logic dies (i.e., 32 cores and 2MB of LLC). In the fixed-distance strategy, the core count and LLC capacity are scaled up with the number of logic dies (i.e., 32 cores and 2MB of LLC per logic die). As expected, the more logic dies that are placed on top of each other, the higher the performance density of a pod will become. For pods with OoO cores, the fixed-pod strategy delivers the highest performance density (note that the performance difference of the two strategies is small). We, therefore, adopt a 3D pod with 32 cores and 2MB of LLC as the preferred pod configuration due to its high performance density. With only two logic dies, the performance density of the 3D pod is 5% higher than that of the 2D pod, and with four logic dies, the improvement in performance density reaches 8%.

Chip-level assessment. Under the constraints specified in Section 6.5.1, a Scale-Out Processor with 1, 2, and 4 stacked logic dies can afford one, two, and four pods, respectively, before reaching the area limit.

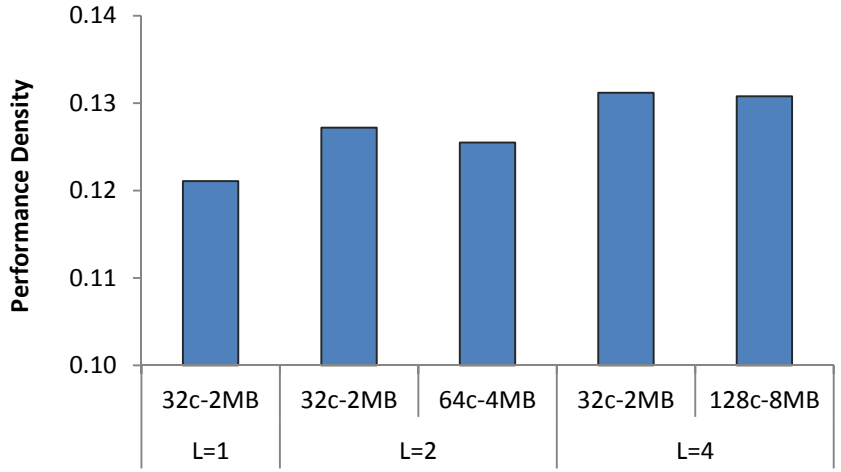


Figure 6.5: Performance density of 3D Scale-Out Processors (OoO) with the fixed-pod and the fixed-distance strategies. The labels on the x-axis show the configuration of the pod and the number of stacked logic dies. The number of pods (not shown) is determined by the constraints.

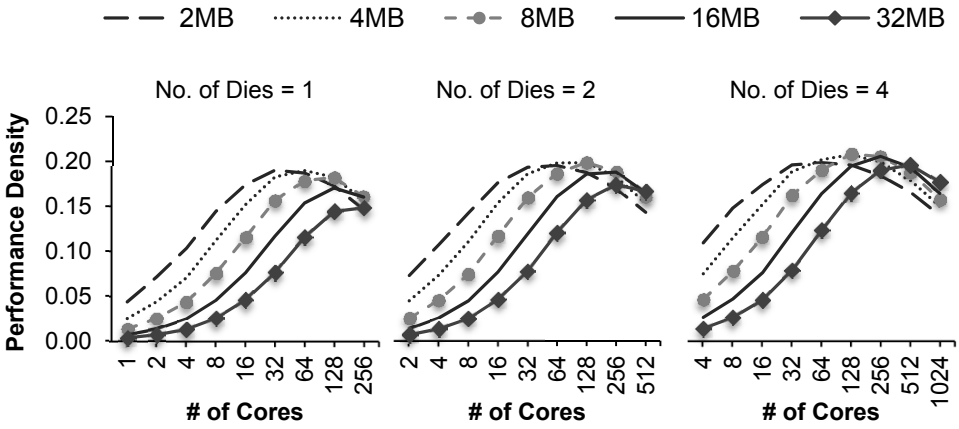


Figure 6.6: Performance density for a system with in-order cores, a range of last-level cache sizes, and various numbers of stacked logic dies.

6.6.2 3D Scale-Out Processors with In-Order Cores

Figure 6.6 illustrates performance density results, averaged across all workloads, for cache sizes ranging from 2 to 32MB and number of stacked logic dies ranging from one to four. The general trends are similar to those described in the previous section; however, simpler cores in a throughput-oriented architecture yield an optimal pod design with 64 cores and 2MB of LLC with one logic die.

Similar to the observation for OoO cores, the increase in the number of cores and LLC capacity in a single logic die (beyond the optimal point) hurts performance due to the growth in the

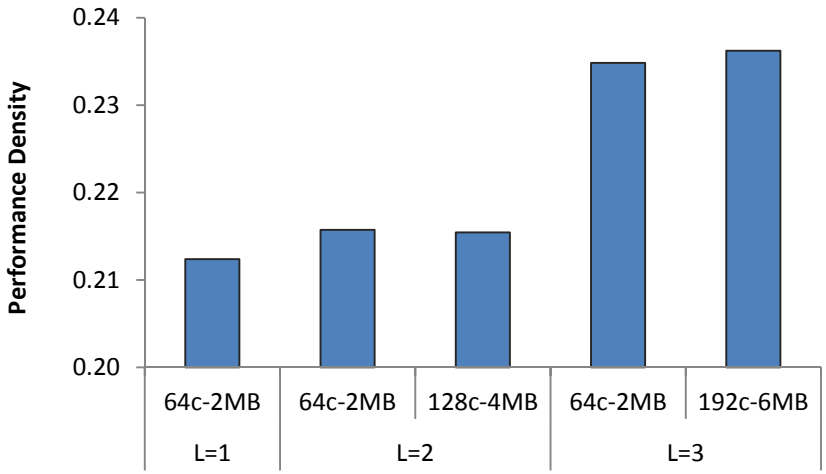


Figure 6.7: Performance density of 3D Scale-Out Processors (in-order) with the fixed-pod and the fixed-distance strategies. The labels on the x-axis show the configuration of the pod and the number of stacked logic dies. The number of pods (not shown) is determined by the constraints.

on-chip distance. The 3D technology allows either keeping the pod constant and reducing the distance (i.e., fixed-pod strategy) or expanding a pod with more cores and larger LLC capacity without increasing the on-chip distance (i.e., fixed-distance strategy). Similar to what we observed for OoO cores, these two strategies are competing to deliver the highest performance density. Figure 6.7 explores these two alternatives for 3D Scale-Out Processors with in-order cores. In the fixed-pod strategy, a pod keeps 64 cores and 2MB of LLC, independent of the number of stacked dies, while in the fixed-distance strategy, the core count and LLC capacity in a pod scale up with the number of logic dies (i.e., 64 cores and 2MB of LLC per logic die). As with four logic dies, both strategies produce chips that are bandwidth-limited, the maximum number of stacked dies in Figure 6.7 is three.

Similar to the results with OoO cores, with two logic dies, the fixed-pod strategy delivers higher performance than the fixed-distance strategy. However, with three logic dies, the fixed-distance strategy delivers higher performance than the fixed-pod strategy, as with three logic dies, 3D Scale-Out Processors with in-order cores are near the bandwidth saturation point. Because the pod with the fixed-distance strategy has a larger LLC, it can use the valuable bandwidth more efficiently and, as such, it can deliver a higher performance. We, therefore, adopt a 3D pod with 64 cores and 2MB of LLC (i.e., fixed-pod strategy) for two logic dies and a 3D pod with 196 cores and 6MB of LLC (i.e., fixed-distance strategy) for three logic dies. As expected, the more logic dies that are placed on top of each other, the higher the performance density of a pod will become. With only two logic dies, the performance density of the 3D pod is 2%, and with three logic dies, 6% larger than that of the 2D pod.

Chip-level assessment. The optimal 3D pod configuration changes with the number of stacked logic dies. A 3D Scale-Out Processor with in-order cores integrates one pod of 64 cores

Table 6.2: Specification of various 2D and 3D Scale-Out Processors.

Core type	#Dies	Configuration	#Pods	Pod		#MCs	PD
				#Cores	LLC (MB)		
OoO	1	**2D Pod**	1	32	2	2	0.121
	2	**Fixed-Pod**	2	32	2	3	0.127
		Fixed-Distance	1	64	4	3	0.125
	4	**Fixed-Pod**	4	32	2	6	0.131
		Fixed-Distance	1	128	8	5	0.130
In-order	1	**2D Pod**	1	64	2	2	0.212
	2	**Fixed-Pod**	2	64	2	4	0.216
		Fixed-Distance	1	128	4	4	0.215
	3	**Fixed-Pod**	3	64	2	6	0.223
		Fixed-Distance	1	192	6	5	0.226

and 2MB of LLC with one logic die, two pods of 64 cores and 2MB of LLC with two logic dies, and one pod of 196 cores and 6MB of LLC with three logic dies.

Table 6.2 summarizes the results of the evaluation of 2D and 3D Scale-Out Processors with out-of-order and in-order cores.

7 Related Work

7.1 Scale-Out Design Methodology

The notion that scale-out workloads benefit from a many-core architecture was advocated by Hardavellas et al. [42], who argued for the use of simpler cores and minimum on-die caches, provided that there is no bandwidth bottleneck. This dissertation extends that idea by introducing a scalable and efficient implementation of such a many-core architecture.

In order to reduce access time to the LLC, researchers have proposed Non-Uniform Cache Architectures (NUCA) [53]. The access latency of NUCA caches is dominated by the interconnect delays in processors with many cores and cache banks. One way to overcome the interconnect delays in NUCA caches is through richly connected topologies [55]; however, these topologies have been shown to have significant area and energy overheads in many-core chips [35]. As an alternative, there is a large body of work attempting to reduce the NUCA cache access latency by a combination of replication and relocation of cache blocks, but they either require complex lookups [18, 20, 11, 95, 19, 50], waste LLC capacity [95, 11], are not scalable [18, 36], optimize only a subset of the LLC accesses [11, 18, 21], or are not applicable to modestly sized LLCs [41]; as such, they are not suitable for Scale-Out Processors. In this dissertation, we show that a pod-based design with a simple crossbar/NOC-Out interconnect overcomes the inefficiency of NUCA designs on scale-out workloads.

Prior work that tried to find the optimal CMP design either focused on finding the optimal cache architecture for a given core count [48, 96] or on finding the optimal core microarchitecture for a given workload [29]. Most of the prior work assumes non-datacenter workloads [29, 74]. Oh et al. [74] presented a simple and effective analytic model to study the core count versus cache capacity trade-off in CMPs under die area constraints, showing that for a fixed cache size, an increase in core count hurts the aggregate performance beyond a certain point. This dissertation corroborates the result on scale-out workloads.

Prior research and industry efforts have attempted to maximize the compute area by reducing the fraction of the die allocated to the cache. Kgil et al. [52] proposed eliminating last-level

caches and devoting their area to cores, while compensating for the increase in memory bandwidth pressure through 3D-stacked DRAM. Similarly, we find that increasing the fraction of the chip dedicated to compute is important for throughput; however, we also observed that scale-out workloads have reuse in their secondary working sets and benefit from a modest cache size. Graphics processors (GPUs) also use a large number of processing elements with minimal cache resources. For instance, Tesla C1060 GPUs have 240 processing elements with under 756KB of aggregate cache capacity [73]. GPU architectures are tuned for high throughput and are unlikely to satisfy latency demands of real-time online services.

Certain commercial and research chips share some of the insights or conclusions of this dissertation. Piranha [8] was the first chip multi-processor designed for commercial server workloads that used simple cores for higher efficiency. In this dissertation, we also showed that using simple cores for scale-out workloads is beneficial from a performance density perspective. Sun Niagara III is a contemporary server processor that, at a high level, resembles a Scale-Out pod, in that it features 16 cores and a 6MB LLC connected via a crossbar switch [81]. However, the cores are 8-way multi-threaded, resulting in poor single-threaded performance and high area overhead. In addition, Niagara chips have not adopted a multi-pod design; instead, they demonstrate scaling-up capabilities through additional resources (e.g., more cores, larger LLC).

7.2 NOC-Out: Microarchitecting a Scale-Out Processor

NOC-Out is not the first attempt to optimize the on-chip interconnect for a specific workload domain. Bakhoda et al. proposed a NOC design optimized for GPU-based throughput accelerators [3]. Significant similarities and differences exist between the two efforts. Both designs address the needs of thread-rich architectures characterized by a memory-resident data working set and a many-to-few-to-many traffic pattern. However, whereas workloads running on throughput accelerators are shown to be insensitive to NOC latency, we show scale-out workloads to be highly sensitive to interconnect delays due to frequent instruction fetches from the LLC. As a result, NOC-Out innovates in the space of delay-optimized on-chip topologies, whereas prior work has focused on throughput and cost in the context of meshes.

One effort aimed at boosting NOC efficiency specifically in the context of server processors was CCNoC, which proposed a dual-mesh interconnect with better cost-performance characteristics than existing multi-network alternatives [87]. This dissertation shows that mesh-based designs are suboptimal from a performance perspective in many-core server processors.

A number of earlier studies sought to reduce NOC area cost and complexity through microarchitectural optimizations in crossbars [54, 88], buffers [70], and links [68]. A recent study examined challenges of NOC scalability in kilo-node chips and proposed an interconnect design that co-optimized buffering, topology, and flow control to reduce NOC area and energy [35]. All of these efforts assume a conventional tiled organization. In contrast, our NOC-Out design lowers NOC area overheads by limiting the extent of on-die connectivity.

However, NOC-Out's efficiency can be further improved by leveraging many of the previously proposed optimizations.

Finally, Huh et al. preceded NOC-Out in proposing a segregated NUCA CMP architecture, in which core and LLC tiles are disjointed [45]. Our design is different from Huh's in that it seeks to reduce the number of cache tiles to lower network cost, whereas Huh relied on a sea of cache tiles to optimize data placement and partitioning.

7.3 Datacenter Analysis

There are various pieces of work targeting estimation of cost or power consumption of datacenters to optimize their operation [59, 75, 86, 51, 44, 38]. These pieces of work estimate the cost or power consumption of datacenters using models that range from simple spreadsheet models [38] to detailed models considering the performance of various workloads running in datacenters [59]. James Hamilton used a simple spreadsheet model to show that server acquisition and power costs constitute the two largest TCO components [38]. The results of our study corroborate the results presented by James Hamilton. Pitt Turner et al. showed that datacenter analysis based on cost per area (i.e., cost normalized to area) may lead to disappointing results [86]. In our study, we use performance per total cost of ownership for the evaluation of different datacenter designs.

One of the key results of our study that multi-pod Scale-Out Processors are beneficial from a TCO perspective is similar to the observation made by Karidis et al., who noted that high-capacity servers are effective in lowering the cost of computation [51]. Moreover, our results corroborate earlier studies that identify efficiency benefits stemming from the use of lower-complexity cores, as compared to those used in conventional server processors. Given this observation, many companies started producing servers using lower-complexity cores [78, 15, 14, 91]. Finally, our TCO analysis was derived by using EETCO [44], which considers four major expense categories: datacenter infrastructure, server and networking hardware, power, and maintenance.

7.4 3D Integration

Three-dimensional integration [26] is an attractive alternative to transistor scaling to extend the validity of Moore's law by offering an opportunity to continue the CMOS scaling trend. In a 3D chip, multiple logic layers are stacked on top of each other. Various vertical connections for 3D integrated chips have been explored, including wire bonded [17], μbump [94], contactless [16], and through-silicon via [26]. Through-silicon via has the potential to offer the greatest density and, as a result, is the most attractive option. There are two different approaches for implementing through-silicon via. The first approach involves the sequential device process, in which the front-end processing (to build a logic layer) is repeated on a single wafer to build multiple active logic layers before the connections between logic layers are

built. The second approach processes each active logic layer separately, using conventional fabrication techniques, and then stacks multiple logic layers together using wafer bonding. The latter approach requires minimal changes to the existing manufacturing process and, consequently, is more attractive. Logic layers can be bonded face-to-face (F2F) or face-to-back (F2B) [62]. The through wafer via in F2F wafer-bonding does not go through the thick buried silicon layer and can be fabricated with smaller via sizes. However, for 3D chips with more than two active layers, F2B stacking provides better scalability [62].

Thermal considerations have been a significant concern for 3D integration [22]. As a result, various techniques have been developed to address thermal issues in 3D architectures, such as physical design optimization through intelligent placement [34], increasing thermal conductivity of the stack through insertion of thermal vias [22], and through the use of novel cooling structures [23].

Many researchers focused on exploring the potential benefits of 3D stacked processor architectures. Beanato et al. [10] designed and fabricated a 3D chip composed of completely identical stacked dies connected together by through-silicon vias. Each die features four 32-bit embedded processors and associated memory modules, interconnected by a 3D network-on-chip, which can route packets in the vertical direction. Black et al. [13] proposed an architecture, which arranges the logic modules of an Intel Pentium 4 microprocessor in clusters and reorganized them in two stacked layers, resulting in considerable performance gains and energy savings at constant frequency. Loh proposed a processor architecture where a baseline architecture can be augmented with additional 3D stacked resources to achieve higher performance [61].

8 Conclusions

Global-scale online services have gained popularity in recent years. Service providers like Microsoft, Google, and Facebook serve millions of users across the world with search capabilities, media streaming, and social networking. Large-scale datacenters with thousands of servers are the computing platforms that deliver such services to the world. The massive scale of such datacenters requires an enormous capital outlay for infrastructure, hardware, and power consumption. With demand for online services skyrocketing around the globe, efficiency has become a paramount concern in the design and operation of large-scale datacenters. As processors are the primary contributor to the performance and one of the major contributors to the cost of datacenters, they were the focus of this dissertation.

Unfortunately, there is a mismatch between what existing processors offer and what is needed for global-scale online services [30]. While online services are data-centric and memory-intensive, existing processors offer aggressive OoO cores that are ineffective for memory-intensive workloads. Moreover, existing processors allocate almost half of their transistor budget to large last-level caches (LLCs) that are ineffective for online services with massive datasets due to limited reuse. Finally, processor vendors plan to build even larger caches in their future processors, which make them even more inefficient. We need to redesign processors to make them suitable for global-scale online services.

Recent research examining scale-out workloads behind many of today's online services has shown that, as a class, these workloads have a set of common characteristics [30]. The presence of common characteristics – namely, (a) request independence; (b) large instruction footprints; and (c) vast dataset sizes – indicates that server processors can be optimized for this workload class. The abundant request-level parallelism argues for processor designs with a large number of cores to maximize throughput. The independent nature of requests virtually eliminates inter-thread communication activity; however, large instruction footprints require a fast communication path between the individual cores and the last-level cache containing the applications' instructions. Finally, the vast dataset dwarfs on-die storage capacities and offers few opportunities for caching, due to limited reuse [30].

Unfortunately, existing processor organizations cannot fulfill all of the requirements simultaneously. Taking an existing processor, it is easy to reduce the size of the cache, replace aggressive OoO cores with simpler ones, and instead increase the core count. However, larger core count translates into larger distance between the individual cores and the LLC in existing processor organizations. As instructions are resident in the LLC, the large distance between the cores and the LLC slows down the instruction fetch and hurts the performance of scale-out workloads. Moreover, as process technology scales and more cores are added to processors, this problem exacerbates.

Taking advantage of common workload features, and driven by the need to increase processor efficiency, we introduced and formalized the Scale-Out Processor (SOP) design methodology. The SOP methodology calls for partitioning a chip into stand-alone modules named pods to break the relationship between the core count and the on-chip distance. Each pod is a server running an operating system and a full software stack. The SOP methodology provides an optimization framework for deriving the optimal core count and LLC capacity in each pod based on microarchitectural and technology parameters and advocates many cores, modestly sized LLCs, and low interconnect delays. The end result is a design called *Scale-Out Processor*, which delivers peak throughput in today's technology and affords near-ideal scalability as process technology scales.

With SOP methodology calling for many-core pods, a many-core organization that enables fast access to the LLC is essential for the success of Scale-Out Processors. While crossbar-based pods suffer from scalability limitations, mesh-based pods enable cost-effective scalability to high core count. Mesh-based pods feature a mesh-based interconnect fabric and a tiled organization where each tile integrates a core, a slice of the shared LLC with directory, and a router. Unlike their cost-effective scalability to high core counts, mesh-based pods sacrifice performance on scale-out workloads due to their large average hop counts [65]. Each hop involves a router traversal, which adds delay that prolongs the core stall time on instruction fetches serviced by the LLC. To reduce NOC latency, researchers have proposed low-diameter NOC topologies, such as the flattened butterfly [55], that leverage the abundant on-chip wire budget to achieve rich inter-node connectivity. By minimizing the number of router traversals, a low-diameter network improves performance over a mesh-based pod by accelerating accesses to the LLC. However, the performance gain comes at a considerable area overhead stemming from the use of many-ported routers and a multitude of repeater-intensive long-range links.

In this dissertation, we addressed the scalability challenge for pods in Scale-Out Processors through NOC-Out, a core, cache, and interconnect organization optimized for the target workload domain. We identified the direct communication between cores and LLC banks, which we termed *bilateral*, as the dominant permutation in scale-out workloads and showed that other forms of communication, including coherence activity, are rare. Based on this insight, NOC-Out decoupled LLC tiles from the cores and localized them in a central portion of the die. The segregated organization naturally accommodated the bilateral core-to-cache access pattern. More importantly, with the traffic flowing between spatially distinct regions

(cores to caches and back to the cores), NOC-Out virtually eliminated the need for direct inter-core connectivity, affording a significant reduction in the network cost.

We further optimized cost and performance of NOC-Out and deployed simple *reduction* and *dispersion trees* to carry messages from the cores to the centrally located LLC banks and vice versa. Each reduction or dispersion tree is shared by a small number of cores. A node in a reduction tree is just a buffered 2-input mux that merges packets from a local port with those already in the network. We reduced cost and delay by eliminating the need for routing, multi-port arbitration, complex switches, and deep buffers. Moreover, a node in a dispersion tree is a logical opposite of that in a reduction tree, allowing packets to either exit the network or advancing them up the tree at minimal cost and delay.

To show the impact of Scale-Out Processors on datacenters, we evaluated the efficiency of datacenters, which we measured by using *performance per total cost of ownership*, for various server processors. We demonstrated that Scale-Out Processors improve the efficiency of existing commercial datacenters by an order of magnitude. Moreover, we showed that in the context of datacenter efficiency, Scale-Out Processors with just one pod are suboptimal, and multi-pod Scale-Out Processors are necessary to maximize the efficiency of datacenters.

Finally, we extended the organization of Scale-Out Processors for implementation with 3D logic-on-logic technology. The 3D logic-on-logic technology is the likely successor of transistor scaling and enables more transistors in the same area. We showed that Scale-Out Processors can take advantage of the features specific to the 3D logic-on-logic technology to reduce the distance from the cores to the LLC and improve performance.

8.1 Limitations and Future Work

We observed that the growing distance between the individual cores and the last-level cache is a performance bottleneck for many-core processors. While we solved this problem by partitioning a processor into multiple pods, this solution is only useful for workloads that are capable of distributing the load across many servers (or pods). Extending the idea presented in this dissertation to be applicable to a broader set of workloads is one of our plans for the future.

The focus of this dissertation was on the allocation and organization of cores and caches in multi-core processors. While cores and caches are important components of a processor, a complete processor also has a variety of IO interfaces. In this dissertation, we considered memory and network interfaces and assumed that memory interfaces are shared across pods and network interfaces are private to pods. To have a complete processor, further research on IO interfaces is necessary. Open questions include the following: (a) what are the right interfaces to integrate on a Scale-Out Processor (e.g., network, storage); (b) for each integrated interface, is it shared across pods or private to a pod; and (c) how can we best share an interface across pods.

Finally, as technology scales and more pods will be integrated on Scale-Out Processors, these processors eventually become constrained by their off-chip traffic and power consumption. As earlier research showed, many-core processors with simple cores first experience the bandwidth wall and then the energy wall [42]. Further research is necessary to investigate how to address the bandwidth and energy walls in the context of Scale-Out Processors. For the bandwidth wall, one opportunity is to take advantage of large DRAM caches (e.g., embedded DRAM [77], 3D-stacked DRAM [13]) to significantly cut the off-chip traffic. To address the energy wall, one possibility is to provide cores with a set of accelerators suitable for scale-out workloads to reduce the energy consumed per operation for those operations that can be mapped to the accelerators.

Bibliography

[1] CloudSuite 1.0. http://parsa.epfl.ch/cloudsuite.

[2] AMD. AMD Opteron 6300 Series Processor. http://www.amd.com/us/Documents/Opteron_6300_QRG.pdf, 2012.

[3] Ali Bakhoda, John Kim, and Tor M. Aamodt. Throughput-Effective On-Chip Networks for Manycore Accelerators. In *Proceedings of the 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 421–432, December 2010.

[4] James D. Balfour and William J. Dally. Design Tradeoffs for Tiled CMP On-Chip Networks. In *Proceedings of the 20th Annual ACM International Conference on Supercomputing*, pages 187–198, June 2006.

[5] Max Baron. The F1: TI's 65nm Cortex-A8. *Microprocessor Report*, 20(7):1–9, July 2006.

[6] Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 2nd edition, 2013.

[7] Luiz André Barroso, Jeffrey Dean, and Urs Hölzle. Web Search for a Planet: The Google Cluster Architecture. *IEEE Micro*, 23(2):22–28, March-April 2003.

[8] Luiz André Barroso, Kourosh Gharachorloo, Robert McNamara, Andreas Nowatzyk, Shaz Qadeer, Barton Sano, Scott Smith, Robert Stets, and Ben Verghese. Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pages 282–293, June 2000.

[9] Luiz André Barroso and Urs Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 1st edition, 2009.

[10] Giulia Beanato, Paolo Giovannini, Alessandro Cevrero, Panagiotis Athanasopoulos, Michael Zervas, Yuksel Temiz, and Yusuf Leblebici. Design and Testing Strategies for Modular 3-D-Multiprocessor Systems Using Die-Level Through Silicon Via Technology. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2(2):295–306, June 2012.

[11] Bradford M. Beckmann, Michael R. Marty, and David A. Wood. ASR: Adaptive Selective Replication for CMP Caches. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 443–454, December 2006.

[12] Kerry Bernstein, Paul Andry, Jerome Cann, Phil Emma, David Greenberg, Wilfried Haensch, Mike Ignatowski, Steve Koester, John Magerlein, Ruchir Puri, and Albert Young. Interconnects in the Third Dimension: Design Challenges for 3D ICs. In *Proceedings of the 44th Annual Design Automation Conference*, pages 562–567, June 2007.

[13] Bryan Black, Murali Annavaram, Ned Brekelbaum, John DeVale, Lei Jiang, Gabriel H. Loh, Don McCauley, Pat Morrow, Donald W. Nelson, Daniel Pantuso, Paul Reed, Jeff Rupley, Sadasivan Shankar, John Shen, and Clair Webb. Die Stacking (3D) Microarchitecture. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 469–479, December 2006.

[14] Jag Bolaria. AMD Attacks Servers on Many Fronts. *Microprocessor Report*, 27(7):9–12, July 2013.

[15] Jag Bolaria. Intel Beats ARM to Microservers. *Microprocessor Report*, 27(4):1–7, April 2013.

[16] Roberto Canegallo, Luca Ciccarelli, Federico Natali, Alberto Fazzi, Roberto Guerrieri, and PierLuigi Rolandi. 3D Contactless Communication for IC Design. In *Proceedings of the IEEE International Conference on Integrated Circuit Design and Technology*, pages 241–244, June 2008.

[17] Flynn Carson, Hun Teak Lee, Jae Hak Yee, Jeffrey Punzalan, and Edward Fontanilla. Die to Die Copper Wire Bonding Enabling Low Cost 3D Packaging. In *Proceedings of the 61st IEEE Electronic Components and Technology Conference*, pages 1502–1507, June 2011.

[18] Jichuan Chang and Gurindar S. Sohi. Cooperative Caching for Chip Multiprocessors. In *Proceedings of the 33rd Annual International Symposium on Computer Architecture*, pages 264–276, June 2006.

[19] Zeshan Chishti, Michael D. Powell, and T. N. Vijaykumar. Distance Associativity for High-Performance Energy-Efficient Non-Uniform Cache Architectures. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 55–66, December 2003.

[20] Zeshan Chishti, Michael D. Powell, and T. N. Vijaykumar. Optimizing Replication, Communication, and Capacity Allocation in CMPs. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture*, pages 357–368, June 2005.

[21] Sangyeun Cho and Lei Jin. Managing Distributed, Shared L2 Caches through OS-Level Page Allocation. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 455–468, December 2006.

[22] Jason Cong and Yan Zhang. Thermal Via Planning for 3-D ICs. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 745–752, November 2005.

[23] Bing Dang, Paul Joseph, Muhannad Bakir, Todd Spencer, Paul Kohl, and James Meindl. Wafer-Level Microfluidic Cooling Interconnects for GSI. In *Proceedings of the IEEE International Interconnect Technology Conference*, pages 180–182, June 2005.

[24] Howard David, Chris Fallin, Eugene Gorbatov, Ulf R. Hanebutte, and Onur Mutlu. Memory Power Management via Dynamic Voltage/Frequency Scaling. In *Proceedings of the 8th ACM International Conference on Autonomic Computing*, pages 31–40, June 2011.

[25] John D. Davis, James Laudon, and Kunle Olukotun. Maximizing CMP Throughput with Mediocre Cores. In *Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques*, pages 51–62, September 2005.

[26] W. Rhett Davis, John Wilson, Stephen Mick, Jian Xu, Hao Hua, Christopher Mineo, Ambarish M. Sule, Michael Steer, and Paul D. Franzon. Demystifying 3D ICs: The Pros and Cons of Going Vertical. *IEEE Design & Test of Computers*, 22(6):498–510, November 2005.

[27] Robert H. Dennard, Jin Cai, and Arvind Kumar. A Perspective on Today's Scaling Challenges and Possible Future Directions. *Solid-State Electronics*, 51(4):518–525, April 2007.

[28] Robert H. Dennard, Fritz H. Gaensslen, Hwa-Nien Yu, V. Leo Rideout, Ernest Bassous, and Andre R. Leblanc. Design of Ion-Implanted MOSFET's with very Small Physical Dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, October 1974.

[29] Magnus Ekman and Per Stenstrom. Performance and Power Impact of Issue-Width in Chip-Multiprocessor Cores. In *Proceedings of the 5th International Conference on Parallel Processing*, pages 359–363, October 2003.

[30] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. Clearing the Clouds: A Study of Emerging Scale-Out Workloads on Modern Hardware. In *Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 37–48, March 2012.

[31] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. Quantifying the Mismatch between Emerging Scale-Out Applications and Modern Processors. *ACM Transactions on Computer Systems (TOCS)*, 30(4):15:1–15:24, November 2012.

[32] JEDEC Global Standards for the Microelectronics Industry. JEDEC Announces Key Attributes of Upcoming DDR4 Standard. http://www.jedec.org/news/pressreleases/jedec-announces-key-attributes-upcoming-ddr4-standard, August 2011.

[33] David Geer. Chip Makers Turn to Multicore Processors. *Computer*, 38(5):11–13, May 2005.

[34] Brent Goplen and Sachin Sapatnekar. Efficient Thermal Placement of Standard Cells in 3D ICs Using a Force Directed Approach. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 86–89, November 2003.

[35] Boris Grot, Joel Hestness, Stephen W. Keckler, and Onur Mutlu. Kilo-NOC: A Heterogeneous Network-on-Chip Architecture for Scalability and Service Guarantees. In *Proceedings of the 38th Annual International Symposium on Computer Architecture*, pages 401–412, June 2011.

[36] Zvika Guz, Idit Keidar, Avinoam Kolodny, and Uri C. Weiser. Utilizing Shared Data in Chip Multiprocessors with the Nahalal Architecture. In *Proceedings of the 20th Annual Symposium on Parallelism in Algorithms and Architectures*, pages 1–10, June 2008.

[37] Tom R. Halfhill. Foundries Stretch for FinFETs: GlobalFoundries, TSMC Struggle to Catch Intel, Obey Moore's Law. *Microprocessor Report*, 27(4):21–27, April 2013.

[38] James Hamilton. Overall Data Center Costs. http://perspectives.mvdirona.com/2010/09/18/OverallDataCenterCosts.aspx, September 2010.

[39] Nikos Hardavellas. *Chip Multiprocessors for Server Workloads*. PhD thesis, Carnegie Mellon University, July 2009.

[40] Nikos Hardavellas, Michael Ferdman, Anastasia Ailamaki, and Babak Falsafi. Power Scaling: The Ultimate Obstacle to 1K-Core Chips. In *Technical Report NWU-EECS-10-05, Northwestern University, Evanston, IL*, March 2010.

[41] Nikos Hardavellas, Michael Ferdman, Babak Falsafi, and Anastasia Ailamaki. Reactive NUCA: Near-Optimal Block Placement and Replication in Distributed Caches. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, pages 184–195, June 2009.

[42] Nikos Hardavellas, Michael Ferdman, Babak Falsafi, and Anastasia Ailamaki. Toward Dark Silicon in Servers. *IEEE Micro*, 31(4):6–15, July-August 2011.

[43] Nikos Hardavellas, Ippokratis Pandis, Ryan Johnson, Naju G. Mancheril, Anastassia Ailamaki, and Babak Falsafi. Database Servers on Chip Multiprocessors: Limitations and Opportunities. In *Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research*, pages 79–87, January 2007.

[44] Damien Hardy, Isidoros Sideris, Ali Saidi, and Yiannakis Sazeides. EETCO: A Tool to Estimate and Explore the Implications of Datacenter Design Choices on the TCO and the Environmental Impact. In *Proceedings of the Workshop on Energy-Efficient Computing for a Sustainable World*, December 2011.

[45] Jaehyuk Huh, Changkyu Kim, Hazim Shafi, Lixin Zhang, Doug Burger, and Stephen W. Keckler. A NUCA Substrate for Flexible CMP Cache Sharing. In *Proceedings of the 19th Annual International Conference on Supercomputing*, pages 31–40, June 2005.

[46] Intel. Intel Xeon Processor X5670. http://ark.intel.com/products/47920/.

[47] International Technology Roadmap for Semiconductors (ITRS), 2011 Edition.

[48] Norman P. Jouppi and Steven J. E. Wilton. Tradeoffs in Two-Level On-Chip Caching. In *Proceedings of the 21st Annual International Symposium on Computer Architecture*, pages 34–45, April 1994.

[49] Andrew B. Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi. ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration. In *Proceedings of the Conference on Design, Automation, and Test in Europe*, pages 423–428, April 2009.

[50] Mahmut Kandemir, Feihui Li, Mary Jane Irwin, and Seung Woo Son. A Novel Migration-Based NUCA Design for Chip Multiprocessors. In *Proceedings of the ACM/IEEE Conference on Supercomputing*, pages 1–12, November 2008.

[51] John Karidis, José E. Moreira, and Jaime Moreno. True Value: Assessing and Optimizing the Cost of Computing at the Data Center Level. In *Proceedings of the 6th ACM Conference on Computing Frontiers*, pages 185–192, May 2009.

[52] Taeho Kgil, Shaun D'Souza, Ali Saidi, Nathan Binkert, Ronald Dreslinski, Steven Reinhardt, Krisztian Flautner, and Trevor Mudge. PicoServer: Using 3D Stacking Technology to Enable a Compact Energy Efficient Chip Multiprocessor. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 117–128, October 2006.

[53] Changkyu Kim, Doug Burger, and Stephen W. Keckler. An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 211–222, October 2002.

[54] John Kim. Low-Cost Router Microarchitecture for On-Chip Networks. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 255–266, December 2009.

[55] John Kim, William J. Dally, and Dennis Abts. Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, pages 126–137, June 2007.

[56] Yoshinori Kohama, Yasufumi Sugimori, Shotaro Saito, Yohei Hasegawa, Toru Sano, Kazutaka Kasuga, Yoichi Yoshida, Kiichi Niitsu, Noriyuki Miura, Hideharu Amano, and

Tadahiro Kuroda. A Scalable 3D Processor by Homogeneous Chip Stacking with Inductive-Coupling Link. In *Proceedings of the Symposium on VLSI Circuits*, pages 94–95, June 2009.

[57] Rajesh Kumar and Glenn Hinton. A Family of 45nm IA Processors. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 58–59, June 2009.

[58] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 469–480, December 2009.

[59] Kevin Lim, Parthasarathy Ranganathan, Jichuan Chang, Chandrakant Patel, Trevor Mudge, and Steven Reinhardt. Understanding and Designing New Server Architectures for Emerging Warehouse-Computing Environments. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, pages 315–326, June 2008.

[60] Yong Liu, Wing Luk, and Daniel Friedman. A Compact Low-Power 3D I/O in 45nm CMOS. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 142–144, February 2012.

[61] Gabriel H. Loh. A Modular 3D Processor for Flexible Product Design and Technology Migration. In *Proceedings of the 5th Conference on Computing Frontiers*, pages 159–170, May 2008.

[62] Gabriel H. Loh, Yuan Xie, and Bryan Black. Processor Design in 3D Die-Stacking Technologies. *IEEE Micro*, 27(3):31–48, May-June 2007.

[63] Pejman Lotfi-Kamran, Michael Ferdman, Daniel Crisan, and Babak Falsafi. TurboTag: Lookup Filtering to Reduce Coherence Directory Power. In *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 377–382, August 2010.

[64] Pejman Lotfi-Kamran, Boris Grot, and Babak Falsafi. NOC-Out: Microarchitecting a Scale-Out Processor. In *Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 177–187, December 2012.

[65] Pejman Lotfi-Kamran, Boris Grot, Michael Ferdman, Stavros Volos, Onur Kocberber, Javier Picorel, Almutaz Adileh, Djordje Jevdjic, Sachin Idgunji, Emre Ozer, and Babak Falsafi. Scale-Out Processors. In *Proceedings of the 39th Annual International Symposium on Computer Architecture*, pages 500–511, June 2012.

[66] Rick Merritt. Moore's Law Dead by 2022, Expert Says. http://www.eetimes.com/document.asp?doc_id=1319330, August 2013.

[67] Michael Loughran. MADE IN IBM LABS: IBM Cools 3-D Chips with H2O. http://www-03.ibm.com/press/us/en/pressrelease/24385.wss, June 2008.

[68] George Michelogiannakis, James Balfour, and William J. Dally. Elastic-Buffer Flow Control for On-Chip Networks. In *Proceedings of the 15th IEEE International Symposium on High-Performance Computer Architecture*, pages 151–162, February 2009.

[69] Gordon E. Moore. Cramming More Components Onto Integrated Circuits. *Electronics*, 38(8):114–117, April 1965.

[70] Thomas Moscibroda and Onur Mutlu. A Case for Bufferless Routing in On-Chip Networks. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, pages 196–207, June 2009.

[71] Andreas Moshovos, Gokhan Memik, Babak Falsafi, and Alok Choudhary. JETTY: Filtering Snoops for Reduced Energy Consumption in SMP Servers. In *Proceedings of the 7th IEEE International Symposium on High Performance Computer Architecture*, pages 85–96, January 2001.

[72] Naveen Muralimanohar, Rajeev Balasubramonian, and Norman P. Jouppi. Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 3–14, December 2007.

[73] NVIDIA. NVIDIA Tesla Computing Processor. http://www.nvidia.com/docs/IO/43395/NV_DS_Tesla_C1060_US_Jan10_lores_r1.pdf, January 2010.

[74] Taecheol Oh, Hyunjin Lee, Kiyeon Lee, and Sangyeun Cho. An Analytical Model to Study Optimal Area Breakdown between Cores and Caches in a Chip Multiprocessor. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pages 181–186, May 2009.

[75] Chandrakant D. Patel and Amip J. Shah. Cost Model for Planning, Development and Operation of a Data Center. Technical Report HPL-2005-107, HP Laboratories Palo Alto, June 2005.

[76] Fred J. Pollack. New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies (Keynote Address)(Abstract Only). In *Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture*, page 2, November 1999.

[77] John Poulton. An Embedded DRAM for CMOS ASICs. In *Proceedings of the 17th Conference on Advanced Research in VLSI*, pages 288–302, September 1997.

[78] Parthasarathy Ranganathan. The Birth of Project Moonshot. http://www8.hp.com/hpnext/posts/birth-project-moonshot#.UZvPbLVaxPM, April 2013.

[79] Vijay Janapa Reddi, Benjamin C. Lee, Trishul Chilimbi, and Kushagra Vaid. Web Search Using Mobile Cores: Quantifying and Mitigating the Price of Efficiency. In *Proceedings of the 37th Annual International Symposium on Computer Architecture*, pages 314–325, June 2010.

[80] SAVVIS. SAVVIS Sells Assets Related to Two Data Centers for $200 Million. http://www.savvis.com/en-US/Company/News/Press/Pages/SAVVIS%20Sells%20Assets%20Related%20to%20Two%20Data%20Centers%20for%20200%20Million.aspx, June 2007.

[81] Jinuk Luke Shin, Kenway Tam, Dawei Huang, Bruce Petrick, Ha Pham, Changku Hwang, Hongping Li, Alan Smith, Timothy Johnson, Francis Schumacher, David Greenhill, Ana Sonia Leon, and Allan Strong. A 40nm 16-Core 128-Thread CMT SPARC SoC Processor. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 98–99, February 2010.

[82] Vason P. Srini and Linda G. Bushnell. *A Crossbar System for Multiprocessors*. University of California, Berkeley, Computer Science Division, 1st edition, 1989.

[83] Tilera. TILE-Gx 3036 Specifications. http://www.tilera.com/sites/default/files/productbriefs/Tile-Gx%203036%20SB012-01.pdf, 2011.

[84] Tony Smith. Intel, AMD and Apple Test On-Chip Water Cooling Tech. http://www.theregister.co.uk/2003/10/07/intel_amd_and_apple_test1, October 2003.

[85] Jim Turley. Cortex-A15 "Eagle" Flies the Coop. *Microprocessor Report*, 24(11):1–11, November 2010.

[86] W. Pitt Turner and John H. Seader. Dollars per kW plus Dollars per Square Foot are a Better Datacenter Cost Model than Dollars per Square Foot Alone. White Paper, June 2005.

[87] Stavros Volos, Ciprian Seiculescu, Boris Grot, Naser Khosro Pour, Babak Falsafi, and Giovanni De Micheli. CCNoC: Specializing On-Chip Interconnects for Energy Efficiency in Cache-Coherent Servers. In *Proceedings of the 6th IEEE/ACM International Symposium on Networks-on-Chips*, pages 67–74, May 2012.

[88] Hangsheng Wang, Li-Shiuan Peh, and Sharad Malik. Power-Driven Design of Router Microarchitectures in On-Chip Networks. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 105–116, December 2003.

[89] Thomas F. Wenisch, Roland E. Wunderlich, Michael Ferdman, Anastassia Ailamaki, Babak Falsafi, and James C. Hoe. SimFlex: Statistical Sampling of Computer System Simulation. *IEEE Micro*, 26(4):18–31, July-August 2006.

[90] Bob Wheeler. Tilera Sees Opening in Clouds. *Microprocessor Report*, 25(7):13–16, July 2011.

[91] Art Wittmann. ARM, AppliedMicro Challenge Intel Xeon. http://www.networkcomputing.com/next-generation-data-center/news/servers/arm-appliedmicro-challenge-intel-xeon/240152678, April 2013.

[92] David A. Wood and Mark D. Hill. Cost-Effective Parallel Computing. *Computer*, 28(2):69–72, February 1995.

[93] Roland E. Wunderlich, Thomas F. Wenisch, Babak Falsafi, and James C. Hoe. SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling. In *Proceedings of the 30th Annual International Symposium on Computer Architecture*, pages 84–97, June 2003.

[94] Aibin Yu, Aditya Kumar, Soon Wee Ho, Hnin Wai Yin, John H. Lau, Khong Chee Houe, Sharon Lim Pei Siang, Xiaowu Zhang, Da-Quan Yu, Nandar Su, Michelle Chew Bi-Rong, Jong Ming Ching, Tan Teck Chun, Vaidyanathan Kripesh, Charles Lee, Jun Pin Huang, James Chiang, Scott Chen, Chi-Hsin Chiu, Chang-Yueh Chan, Chin-Huang Chang, Chih-Ming Huang, and Cheng-Hsu Hsiao. Development of Fine Pitch Solder Microbumps for 3D Chip Stacking. In *Proceedings of the 10th Electronics Packaging Technology Conference*, pages 387–392, December 2008.

[95] Michael Zhang and Krste Asanović. Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture*, pages 336–345, June 2005.

[96] Li Zhao, Ravi Iyer, Srihari Makineni, Jaideep Moses, Ramesh Illikkal, and Donald Newell. Performance, Area and Bandwidth Implications on Large-Scale CMP Cache Design. In *Proceedings of the Workshop on Chip Multiprocessor Memory Systems and Interconnects*, February 2007.

Pejman Lotfi-Kamran **September 2013**

Contact Information

EPFL IC ISIM PARSA
INJ 235 (Batiment INJ)
Station 14
CH-1015 Lausanne

http://parsa.epfl.ch/ plotfi/
pejman.lotfikamran@epfl.ch
Cell: (78) 668-4832
Office: (21) 693-5298

Research Interests

Computer architecture. More specifically, I am interested in cross-stack and technology-driven innovations for improving performance and energy efficiency of computer systems for emerging applications including big data.

Education

- **Ph.D. in Computer Science.** Swiss Federal Institute of Technology in Lausanne (EPFL) — Sep. 2013
  Advisor: Prof. Babak Falsafi
  Thesis: *Scale-Out Processors*
- **M.S. in Electrical and Computer Engineering.** University of Tehran — Feb. 2005
  Advisor: Prof. Zainalabedin Navabi
  Co-Advisor: Prof. Mehran Massoumi
  Thesis: *An Efficient Data Structure for RTL Representation*
  GPA: 18.78/20.00 (graduated with honors)
- **B.S. in Electrical and Computer Engineering.** University of Tehran — Sep. 2002
  Advisor: Prof. Zainalabedin Navabi
  Thesis: *Implementation of a VHDL-AMS-to-CHIRE compiler*
  GPA: 17.40/20.00 (graduated with honors)

Honors and Awards

- **Intel Ph.D. Fellowship Award.** 2012-2013 Academic Year — 2012
- **Paper Award** from the European Network of Excellence on High Performance and Embedded Architecture and Compilation (HiPEAC) for "NOC-Out: Microarchitecting a Scale-Out Processor" — 2012
- **Paper Award** from the European Network of Excellence on High Performance and Embedded Architecture and Compilation (HiPEAC) for "Scale-Out Processors" — 2012
- **Paper Award** from the School of Computer and Communication Sciences at EPFL for "Cuckoo Directory: A Scalable Directory for Many-Core Systems" — 2011
- **Paper Award** from the European Network of Excellence on High Performance and Embedded Architecture and Compilation (HiPEAC) for "Cuckoo Directory: A Scalable Directory for Many-Core Systems" — 2011
- **Best Student Paper Finalist** at the 17th International Symposium on High Performance Computer Architecture (HPCA) for "Cuckoo Directory: A Scalable Directory for Many-Core Systems" — 2011
- **Best Paper Award** from the 13th Iranian Conference on Electrical Engineering for "Improving Logic-Level Representation of BMD/TED Diagrams" — 2005
- **Dean's Honored Graduate.** College of Engineering, University of Tehran — 2002
  Ranked 3^{rd} among graduates of the 2002 class
- **Faculty of Engineering (FOE) Award.** University of Tehran, Computer Engineering track — 2002
  Annually awarded to the top three students in each track by College of Engineering
- **Ranked 8^{th}** among 5,040 participants of Iran's national M.S. entrance exam. Computer Engineering track — 2002
- **Ranked 2^{nd}** among 50 computer engineering graduated students of the 1998 class — 2002

- **Ranked 11**th among 30,000 participants of Iran's Region 3 B.S. entrance exam. Mathematics and Physics track 1998
- **Ranked 1**st among 80 high-school graduates of the 1994 class 1998

PUBLICATIONS **Conference Papers**

1. **P. Lotfi-Kamran**, B. Grot, and B. Falsafi, "NOC-Out: Microarchitecting a Scale-Out Processor," in *International Symposium on Microarchitecture (MICRO)*, pp. 177–187, December 2012.
2. D. Milojevic, S. Idgunji, D. Jevdjic, E. Ozer, **P. Lotfi-Kamran**, A. Panteli, A. Prodromou, C. Nicopoulos, D. Hardy, B. Falsafi, and Y. Sazeides, "Thermal Characterization of Cloud Workloads on a Power-Efficient Server-on-Chip," in *International Conference on Computer Design (ICCD)*, pp. 175–182, October 2012.
3. **P. Lotfi-Kamran**, B. Grot, M. Ferdman, S. Volos, O. Kocberber, J. Picorel, A. Adileh, D. Jevdjic, S. Idgunji, E. Ozer, and B. Falsafi, "Scale-Out Processors," in *International Symposium on Computer Architecture (ISCA)*, pp. 500–511, June 2012.
4. M. Hosseinabady, **P. Lotfi-Kamran**, J. Mathew, S. Mohanty, and D. Pradhan, "Single-Event Transient Analysis in High Speed Circuits," in *International Symposium on Electronic System Design (ISED)*, pp. 112–117, December 2011.
5. M. Ferdman, **P. Lotfi-Kamran**, K. Balet, and B. Falsafi, "Cuckoo Directory: A Scalable Directory for Many-Core Systems," in *International Symposium on High Performance Computer Architecture (HPCA)*, pp. 169–180, February 2011. (**Selected by the program committee for the best student paper session**)
6. **P. Lotfi-Kamran**, M. Ferdman, D. Crisan, and B. Falsafi, "TurboTag: Lookup Filtering to Reduce Coherence Directory Power," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 377–382, August 2010.
7. A.-M. Rahmani, I. Kamali, **P. Lotfi-Kamran**, A. Afzali-Kusha, and S. Safari, "Negative Exponential Distribution Traffic Pattern for Power/Performance Analysis of Network on Chips," in *International Conference on VLSI Design (VLSID)*, pp. 157–162, January 2009.
8. **P. Lotfi-Kamran**, M. Daneshtalab, C. Lucas, and Z. Navabi, "BARP–A Dynamic Routing Protocol for Balanced Distribution of Traffic in NOCs," in *Design, Automation and Test in Europe (DATE)*, pp. 541–546, March 2008.
9. **P. Lotfi-Kamran**, A.-A. Salehpour, A.-M. Rahmani, and A. Afzali-Kusha, "Stall Power Reduction in Pipelined Architecture Processors," in *International Conference on VLSI Design (VLSID)*, pp. 541–546, January 2008.
10. **P. Lotfi-Kamran**, M. Massoumi, M. Mirzaei, and Z. Navabi, "Enhanced TED: A New Data Structure for RTL Verification," in *International Conference on VLSI Design (VLSID)*, pp. 481–486, January 2008.
11. M. Hosseinabady, M. H. Neishaburi, **P. Lotfi-Kamran**, and Z. Navabi, "A UML Based System Level Failure Rate Assessment Technique for SoC Designs," in *VLSI Test Symposium (VTS)*, pp. 243–247, May 2007.
12. M. Hosseinabady, **P. Lotfi-Kamran**, G. Di Natale, S. Di Carlo, A. Benso, and P. Prinetto, "Single-Event Upset Analysis and Protection in High Speed Circuits," in *European Test Symposium (ETS)*, pp. 29–34, May 2006.
13. M. Hosseinabady, **P. Lotfi-Kamran**, P. Riahi, F. Lombardi, and Z. Navabi, "A Flow Graph Technique for DFT Controller Modification," in *International System-on-Chip Conference (SOCC)*, pp. 55–60, September 2005.
14. A. Hooshmand, S. Shamshiri, M. Alisafaee, B. Alizadeh, **P. Lotfi-Kamran**, M. Naderi, and Z. Navabi, "Binary Taylor Diagrams: An Efficient Implementation of Taylor Expansion Diagrams," in *International Symposium on Circuits and Systems (ISCAS)*, vol. 1, pp. 424–427, May 2005.
15. **P. Lotfi-Kamran**, H. Shojaei, H. Parandeh-Afshar, M. Naderi, and Z. Navabi, "Improving Logic-Level Representation of BMD/TED Diagrams," in *Iranian Conference on Electrical Engineering (ICEE)*, pp. 448–453, May 2005. (**Recognized as best paper by the program committee**)
16. **P. Lotfi-Kamran**, M. Hosseinabady, H. Shojaei, M. Massoumi, and Z. Navabi, "TED+: A Data Structure for Microprocessor Verification," in *Asia South Pacific Design Automation Conference (ASP–DAC)*, vol. 1, pp. 567–572, January 2005.

Journal Papers

1. B. Grot, D. Hardy, **P. Lotfi-Kamran**, B. Falsafi, C. Nicopoulos, and Y. Sazeides, "Optimizing Data-Center TCO with Scale-Out Processors," in *IEEE Micro, Special Issue on Energy-Aware Computing*, vol. 32, no. 5, pp. 52–63, September/October 2012.
2. **P. Lotfi-Kamran**, A.-M. Rahmani, M. Daneshtalab, A. Afzali-Kusha, and Z. Navabi, "EDXY–A Low-Cost Congestion-Aware Routing Algorithm for Network-on-Chips," in *Elsevier Journal of Systems Architecture – Embedded Systems Design (JSA–ESD)*, vol. 56, no. 7, pp. 256–264, July 2010.
3. M. Hosseinabady, **P. Lotfi-Kamran**, F. Lombardi, and Z. Navabi, "Low Overhead DFT Using CDFG by Modifying Controller," in *IET Computers & Digital Techniques (IET–CDT)*, vol. 1, no. 4, pp. 322–333, July 2007.
4. M. Hosseinabady, **P. Lotfi-Kamran**, and Z. Navabi, "Low Test Application Time Resource Binding for Behavioral Synthesis," in *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 12, no. 2, article 16, April 2007.
5. **P. Lotfi-Kamran** and Z. Navabi, "Improving Logic-Level Representation of Taylor Expansion Diagram Using Attributed Edges," in *Iranian Journal of Science and Technology (IJST), Transaction B, Technology*, vol. 30, no. B6, pp. 735–748, 2006.

Workshop & Poster Papers

1. S. Idgunji, D. Milojevic, E. Ozer, **P. Lotfi-Kamran**, D. Jevdjic, and B. Falsafi, "Performance and Efficiency of 3D-Stacked DRAM in a Multicore System," in *Workshop on 3D Integration – Applications, Technology, Architecture, Design, Automation and Test in Conjunction with Design, Automation and Test in Europe (DATE)*, March 2012.
2. E. Ozer, K. Flautner, S. Idgunji, A. Saidi, Y. Sazeides, B. Ahsan, N. Ladas, C. Nicopoulos, I. Sideris, B. Falsafi, A. Adileh, M. Ferdman, **P. Lotfi-Kamran**, M. Kuulusa, P. Marchal, and N. Minas, "EuroCloud: Energy-Conscious 3D Server-on-Chip for Green Cloud Services," in *Workshop on Architectural Concerns in Large Datacenters in Conjunction with International Symposium on Computer Architecture (ISCA)*, June 2010.
3. M. Hosseinabady, **P. Lotfi-Kamran**, and Z. Navabi, "Controller-Aware Hierarchical Test Generation and Testability Analysis," in *European Test Symposium (ETS)*, May 2005.
4. M. Alisafaee, **P. Lotfi-Kamran**, S. Shamshiri, H. Esmaeilzadeh, A. Pedram, and Z. Navabi, "MCBIST: A New Online BIST Scheme," in *Workshop on RTL and High Level Testing (WRTLT)*, pp. 85–90, November 2004.
5. M. Hosseinabady, **P. Lotfi-Kamran**, A. Pedram, and Z. Navabi, "A Binary Wavelet Test Compression," in *Workshop on RTL and High Level Testing (WRTLT)*, November 2004.
6. S. Shamshiri, H. Esmaeilzadeh, M. Alisafaee, **P. Lotfi-Kamran**, and Z. Navabi, "Test Instruction Set (TIS): An Instruction Level CPU Core Self-Testing Method," in *European Test Symposium (ETS)*, May 2004.

Research Experience

- **Research assistant.** *Parallel Systems Architecture (PARSA) Lab* Sep. 2008–Sep. 2013
  Swiss Federal Institute of Technology in Lausanne (EPFL)
  Advisor: Babak Falsafi

  Organization of Scale-Out Processors. *I proposed NOC-Out, a many-core organization for Scale-Out Processors that has low area overhead and provides fast access to the last-level cache (LLC) for delivering high performance. Today's many-core organizations force a compromise between performance and cost. Thus, many-core organizations based on a mesh interconnect have a modest area and wire cost, yet incur latency overheads through a many-hop topology. In contrast, many-core organizations based on richly connected topologies, such as a flattened butterfly, offer low latency at high area and wire cost. While existing many-core organizations offer an uneasy compromise between low area overhead and fast access to the LLC, NOC-Out offers both features simultaneously. The proposed organization is based on one simple and critical observation: there is almost no core-to-core communication in scale-out workloads. Based on this observation, this organization decouples cores and the last-level cache, eliminates all unneeded core-to-core links, and uses specialized core-to-LLC networks to connect cores to the last-level cache and vice versa. The bottom line is that NOC-Out delivers the performance of the state-of-the-art many-core organization with 1/10th of the area [MICRO'12].*

Scale-Out Processors. *I proposed a methodology for the design of highly efficient many-core processors for scale-out workloads. This research relies on two critical observations with regard to such many-core processors. First, large LLCs waste precious silicon real estate that could have been better used to integrate more cores. Second, the organization of a many-core processor has a significant impact on its performance. Existing many-core chips, such as those offered by Tilera, sacrifice much of the on-die real estate to LLC and employ a tiled organization that incurs a high on-chip communication overhead. In contrast, I proposed a many-core processor based on the notion of pods. A pod is a module that tightly couples many cores to a modestly sized LLC through a low-latency interconnect. The proposed processor integrates many pods wherein each pod is a self-contained server-on-a-chip running a full software stack. I formulated a methodology to determine the optimal number of cores and LLC capacity to integrate in a pod for peak throughput. The proposed design, called the Scale-Out Processor, delivers peak throughput in today's process technology and affords near-ideal scalability as the technology scales [ISCA'12].*

Area- and Energy-Efficient Coherence Directories for Many-Core CMPs. *We proposed the Cuckoo directory, an energy- and area-efficient scalable directory organization. Existing directory organizations suffer from the lack of energy or area efficiency at high core counts due to wide associative lookups or capacity overprovisioning. The Cuckoo directory, however, scales to high core counts without the energy costs of wide associative lookup and without gross capacity overprovisioning. The Cuckoo directory borrows heavily from Cuckoo Hashing, a dense-storage software hashing technique. Rather than significantly over-provisioning storage capacity to avoid storage conflicts in a traditional lookup table, the Cuckoo directory uses the Cuckoo Hashing algorithm to relocate conflicting entries within the directory to alternate non-conflicting locations. Leveraging the mathematically robust properties of Cuckoo Hashing enables compact and energy-efficient coherence directories with predictable asymptotic behavior and without degenerate cases, improving the state-of-the-art directory design without increasing complexity [HPCA'11].*

Lookup Filtering to Reduce Power Consumption of Coherence Directories. *To reduce the energy usage of coherence directories in many-core processors, I proposed the TurboTag filtering mechanism. Coherence directories dissipate a significant fraction of their power on unnecessary lookups when running commercial server and scientific workloads. These workloads have large working sets that are beyond the reach of on-chip caches of modern processors. Limited to capturing a small part of the working set, private caches retain cache blocks only for a short period of time before replacing them with new blocks. Moreover, coherence enforcement is a known performance bottleneck of multi-threaded software; hence, data sharing in optimized high-performance software is minimal. Consequently, the majority of the accesses to the coherence directory find no sharers in the directory because the data are not available in the on-chip private caches, effectively wasting power on the coherence checks. TurboTag reduces power consumption of coherence directories by filtering (almost all) needless directory lookups [ISLPED'10].*

CloudSuite Workloads. *CloudSuite is a benchmark suite for emerging scale-out applications. To enable full-system simulation of CloudSuite benchmarks, we brought up CloudSuite workloads on Flexus, our in-house, full-system simulator. I led the effort on bringing up CloudSuite workloads. We released the workloads to the broader research community.*

Flexus Full-System Simulator. *Flexus is a family of component-based C++ computer architecture simulators that enable full-system, timing-accurate simulation of uni- and multi-processor systems running unmodified commercial applications and operating systems. I contributed to the development of Flexus and also served as the coordinator of its mailing list.*

- **Research assistant.** *Computer-Aided Design (CAD) Lab* Jul. 2002–Aug. 2008
  University of Tehran

  Advisor: Zainalabedin Navabi

  Cost-Effective Globally Aware Dynamic Routing Protocols to Avoid Congestion in NOCs. *I proposed low-cost adaptive routing algorithms for network-wide congestion avoidance. This research enhanced conventional adaptive routings that only rely on local indicators for congestion estimation with low-cost, non-local (global) indicators [DATE'08, JSA–ESD'10].*

  Statistical Analysis for Soft Error Rate Estimation. *The effect of Single-Event Transients (SETs) on the system reliability is a big concern for ICs manufactured using advanced technologies. An SET in the combinational part of a circuit may propagate as a transient pulse to the input of a flip-flop and, consequently, gets latched, thus generating a soft error. Using the Probability Density Function (PDF) of an SET, we proposed a statistical method to compute the probability of soft errors considering dynamic behavior of a circuit [ETS'06, ISED'11].*

  Low-Overhead Controller-Aware Design for Test. *We proposed a low-cost design for test (DFT) that requires minor modifications to the controller of digital systems. In this research, we took advantage of existing data paths in digital systems to provide controllability and observability for the test process. Furthermore, we introduced additional data paths by altering states or adding new transitions in the controllers of digital systems. The proposed DFT considerably reduces the test application time by ignoring unnecessary control states in the test process [IET–CDT'07].*

Low Test Application Time Test Synthesis. *Increased density and the need to test for new types of defects in nanometer technologies have resulted in a tremendous increase in test application time. We presented a test synthesis mechanism to reduce test application time for testing the datapath of a digital system. The reduction in the test application time was achieved by applying a test time-aware, resource-sharing algorithm on a scheduled control data flow graph of a design [TODAES'07].*

Data Structure for Efficient RT-Level Representation. *Formal verification of microprocessors requires a mechanism for efficient representation and manipulation of both arithmetic and random Boolean functions. State-of-the-art representations can effectively represent arithmetic expressions at the word-level but are not memory efficient in representing bit-level logic expressions. In this research, I presented modifications to the state-of-the-art representations to improve the ability of bit-level logic representation while maintaining robustness in arithmetic word-level representation [ASP–DAC'05].*

VHDL-AMS Analyzer. *VHDL-AMS is a derivative of the hardware description language VHDL that includes analog and mixed-signal extensions (AMS) in order to define the behavior of analog and mixed-signal systems. I developed an analyzer to compile VHDL-AMS to CHIRE, our in-house intermediate format.*

Teaching Experience

- **Teaching assistant.** Held office hours, answered email/newsgroup queries, led review sessions, designed and graded student homework.

CS471: Advanced Multiprocessor Architecture. EPFL	Fall 2009 and Fall 2012
CS198: Information Technology Project. EPFL	Fall 2011
ECE367: Digital Logic Circuits. University of Tehran	Spring 2004–Spring 2005
ECE354: Telecommunications 1. University of Tehran	Fall 2003–Spring 2004
ECE046: Microprocessors Lab. University of Tehran	Spring 2003
ECE207: Microprocessors. University of Tehran	Spring 2002

- **Instructor.** Designed course, gave lectures, held office hours, answered email/newsgroup queries, designed and graded homework and exams.

ECE061: Discrete Mathematical Structures. University of Tehran	Spring 2007–Spring 2008
Operating Systems. University of Tehran, IT education series for professionals with no computer science background.	Spring 2003

Professional Activities

- **Reviewer.**

ACM Transactions on Embedded Computing Systems (TECS)	2013
International Symposium on High Performance Computer Architecture (HPCA)	2013
ACM Transactions on Architecture and Code Optimization (TACO)	2012
International Symposium on Microarchitecture (MICRO)	2012
Elsevier Journal of System Architecture (JSA)	2012

- **Member.**

ACM and ACM SIGARCH	Nov. 2008–date
IEEE and IEEE Computer Society	Jan. 2005–date

Tutorials

- **CloudSuite on Flexus.**

CSICC 2013, Tehran, Tehran, Iran	Mar. 2013
ISCA 2012, Portland, Oregon, USA	Jun. 2012

- **Flexus.**

IISWC 2010, Atlanta, Georgia, USA	Dec. 2010

Work Experience

- **Senior Design Engineer.** Public Switched Telephone Network (PSTN) Branch, Parstel Information and Telecommunication Technology Co. Inc. — Dec. 2005–Aug. 2008
- **Design Engineer.** Programmable Logic Controller (PLC) Branch, Arman Optimized Systems — Jun. 2001–Feb. 2002