Thesis for The Degree of Licentiate of Engineering

Design, Analysis, and Modeling of On-chip Interconnects for Large Scale Systems

Panagiotis Strikos

Division of Computer and Network Systems
Department of Computer Science & Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2026

Design, Analysis, and Modeling of On-chip Interconnects for Large Scale Systems

Panagiotis Strikos

Advisor:
Professor Ioannis Sourdis, Chalmers University of Technology

Co-Advisors:
Ahsen Ejaz, PhD, InfiniNode Technologies

Examiner:
Professor Pedro Petersen Moura Trancoso, Chalmers University of Technology

Discussion Leader:
Georgios Michelogiannakis, PhD, Microsoft

Department of Computer Science & Engineering
Division of Computer and Network Systems
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Gothenburg, Sweden 2026.

It is what it is...

Abstract

As the semiconductor industry faces increasing challenges in technology scaling, integrading more devices onto a single monolithic die incurs prohibitive manufacturing costs due to low yield. Chiplet-based systems have emerged as a cost-effective alternative, offering higher yield and enabling heterogeneous integration across different process technologies. However, chiplet-based systems come with performance overheads. In particular, inter-chiplet communication is constrained by limited bandwidth and higher latency overheads that are further exacerbated by the use of passive silicon interposers, which do not include any active components and limit the network design. Studying these overheads requires an infrastructure capable of simulating such systems. To that end, this thesis produces BZSim, an open-source framework for fast, large-scale microarchitectural simulation with detailed interconnect modeling. BZSim integrates BookSim into ZSim, combining accurate network modeling with fast parallel execution. It detects low-contention network traffic and analytically calculates its latency instead of simulating it, reducing the simulation overhead. This way, it achieves 2–4× faster simulations with a maximum normalized IPC error of 3–5% and an average packet latency error of 10–20%. Therefore it is an order of magnitude faster than gem5 while remaining just 3× slower than standalone ZSim. While the cost and yield advantages of chiplet-based designs are well established, their system performance overheads relative to monolithic designs have not been systematically studied. Using BZSim, this thesis conducts a comprehensive analysis of chiplet-based systems across a range of design parameters, measuring performance overheads relative to monolithic designs and their scalability to system and chiplet size. The study covers design alternatives in memory hierarchy organization, interconnect choices, and technology aspects. The analysis reveals that chiplet-based chips reduce recurring engineering costs by almost half compared to monolithic designs, at the cost of about a third of the monolithic performance. The inter-chiplet interconnect identified as a key source of this overhead, motivating a targeted network solution. For this reason, this thesis proposes PRISM: a new network for multi-chiplet chips on passive silicon interposers. PRISM introduces a new topology which offeres low hop count for inter-chiplet traffic, and a deterministic routing algorithm with virtual-network separation for deadlock-free routing. It also improves chiplet-to-chiplet bandwidth utilization through selective flit compression, applied dynamically per flit only when inter-chiplet congestion is detected, minimizing compression overheads. Evaluated on a 256-core system with 16 chiplets, PRISM improves inter-chiplet packet latency by up to 5.8× and overall system performance by up to 17.4%, with selective compression outperforming always-on compression by up to 6.5%.

Keywords: Chiplet-based systems, Network-on-Chip, Microarchitectural simulation, In-network compression

Acknowledgments

First and foremost, I would like to thank my supervisor Yiannis Sourdis and co-supervisor Ahsen Ejaz for their continuous guidance, support, and invaluable feedback throughout this work. My deepest gratitude goes to my officemates Konstantinos, Magnus, and especially Neethu, with whom I had the pleasure of collaborating. The discussions, shared ideas, and camaraderie you all brought to everyday academic life made this journey far more enjoyable and meaningful. I would also like to thank all my other current and former colleagues for the stimulating and friendly environment they helped create.

Finally, to my family and friends: thank you for always being there for me.
Σας ευχαριστώ για όλα.

This work is supported by the Swedish Foundation for Strategic Research (contract number CHI19-0048) under the PRIDE project.

List of Publications

Appended publications

This thesis is based on the following publications:

[I] **Panagiotis Strikos**, Ahsen Ejaz, and Ioannis Sourdis
BZSim: Fast, Large-scale Microarchitectural Simulation with Detailed Interconnect Modeling
2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS).

[II] Neethu Bal Mallya, **Panagiotis Strikos**, Bhavishya Goel, Ahsen Ejaz, and Ioannis Sourdis
A Performance Analysis of Chiplet-Based Systems
2025 Design, Automation and Test in Europe Conference and Exhibition (DATE), Lyon, France, Mar 2025.

[III] **Panagiotis Strikos**, Ahsen Ejaz, and Ioannis Sourdis
PRISM: Reducing Performance Overhead of Multi-chiplet Chip Interconnects
Under Submission.

Contents

Chapter 1

Introduction

Since the early 2000s, the semiconductor industry has relied on multicore scaling, increasing the number of cores and on-chip resources to continue delivering performance growth. At the same time, technology scaling has become increasingly challenging, and integrating more devices on a monolithic chip incurs substantial costs: large monolithic dies have low manufacturing yield, making them economically unviable at scale [1, 2]. Building chips out of multiple smaller chiplets offers a cheaper, higher-yield alternative while also enabling better scalability, modular design, and heterogeneous integration of components across different process technologies. In the past years, chiplet-based systems have been widely adopted by the industry, as seen in AMD's EPYC and RYZEN families [3]. More recent products, such as AMD MI300 [4] and Intel Sapphire Rapids [5], combine many CPU and GPU chiplets alongside high-bandwidth memory (HBM) nodes.

Despite their manufacturing advantages, multi-chiplet chips come with performance overheads. Due to their large size, such systems inevitably use multiple Non-Uniform Memory Access (NUMA) nodes, leading to performance bottlenecks that arise from varying memory access latencies. Beyond NUMA effects, inter-chiplet communication introduces further overheads: exchanging data between chiplets requires traversing longer links via a silicon interposer, adding extra latency, while the available off-chiplet bandwidth is inherently constrained by the limits of chiplet integration. Passive silicon interposers, often selected due to their lower cost [6], make these latency overheads more severe.

This thesis investigates the performance overheads of multi-chiplet systems, with a particular focus on their interconnection networks. It introduces a new, more scalable simulation infrastructure for studying large-scale multi-chiplet systems, measures and analyzes their performance overheads compared to monolithic designs, and proposes a network solution to mitigate inter-chiplet communication bottlenecks.

The remainder of the introductory chapter is structured as follows: Section 1.1 outlines the research problem, and Section 1.2 presents the thesis objectives and contributions.

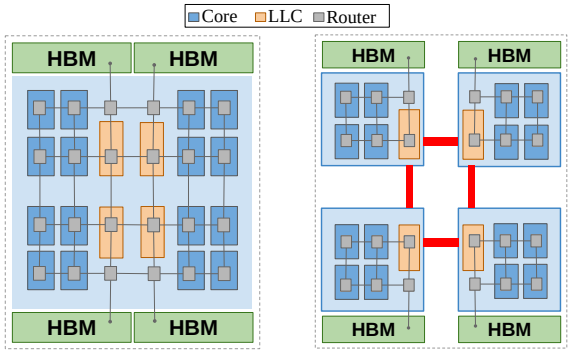


Figure 1.1: Example of a 16-core monolithic chip and its equivalent four-chiplet design, showing cores, LLC banks, HBM controllers, and their on-chip network.

1.1 Problem Statement

Multi-chiplet chips have emerged as a practical response to the scaling limitations of monolithic designs: rather than a single large chip, the system is composed of multiple smaller chiplets interconnected through chiplet-to-chiplet links, as illustrated in Figure 1.1. This partitioning offers improved yield and lower manufacturing costs [1], but comes with performance trade-offs. Chiplet-based systems suffer from overheads related to remote memory accesses and inter-chiplet communication. This thesis focuses on the latter: data exchanged between chiplets must traverse longer interconnect paths with limited bandwidth, introducing latency and throughput constraints that directly limit system performance. These overheads are the core problem this thesis addresses. However, tackling them first required developing a simulation infrastructure capable of modeling chiplet systems at scale, then a systematic performance analysis to understand where performance is lost, and finally targeted network solutions to mitigate the identified bottlenecks.

1.1.1 The Simulation Scalability Challenge

Computer architecture research relies extensively on simulation, which enables the study and validation of new designs. Including Network-on-Chip (NoC) models into those simulations allows the assessment of how the on-chip interconnect affects overall system performance and exploration of the benefits of different NoC designs.

However, detailed microarchitectural simulation is computationally expensive, and the cost scales with system complexity. Simulators that prioritize accuracy, such as gem5 [7], become impractically slow for the scale at which chiplet systems operate. Faster alternatives, such as ZSim [8] and Sniper [9], forgo network modeling entirely to achieve their speed. Neither is well-suited for studying chiplet architectures, which require both simulation speed and interconnect accuracy. This thesis addresses this gap by developing a simulation framework that integrates accurate network modeling into a fast parallel simulator, enabling the large-scale chiplet studies that follow.

1.1.2 Performance Analysis of Chiplet-Based Systems

Despite the well-established cost and yield advantages of chiplet-based designs [1], their system performance overheads relative to monolithic chips had not been systematically studied. The design space is large: chiplet size, network bandwidth, last-level cache organization, and interposer type all influence system performance and interact in non-trivial ways. A comprehensive analysis covering these parameters is needed to measure and analyze the system performance overhead of a chiplet-based system. This thesis conducts such an analysis, revealing the inter-chiplet network as a primary performance overhead and establishing the motivation for the network solution presented next.

1.1.3 Reducing Inter-Chiplet Communication Overhead

Once the performance bottlenecks of chiplet-based systems are understood, addressing them requires targeted network solutions. The inter-chiplet network faces constraints that differ fundamentally from those of intra-chiplet networks: chip-to-chip links carry higher latency, bandwidth across chiplet boundaries is limited, and passive silicon interposers provide no active routing support [6]. Standard network topologies and routing algorithms, designed for monolithic chips, do not account for these constraints. Most proposals for multi-chiplet systems assume active interposer components that are absent in the passive interposer case [2, 10, 11]. A network solution that jointly addresses topology, routing, and bandwidth utilization under these constraints is therefore needed. This thesis proposes a new network design for multi-chiplet chips on passive silicon interposers, targeting the latency and bandwidth constraints of inter-chiplet communication.

1.2 Thesis Objectives and Contributions

1.2.1 Fast, Large-Scale Simulation with Detailed Interconnect Modeling

Objective:

The first objective of this thesis is to develop a microarchitectural simulator that combines the speed of fast parallel simulators with the detailed interconnect modeling of accurate network simulators.

Related Work:

Plenty of microarchitectural and system simulators exist. gem5 [7], one of the most widely used, offers detailed interconnect modeling, but it is sequential, which makes it relatively slow. ZSim [8] and Sniper [9] are parallel and orders of magnitude faster, but neither models on-chip interconnects. On the network simulation side, BookSim [12] is a widely used cycle-accurate simulator that is flexible in terms of topologies, allocators, and routing algorithms. Garnet [13], integrated into gem5, uses a more abstract pipeline approach, making it faster but less detailed than BookSim. Integrating either into a microarchitectural simulator substantially decreases simulation speed [14].

Analytical methods offer a lighter-weight alternative [15, 16], though their accuracy degrades significantly under network congestion.

Accurately modeling the network is increasingly important as NoC designs have evolved substantially, with recent high-performance designs such as ShortPath [17] and FastTrackNoC [18] approaching the theoretical minimum latency through fine-grained router bypassing. These designs are used in this work to assess the impact of high-performance NoCs on system performance.

Thesis Approach:

BookSim is integrated into ZSim to produce BZSim, a fast, large-scale microarchitectural simulator with detailed interconnect modeling. To make the network simulation lightweight, BZSim is based on the conjecture that the latency of low-contention traffic can be calculated analytically relatively accurately. Traffic is classified by its estimated level of contention using a user-defined threshold. For traffic below that threshold, latency is calculated analytically rather than simulated. This reduces the amount of traffic that needs to be simulated, lowering the workload of network modeling. In addition, individual parts of the network are monitored separately and their evaluation is skipped when they are not used. The combination of these two mechanisms offers a knob for trading simulation accuracy for speed.

Thesis Contributions: In line with this objective, **Paper I** developed BZSim and made the following contributions:

- Integrated BookSim, an accurate network simulator into ZSim, a fast parallel system simulator to produce BZSim. BZSim is an open-source fast, large-scale microarchitectural simulator with detailed on-chip interconnect modeling.
  - BZSim is an order of magnitude faster than an optimistic gem5 setup with point-to-point interconnects, and 3× slower than ZSim, which does not model on-chip networks.
- Developed a novel mechanism that identifies low-contention network traffic and skips its simulation using an analytically calculated latency. This enables a tunable tradeoff between simulation speed and accuracy.
  - BZSim skips network simulation for 20–85% of traffic depending on the configured threshold
  - BZSim achieves 2–4× speedup with a maximum error of 3–5% in normalized IPC and 10–20% in average packet latency.
- Modeled and evaluated FastTrackNoC, and ShortPath, two state-of-the-art high-speed NoC designs and their impact on system performance on a 32-core system.
  - FastTrackNoC on average improves system performance by 18% over a baseline NoC, reducing the average packet latency of the network to 49% of the baseline.
  - ShortPath on average improves system performance by 10% over a baseline NoC, reducing the average packet latency of the network to 69% of the baseline.

1.2.2 Performance Analysis of Chiplet-Based Systems

Objective:
The second objective of this thesis is to analyze the performance of chiplet-based systems and compare it to monolithic chips. The analysis covers a range of design parameters, including system and chiplet size, memory hierarchy organization, interconnect configuration, and technology aspects.

Related Work:
Commercial chiplet designs such as the AMD MI300 [4], and Intel Sapphire Rapids [5] demonstrate the industry shift away from monolithic dies. The cost and yield benefits of this approach have been thoroughly quantified [1].

Prior work has studied design choices that affect inter-chiplet networks, including the choice of silicon interposer [6,19], and inter-chiplet link placement and width [20–23]. The effect of memory organization on inter-chiplet traffic is equally significant: multi-chiplet chips are inherently NUMA [24–26], and both data placement and cache organization determine how much traffic must cross chiplet boundaries. Yet the combined effect of these design choices on the performance of a multi-chiplet system compared to a monolithic had not been systematically evaluated.

Thesis Approach:
Performance is evaluated by comparing three system configurations: a chiplet-based design, its monolithic counterpart of equivalent core count, and an ideal chiplet baseline that eliminates remote memory accesses to serve as a performance upper bound. The analysis covers several design parameters: system and chiplet size, LLC organization, intra-chiplet NoC datapath width, and silicon interposer type. For LLC organization, two configurations are studied: a sliced design, where the cache is distributed across chiplets, and a private design, where each chiplet maintains its own local LLC. BZSim is extended to support the modeling of chiplet-based architectures with detailed cycle-accurate interconnect and memory system models. The performance analysis is complemented by a cost evaluation using the Feng-Ma chiplet actuary model [1], to put the performance overheads in perspective against the cost benefits of chiplet-based designs.

Thesis Contributions: In line with this objective, **Paper II** conducted a systematic performance analysis and made the following contributions:

- Extended BZSim to model large-scale chiplet-based architectures with detailed cycle-accurate models of their interconnection network and memory system.
- Conducted the first thorough analysis of performance overheads and cost-performance tradeoffs of chiplet-based chips compared to monolithic. Despite costing only 55% as much, chiplet-based systems:
  - Achieve only 43–75% of monolithic IPC (58% on average), while an ideal chiplet system with perfect data locality outperforms monolithic by 24%.

  - Suffer 3.7× higher average packet latency and 40% higher average memory access time.

- Analyzed technological aspects that determine key system parameters, including inter-chiplet link lengths and delays, microbump budgets, and the yield and cost of chiplet-based versus monolithic chips, confirming that chiplet-based designs reduce recurring engineering costs by nearly half.
- Investigated the impact of key design parameters on system performance, including system size, chiplet size, LLC organization, NoC datapath width, and silicon interposer type.
  - Private LLC improves the system performance by about 30% compared to sliced LLC in multi-chiplet systems.
  - Active interposers recover most of the performance overhead, bringing multi-chiplet systems to within 3% of monolithic performance, but at a 61% higher cost due to their lower yield.
  - Wider intra-chiplet NoC links improve the network throughput, reducing average memory access time across both chiplet and monolithic configurations.
  - The performance gap remains relatively stable as system size increases.
  - For a constant system size, performance decreases with finer chiplet granularity: systems with 2, 4, and 8 cores per chiplet achieve 76%, 58%, and 44% of monolithic IPC respectively.

Appendix A of **Paper II** provides additional results on the cost analysis of various design options for chiplet-based systems, which were not included in the paper due to space constraints.

1.2.3 Reducing Inter-Chiplet Communication Overhead

Objective:

Paper II showed that inter-chiplet links are a key performance bottleneck in multi-chiplet systems. Crossing the chiplet boundaries has bandwidth limitations and latency overheads. The bandwidth limitation stems from the number of microbumps a chiplet can accommodate, which is constrained by the chiplet size and density allowed by the packaging technology. Moreover, cross-chiplet connections via a silicon interposer, especially a passive one, add extra delay. The third objective of this thesis is to alleviate these overheads by introducing a novel network design for multi-chiplet chips on passive silicon interposers.

Related Work:

Several works have explored network topologies for multi-chiplet systems based on active interposers [2, 10, 11, 20], which can embed routing logic but are significantly more expensive than passive alternatives [6]. For passive interposers, Feng et al. [27] showed that grouping inter-chiplet links at chiplet edges enables high-radix topologies, delivering up to 2× saturation throughput

and 45% lower latency than a conventional 2D-mesh multi-chiplet network. Network-on-Die (NoD) [28] moves all inter-chiplet routers onto a standalone routing chiplet, though this does not scale well as chiplet count increases.

Regardless of topology, combining intra- and inter-chiplet networks introduces deadlock risks. Common deadlock avoidance strategies include turn restriction [29, 30] and injection gating [31] which break dependency cycles but either reduce path diversity or throttle traffic under congestion. Alternatively, virtual-network (VN) separation offers better performance [32–34], but existing VN-based schemes target active interposers.

Compression offers a complementary direction to address the limited C2C bandwidth. Several designs compress all network traffic at every network interface or router [35, 36], adding latency overhead even at low network load. Selective approaches compress only congested paths to avoid this penalty [37–39]. All prior compression work, however, targets monolithic chips; no prior work has applied compression to inter-chiplet links.

Thesis Approach:

PRISM introduces a new network for multi-chiplet chips on a passive silicon interposer which alleviates the bandwidth limitations and latency overheads of inter-chiplet communication. It describes a new topology that uses dedicated edge routers at chiplet boundaries interconnected within each chiplet with bypass links to reduce hop count for inter-chiplet traffic. In addition, it utilizes more effectively the chiplet-to-chiplet (C2C) bandwidth by compressing inter-chiplet traffic. The compression decision is made dynamically per flit and chosen only when C2C congestion is detected, thereby minimizing compression overheads. A deterministic routing algorithm with virtual-network separation provides deadlock-free routing.

Thesis Contributions: In line with this objective, **Paper III** introduced PRISM and made the following contributions:

- A new topology for multi-chiplet chips with passive silicon interposer is introduced.
  - Dedicated edge routers at chiplet boundaries handle inter-chiplet traffic, avoiding interference with intra-chiplet traffic.
  - Peripheral bypass links connect edge routers within a chiplet, reducing hop count for inter-chiplet traffic.
- A deterministic deadlock-free routing algorithm for the above topology is developed by adapting a two-virtual-network separation scheme [33], originally designed for active interposers, to fit the constraints of passive silicon interposers.
- PRISM is the first to apply compression exclusively for C2C communication on multi-chiplet chips. Its edge routers are designed to selectively compress inter-chiplet traffic based on their latency criticality and router congestion, thereby increasing effective C2C bandwidth with reduced latency overheads.
  - 11–33% of network traffic crosses chiplet boundaries and is eligible for compression, achieving a compression ratio of 1.1–4.9×.

  - Selective compression skips compression in 18–44% of cases, improving system performance over always-compressing by up to 6.5%.
- PRISM is evaluated on a 256-core system with 16 chiplets, demonstrating:
  - Up to 1.9× higher peak network throughput.
  - Up to 5.8× reduction in inter-chiplet packet latency.
  - Up to 17.4% improvement in system performance (10.0% on average).

1.3 Thesis Outline

The rest of this thesis consists of the three included papers and is organized as follows. Chapter 2 contains **Paper I**, "*BZSim: Fast, Large-scale Microarchitectural Simulation with Detailed Interconnect Modeling*". Chapter 3 contains **Paper II**, "*A Performance Analysis of Chiplet-Based Systems*". Chapter 4 contains **Paper III**, "*PRISM: Reducing Performance Overhead of Multi-chiplet Chip Interconnects*".

Bibliography

[1] Y. Feng and K. Ma, "Chiplet Actuary: A Quantitative Cost Model and Multi-Chiplet Architecture Exploration," in *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC)*, Jul 2022, pp. 121–126.

[2] A. Kannan, N. E. Jerger, and G. H. Loh, "Enabling Interposer-based Disintegration of Multi-core Processors," in *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2015, pp. 546–558.

[3] S. Naffziger, N. Beck, T. Burd, K. Lepak, G. H. Loh, M. Subramony, and S. White, "Pioneering Chiplet Technology and Design for the AMD EPYC™ and Ryzen™ Processor Families: Industrial Product," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, Jun 2021, pp. 57–70.

[4] Advanced Micro Devices, Inc., "AMD CDNA™ 3 Architecture," 2023. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/white-papers/amd-cdna-3-white-paper.pdf

[5] N. Nassif, A. O. Munch, C. L. Molnar, G. Pasdast, S. V. Lyer, Z. Yang *et al.*, "Sapphire Rapids: The Next-Generation Intel Xeon Scalable Processor," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2022, pp. 44–46.

[6] D. Stow, Y. Xie, T. Siddiqua, and G. H. Loh, "Cost-Effective Design of Scalable High-performance Systems Using Active and Passive Interposers," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2017, pp. 728–735.

[7] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, 2011.

[8] D. Sanchez and C. Kozyrakis, "ZSim: Fast and accurate microarchitectural simulation of thousand-core systems," in *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA)*, 2013, pp. 475–486.

[9] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation,"

in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2011.

[10] H. Shabani and X. Guo, "Cluscross: A new topology for silicon interposer-based network-on-chip," in *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, 2019.

[11] S. Bharadwaj, J. Yin, B. Beckmann, and T. Krishna, "Kite: A family of heterogeneous interposer topologies enabled via accurate interconnect modeling," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[12] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally, "A detailed and flexible cycle-accurate network-on-chip simulator," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013, pp. 86–96.

[13] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2009, pp. 33–42.

[14] I. Pérez, E. Vallejo, M. Moretó, and R. Beivide, "BST: A BookSim-based toolset to simulate NoCs with single- and multi-hop bypass," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2020, pp. 47–57.

[15] A. E. Kiasari, Z. Lu, and A. Jantsch, "An analytical latency model for networks-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 113–123, 2013.

[16] S. Y. Narayana, S. K. Mandal, R. Ayoub, M. M. Islam, M. Kishinevsky, and U. Y. Ogras, "Fast analysis using finite queuing model for multilayer NoCs," *IEEE Design & Test*, vol. 40, no. 6, pp. 112–124, 2023.

[17] A. Psarras, I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos, "ShortPath: A network-on-chip router with fine-grained pipeline bypassing," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 3136–3147, 2016.

[18] A. Ejaz and I. Sourdis, "FastTrackNoC: A NoC with FastTrack router datapaths," in *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 971–985.

[19] D. Stow, I. Akgun, and Y. Xie, "Investigation of Cost-Optimal Network-on-Chip for Passive and Active Interposer Systems," in *2019 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*, Jun 2019, pp. 1–8.

[20] N. E. Jerger, A. Kannan, Z. Li, and G. H. Loh, "NoC Architectures for Silicon Interposer Systems: Why pay for more wires when you can get them (from your interposer) for free?" in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2014, pp. 458–470.

[21] A. Coskun, F. Eris, A. Joshi, A. B. Kahng, Y. Ma, and V. Srinivas, "A Cross-Layer Methodology for Design and Optimization of Networks in 2.5D Systems," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2018, pp. 1–8.

[22] Y. Feng, D. Xiang, and K. Ma, "A Scalable Methodology for Designing Efficient Interconnection Network of Chiplets," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb 2023, pp. 1059–1071.

[23] ——, "Heterogeneous Die-to-Die Interfaces: Enabling More Flexible Chiplet Interconnection Systems," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct 2023, pp. 930–943.

[24] D. Lenoski, J. Laudon, K. Gharachorloo, W.-D. Weber, A. Gupta, J. Hennessy, M. Horowitz, and M. S. Lam, "The Stanford Dash Multiprocessor," *Computer*, vol. 25, no. 3, pp. 63–79, Apr 1992.

[25] J. Laudon and D. Lenoski, "The SGI Origin: A ccNUMA Highly Scalable Server," in *24th Annual International Symposium on Computer Architecture (ISCA)*, Jun 1997, pp. 241–251.

[26] F. Dahlgren and J. Torrellas, "Cache-Only Memory Architectures," *Computer*, vol. 32, no. 6, pp. 72–79, Jun 1999.

[27] Y. Feng, D. Xiang, and K. Ma, "A scalable methodology for designing efficient interconnection network of chiplets," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 1059–1071.

[28] M. Ebrahimi, A. Y. Weldezion, and M. Daneshtalab, "NoD: Network-on-die as a standalone NoC for heterogeneous many-core systems in 2.5D ICs," in *2017 19th International Symposium on Computer Architecture and Digital Systems (CADS)*, 2017, pp. 1–6.

[29] J. Yin, Z. Lin, O. Kayiran, M. Poremba, M. S. B. Altaf, N. Enright Jerger, and G. H. Loh, "Modular routing design for chiplet-based systems," in *Proceedings of the 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018, pp. 726–738.

[30] W. Cao, S. Jiang, and L. Huang, "Fast turn restriction algorithm to build deadlock-free modular chiplet integration systems," in *2022 IEEE 4th International Conference on Circuits and Systems (ICCS)*, 2022, pp. 27–32.

[31] P. Majumder, S. Kim, J. Huang, K. H. Yum, and E. J. Kim, "Remote control: A simple deadlock avoidance scheme for modular systems-on-chip," *IEEE Transactions on Computers*, vol. 70, no. 11, pp. 1928–1941, 2021.

[32] E. Taheri, S. Pasricha, and M. Nikdast, "DeFT: A deadlock-free and fault-tolerant routing algorithm for 2.5D chiplet networks," in *Design, Automation and Test in Europe Conference (DATE)*, 2022, pp. 1047–1052.

[33] ——, "ReD: A reliable and deadlock-free routing for 2.5-D chiplet-based interposer networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 12, pp. 4599–4612, 2024.

[34] D. Yu, A. Li, N. Jing, J. Jiang, W. Sheng, and Q. Wang, "VDA: A simple but efficient virtual-channel-based deadlock avoidance scheme for scalable chiplet networks," in *Proceedings of the Great Lakes Symposium on VLSI 2024*, 2024, pp. 357–363.

[35] J. Zhan, M. Poremba, Y. Xu, and Y. Xie, "Noδ: Leveraging delta compression for end-to-end memory access in NoC based multicores," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014, pp. 586–591.

[36] D. Deb, R. M.K., and J. Jose, "Flitzip: Effective packet compression for NoC in multiprocessor system-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 1, pp. 117–128, 2022.

[37] Y. Jin, K. H. Yum, and E. J. Kim, "Adaptive data compression for high-performance low-power on-chip networks," in *2008 41st IEEE/ACM International Symposium on Microarchitecture*, 2008, pp. 354–363.

[38] F. Jiang, C. Li, L. Chen, J. Liu, W. Zhang, and J. Xu, "SCNoCs: An adaptive heterogeneous multi-NoC with selective compression and power gating," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2024.

[39] Y. Wang, H. Li, Y. Han, and X. Li, "A low overhead in-network data compressor for the memory hierarchy of chip multiprocessors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 6, pp. 1265–1277, 2018.

I

Included Papers

Paper I

BZSim: Fast, Large-scale Microarchitectural Simulation with Detailed Interconnect Modeling

Panagiotis Strikos, Ahsen Ejaz, and Ioannis Sourdis

2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)

Chapter 2

BZSim: Fast, Large-scale Microarchitectural Simulation with Detailed Interconnect Modeling

Abstract

Modeling of on-chip interconnects in microarchitectural simulations is becoming more important. Chips continue to increase their number of cores, i.e., via 3D stacking and multi-chiplet integration, and their performance is gradually more affected by their network. However, existing simulation alternatives are either too slow for large systems or lack support for modeling interconnects, which is on its own a computationally intensive task. This work offers an open-source solution that integrates an accurate, widely used network simulator into an existing fast, parallel microarchitectural simulator enhanced with a new mechanism to make network modeling lightweight. It is based on the observation that a significant fraction of the network traffic has low contention and on the conjecture that the latency of such traffic can be calculated analytically with good accuracy. The proposed approach offers a mechanism to detect low contention traffic and analytically calculate its latency reducing the overheads of network simulation. In essence, it offers a knob for trading simulation accuracy for speed. This tradeoff is explored by demonstrating 2–4× faster simulations within 3–5% error in normalized IPC and 10–20% in average packet latency. Our simulation setup is an order of magnitude faster than an optimistic gem5 setup with point-to-point interconnects and 3× slower than a ZSim setup without network modeling. Our approach is further used to assess the impact of Network-on-Chip designs on system performance and shows that a 32-core system is on average 1.2× faster when using the current state of the art NoC instead of a baseline NoC.

2.1 Introduction

The exploration and evaluation of new computer architectures relies extensively on simulation. However, accurate microarchitectural simulation is computationally intensive and does not scale well with the increasing complexity of future architectures. More specifically, processor chips become larger as they integrate more cores, i.e., via 3D stacking of multiple chiplets, while their performance depends increasingly on their interconnects [10]. In turn, this puts more pressure on simulating fast, larger systems including accurate models of their interconnects.

In the past, there have been many attempts to achieve faster, large-scale simulations. FPGA-based solutions have improved simulation times [14, 24], but they are difficult to modify and offer limited monitoring and control. Statistical sampling [48] or other methods for selecting to simulate a small representative fraction of the workload [22] have been employed to shorten simulations, but are orthogonal to the actual speed of a simulator and increase error. Finally, parallel simulators have offered significant speedup by simulating simultaneously different parts of a system [11, 45].

There are several reasons why including interconnects in the simulation model of a system becomes more important. As single-chip systems fit a larger number of cores, their performance is increasingly affected and sometimes dominated by the performance of their on-chip interconnects [6, 10, 16, 33, 41]. Adding Network-on-Chip (NoC) models to the microarchitectural simulations allows the assessment of the network impact on the system's performance and promotes the exploration of new NoC designs. Furthermore, taking into

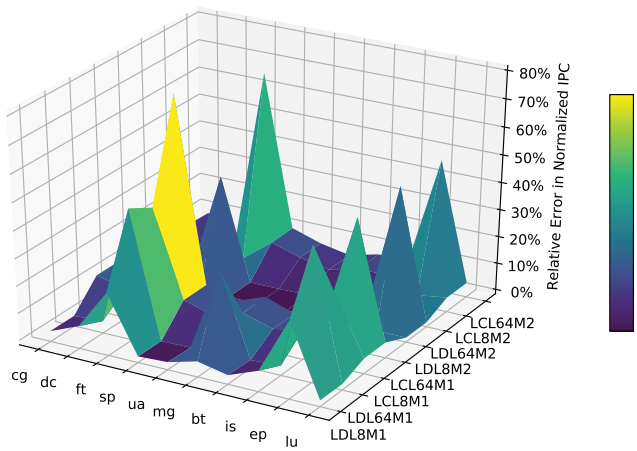


Figure 2.1: Relative error/difference between the performance results of a microarchitectural simulator (ZSim) with and without network modeling (BookSim). The normalized IPC results of the two simulation setups are compared for a 32-core system running NAS benchmarks. The explored design points select one of two alternative options for the following design parameters: (i) LLC distributed sliced in titles (LD) or centralized (LC), (ii) LLC capacity 8 MB (L8) or 64MB (L64), (iii) DDR4@2.6GHz (M1) or DDR3@666MHz (M2). In each case, IPC is normalized to LDL8M2 of the respective simulation setup.

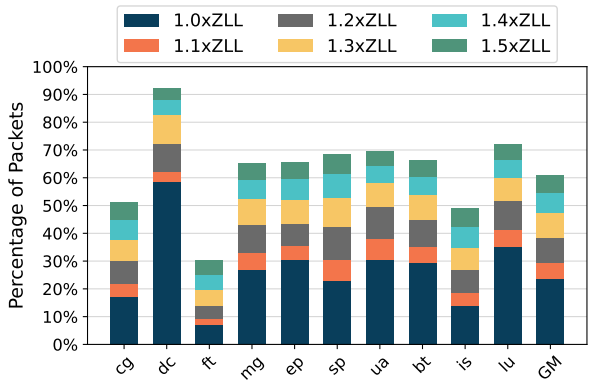


Figure 2.2: Percentage of packets with latency close to ZLL.

account NoC performance yields more accurate system performance results even when the focus of a study is not on the network. To exemplify, Figure 2.1 shows the relative error in performance results produced by a simulation with and without a NoC model exploring a 32-core system with different memory hierarchy configurations and running NAS Parallel Benchmarks (NPB) [5]. Although the two alternative simulation setups show in general similar trends, they produce different results. The speedup, i.e., normalized Instructions per Cycle (IPC), reported by each simulator setup with respect to a baseline design point that uses the same simulation setup differs by up to 77% and on average by 12%. At the same time, the absolute performance results, i.e., actual IPC, are up to 100% and on average 17% different. It is worth noting that the difference between the two simulation setups reduces to half when evaluating an 8-core system. The above shows the importance of accurate network modeling and its impact on system performance, especially for larger systems.

Current simulation alternatives are unfortunately either too slow and hence impractical for larger and longer simulations, e.g., gem5 [9], or do not model the network part of a system, e.g., ZSim [45] and Sniper [11]. More specifically, gem5 can use network simulators such as Garnet 2.0 [1] and BookSim 2.0 [23], henceforth denoted as Garnet and BookSim respectively, or Noxim [12]. However, even without a network model, gem5 is able to simulate only a few hundred kilo instructions per second, which translates to several hours for simulating a single core for a second, and days for simulating a few tens of cores for a second. On the other hand, ZSim is parallel and about 2 orders of magnitude faster than gem5 [45] but does not model on-chip interconnects. To make matters worse, simulating a network is on its own a computationally intensive task as it needs to accurately model contention across the different network resources. Despite recent attempts to skip intervals of network simulation when empty, integrating Garnet or BookSim into gem5 substantially decreases the simulation speed [42].

This work delivers a new, open-source solution for fast, large-scale microarchitectural simulation, which models accurately the network of the simulated system. BookSim, a popular network simulator [23], is integrated into ZSim, one of the fastest, parallel microarchitectural simulators [45] to produce BZSim. BZSim introduces a new method to make the network part of the simulation lightweight and proportional to the actual contention of the network. It is based

on the insight that a significant fraction of the network traffic on a multicore system finds little, if any contention and on the conjecture that the latency of such low contention traffic can be estimated analytically with little loss of accuracy. As shown in Figure 2.2, on a 32-core system like the one used for the evaluation and described in Table 2.1, when running the NAS benchmarks, on average 24% of the packets have latency equal to their zero-load latency (ZLL). This percentage increases to 39%, and 61% for latency lower than 1.2×ZLL, and 1.5×ZLL, respectively. Our suggestion is to identify such low contention traffic and calculate its latency analytically, rather than injecting it into the network simulation model. Thereby, the network part of the simulation can be offloaded and become faster. To make the network simulation even more proportional to the traffic necessary to simulate, each part of the network model is monitored separately and their evaluation is skipped when empty. In effect, the total traffic injected is significantly reduced improving simulation speed while maintaining the benefits of network modeling.

Concisely, the contributions of this paper are the following:

- A widely used network simulator, BookSim, is integrated into one of the fastest microarchitectural simulators, ZSim, to produce BZSim, a fast, large-scale microarchitectural simulator with detailed interconnect modeling.
- BZSim is enhanced with a novel mechanism that identifies low contention traffic and skips its simulation using an analytically calculated latency rather than producing one via simulation.
- A tradeoff between simulation speed and accuracy is analyzed and explored under different simulation parameters.
- Current state of the art high-speed NoCs are modeled and their impact on system performance is measured.

Although BZSim does not solve the scaling problem of simulation speed to larger systems, it does offer faster system simulation with interconnect modeling.

The remainder of this paper is organized as follows: Section 2.2 discusses the various simulator alternatives including the ones used in this work and presents the current state of the art designs of high performance NoCs. Section 2.3 describes the proposed simulator focusing on the techniques introduced for reducing network modeling overheads. Section 2.4 presents and validates the network models for the fastest state of the art NoC routers. Then, Sections 2.5 and 2.6 describe the experimental setup, evaluate the tradeoffs between simulation speed and accuracy offered by this work, and assess the impact of high performance NoCs on system performance. Finally, Section 2.7 summarizes our conclusions.

2.2 Background

An overview of microarchitectural and network simulators is provided focusing on ZSim [45] and BookSim [23], which are the basis of the proposed BZSim. Furthermore, current state of the art NoCs are discussed, describing new features employed by the fastest existing NoCs modeled in this work.

2.2.1 ZSim and other microarchitectural simulators

There is a plethora of microarchitectural and system simulators, most of them being event-driven with support to connect to DRAM models, such as DRAMSim2 [44]. One of the most widely used ones, gem5 [9], is a full system simulator capable of modeling OS features. Gem5 uses an emulation engine to support Instruction Set Architectures (ISA) which are different than the host. Although it offers a detailed model of the uncore part of a system including its interconnects, it is sequential, which makes the simulation relatively slow. Other simulators such as ZSim [45] and Sniper [11] are more focused on the microarchitecture of a system and so do not support full system features, although ZSim is capable of supporting multiprocess workloads along with some application management. Sniper and ZSim are instrumentation-based simulators that use Pin [34] to perform dynamic binary translation and eliminate the need for functional instruction modeling. Unfortunately, none of them includes a network model in their uncore part. Although both are parallel, ZSim is substantially faster mainly due to its *bound-weave* two-phase parallelization algorithm described next. This feature of ZSim, is the primary reason for selecting it as the basis of this work.

ZSim simulations progress by repeating two phases, working on time intervals of a fixed number of cycles, e.g., 1000 cycles. During ZSim's (first) *bound* phase, individual threads are assigned to each simulated core, running in parallel with other threads to record and interlink events for different memory operations. Network latency is estimated using zero-load latency (ZLL) without taking into account contention. Following each operation, the generated events become part of an event chain that is later used in the (second) *weave* ZSim phase. During the weave phase, a parallel, event-driven simulation is performed, using the trace created in the bound phase, which simulates the events and produces an updated more accurate latency of each one. In case an updated latency diverges from the initial estimation that took place during the bound phase, subsequent events are updated and delayed accordingly. This process is repeated for each interval.

2.2.2 BookSim and other network simulators

BookSim [23] and Garnet [1], are two widely used network simulators. Garnet is integrated with gem5 and uses a flexible pipeline approach for modeling network routers, which is faster but more abstract than BookSim, making it difficult to model detailed router features [42]. This led us to choose BookSim for our work. BookSim is time-driven and cycle-accurate. It allows assessing the impact of different design choices on network performance metrics such as latency, throughput, and scalability. Moreover, BookSim is flexible and supports a wide range of topologies, allocators, routing algorithms, etc. Besides a typical 3-stage NoC router there is a wide variety of NoCs modeled in BookSim, many of which are described in [23]. The most recent features modeled in BookSim include single and multi-hop bypassing [42]. As explained next, this work adds models of the latest high performance NoC routers features, such as Dual Data Rate (DDR) datapaths, Switch Traversal bypassing, and fine grain bypassing of individual control stages, e.g. switch and virtual channel allocation.

Analytical methods for calculating network latency can be an alternative

to network simulation. However, such methods often do not support saturated networks [25]. Although a recent approach, concurrent to this work, addresses this shortcoming, modeling analytically some network features, e.g., bypassing mechanisms or arbitration schemes, can be complex, while performance analysis of congested networks introduces considerable error [36].

2.2.3 Current state-of-the-art high-performance NoCs

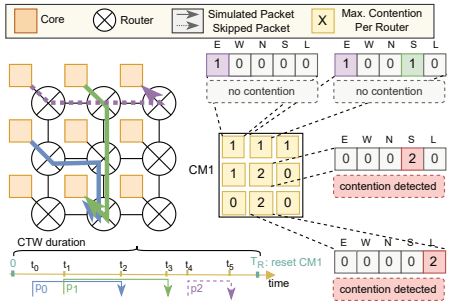
In the past, the performance of Networks-on-Chip (NoC) has improved substantially. New network topologies have improved network throughput [7, 28, 39] and latency [21]. New routing algorithms have reduced packet latency [29], and more efficient allocation algorithms have increased throughput [8, 43]. Finally, one of the primary factors that affect NoCs performance is the router design, which is discussed next.

A flit in a baseline NoC router goes through three pipeline stages [15], which are: i) an allocation stage where virtual channel (VC) and switch allocators allocate VCs and desired input and output ports of a crossbar switch to the flit, ii) a switch traversal (ST) stage for the flit to propagate the crossbar switch and iii) a link traversal (LT) stage for the flit to propagate from its current router to a neighboring router. The allocation logic in the first stage is usually in the critical path and it limits NoC clock frequency and throughput [17, 26, 35, 37, 43].

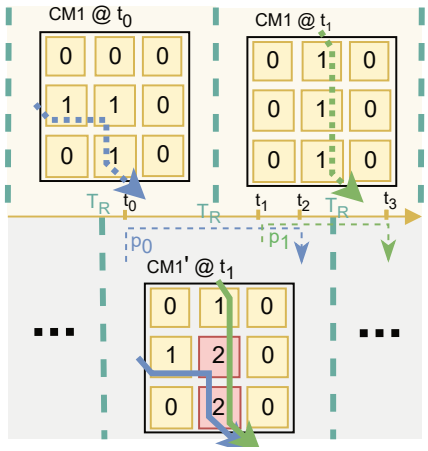
Recent NoC designs addressed this bottleneck by allowing the datapath of a router to be traversed twice in a cycle using both clock edges [17–20]. The control of such a Dual Data Rate (DDR) router uses an entire, longer cycle to make decisions, so it is no longer in the critical path. In turn, this increases DDR NoC's throughput to switching rates defined solely by the datapath delay, i.e., ST or LT [18, 19].

The latency for a single network hop, i.e., crossing a tile, can be in theory as short as the register-to-register delay of a link that traverses the tile dimension [27]. Some existing networks such as single-cycle multi-hop routers [13, 31], routers with express links [21, 32], dimension-sliced routers [27] or routerless rings [4] offer low latency, but sacrifice substantially their throughput. On the other hand, as explained next, new router designs have achieved to offer a minimum hop latency close to the aforementioned theoretical, while also maintaining high throughput [19]. This is achieved by a combination of techniques. The dependency between route computation and allocation can be removed and allow them to be performed in parallel saving a cycle if routing is precomputed at the upstream router [15]. Speculative switch allocation allows virtual channel allocation (VA) and switch allocation (SA) to be performed in parallel saving one more cycle [38]. Lookahead control signaling allows the control information of a flit to be generated early and travel forward one cycle ahead of the flit on a separate narrow link to enable downstream allocation a cycle earlier [16, 30]. Detecting the absence of contention between incoming flits allows to bypass allocation stage(s) [16, 40]. Finally, switch traversal can also be bypassed for non-turning hops allowing incoming flits to proceed directly to the link traversal [19].

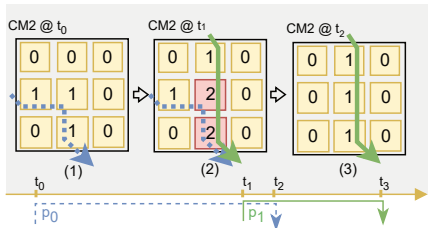
FastTrackNoC introduced ST bypassing and combined it with several of the above techniques to achieve a latency that is at best equal to the LT latency plus the delay of a 2:1 multiplexor [19]. Thereby, it demonstrated lower



(a) Contention map (CM1) is updated in a CTW at times t_0, t_1, and t_4 when packets p_0, p_1 and p_2 are created, respectively. After p_1 is created, contention in the output of the last router is detected, requiring p_0 and p_1 to be simulated. When p_2 is created at t_4, no contention is detected and the packet skips the simulation. CM will be reset at T_R, at the end of each CTW.



(b) Contention between packets p_0 and p_1 cannot be detected from CM1 alone, since CM1 will be reset between the p_0 and p_1's creation. To detect this contention between p_0 and p_1, CM1' is introduced to overlap between two consecutive CM1 instances.



(c) Phase-2 contention map CM2. (1) When p_0 arrives, given that CM1 does not indicate any contention, it skips the simulation but still updates the CM2. (2) When p_1 arrives, it is simulated because contention is indicated by CM2. (3) ZLL cycles after p_0 was injected, CM2 is updated to reflect p_0 has reached its destination. After p_1 finishes simulation, CM2 will be updated again.

Figure 2.3: BZSim's contention detection mechanism.

average packet latency than the previous best ShortPath router [40], which uses allocation bypassing. FastTrackNoC is further able to support DDR datapaths combining its low latency with maximum network throughput.

Although single and multi-hop bypassing, lookahead routing, and other existing techniques have been already modeled in BookSim [42], bypassing of individual control stages (employed by ShortPath), ST bypassing, and DDR router datapaths are first modeled in this work.

2.3 BZSim: exposing the tradeoff between speed and accuracy of network modeling

This work offers a knob in microarchitectural simulation for trading network modeling accuracy for simulation speed. One of the most commonly used network simulators, BookSim [23], is integrated, as described at the end of this Section, into one of the fastest microarchitectural simulators, ZSim [45], to produce BZSim. BZSim is based on the conjecture that the latency of lower contention traffic can be analytically calculated relatively accurately. It classifies traffic based on its estimated level of contention using a user-defined threshold. Then, the latency of low contention traffic is analytically calculated rather than simulated. In turn, this reduces the amount of simulated traffic and hence the workload of network modeling. In addition, BZSim monitors the different parts of its network model separately and skips their evaluation when unused. The combination of the two above mechanisms, described next in detail, exposes a useful tradeoff between simulation speed and accuracy.

2.3.1 Detecting and skipping simulation of low contention traffic

BZSim exploits the two-phase simulation structure of ZSim to detect and avoid simulation of low contention traffic. A simple data structure, called contention map (CM), is created and used in the first ZSim simulation phase (CM1) to capture contention information about network traffic injected during fixed simulated time windows, called contention time windows (CTW). The duration of a CTW is set to be comparable to the maximum zero-load latency (ZLL) of a packet in the simulated network. Increasing or decreasing CTW duration causes over- or under-estimation of network contention leading to higher or lower accuracy of the simulated network, respectively.

CM1 keeps track of the number of packets that pass through each output port of the NoC routers during a CTW, i.e., CM1 is reset every CTW cycles. An output port is considered congested if CM1 indicates more packets than a user-defined contention threshold (CT) pass through the port within CTW.

Let us consider an example where three packets p_0, p_1 and p_2 enter the network during the same CTW, at times t_0, t_1 and t_4 respectively, as shown in Figure 2.3a. Here we set the CT to one. The CM1 is updated at each packet insertion within this CTW indicating the number of packets passing through the output ports of each NoC router. For simplification in Figure 2.3a, only the maximum contention value of a router port in the CM1 is shown. CM1 then indicates possible contention between p_0 and p_1 in an output port shared

by both packets within the same CTW. These contending packets are then sent for network simulation. On the contrary, for p_2, which does not encounter contention, latency is analytically calculated.

A special case occurs when contending packets, generated close to each other in time, fall in different consecutive CTWs, i.e., CM1 is reset before all contenting packets are injected, as exemplified in the top half of Figure 2.3b. In such cases, the opportunity to detect potential contention can be missed leading to inaccuracies in packet latency estimation. BZSim adds a second overlapping CM, called CM1', to detect such contention across CTW boundaries, which is reset with an offset of CTW/2 cycles compared to CM1, as shown at the bottom of Figure 2.3b.

During ZSim's bound phase, a preliminary minimum timing is given to the generated events such as cache accesses, network packet deliveries, and DRAM accesses. For example, the ZLL of a packet is considered assuming no contention. The timing of all events is subsequently updated during the weave phase of the simulation. As a consequence, packets that fall within the same CTW and thus the same CM1/CM1' instance in the bound phase of the simulation may drift apart or come closer during the weave phase. This may introduce inaccuracies in the detection of network contention using CM1 and CM1' in the weave phase. BZSim creates an additional CM during the weave phase, called *CM2*, to detect contention based on the updated latency of packets. However, instead of maintaining a CM2 per CTW, only a single CM2 is used for the entire simulation and it is updated upon packet injection and ejection.

The contention maps for the first and second phases together indicate contention between injected packets. More precisely, upon packet injection, an updated CM value P, corresponding to an output port of a router, indicates that the packet competes for that output port with $P-1$ packets, as shown in the examples of Figures 2.3a and 2.3b. If P is greater than CT in any contention map of phases 1 or 2 (CM1, CM1', or CM2), then, the contending packets are marked to be inserted for network simulation. Otherwise, it is skipped and its latency is provided by the following formula:

$$P_{latency} = ZLL + (P_{max} - 1) * Hop_{delay} \tag{2.1}$$

where ZLL is the zero-load latency of the packet, P_{max} is the maximum value P updated by the inserted packet in any contention map (corresponding to an output port it passed through), and Hop_{delay} is the time needed by a packet to pass through a router[1]. In essence, this formula delivers the ZLL of a packet when all updated CM values are equal to '1', indicating no contention. Otherwise, the blocking delay of the packet's most congested hop is estimated and added to ZLL.

There is a reason why the contention maps are used in both the bound and weave phases of simulation to detect low contention traffic. As mentioned earlier, the first (bound) phase may not be accurate since the latency of events is preliminary and may get updated in the second (weave) phase. Hence CM1 and CM1' in the first phase may have both false positives and false negatives,

[1]For variable size packets, besides P, the contention map needs to maintain the packet lengths of contenting packets.

the former ones hurting simulation speed and the latter ones accuracy. CM2, in the second phase, alone is also not sufficient because it would always skip the first T packets before detecting contention, introducing significant inaccuracies. For instance, in the example depicted in Figure 2.3c, although p_1 would be sent for simulation, p_0 would not, because at the time of p_0 injection the corresponding values in the CM remained within the threshold.

In the extreme case, BZSim can be configured to completely ignore the threshold, hence never invoking network simulation. Then, the latency for all packets is analytically calculated as described above. The effects of this extreme configuration in simulation accuracy and speed as well as the effect of various threshold values and window sizes are explored in our Evaluation Section 2.6.

2.3.2 Skipping the evaluation of empty network parts

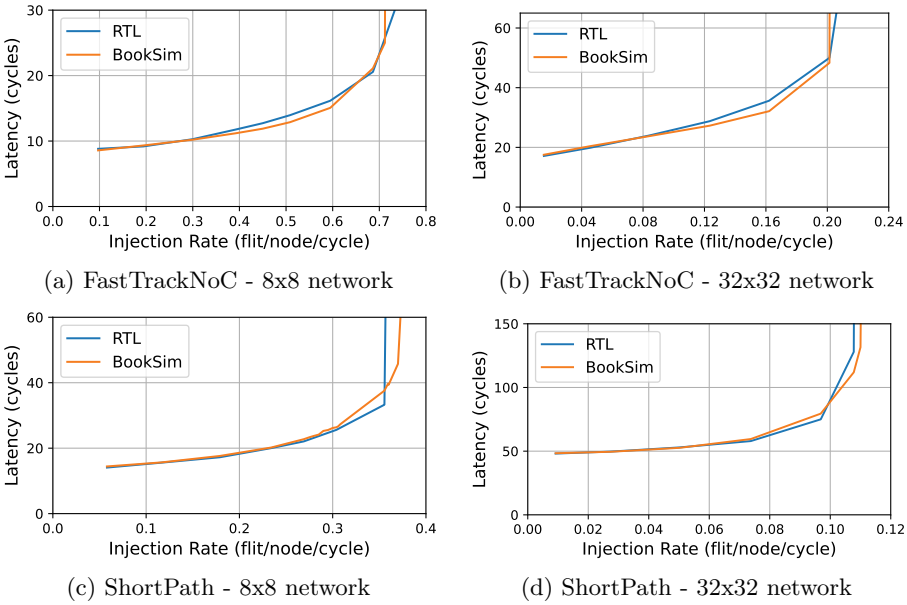
Individual network parts are monitored separately in BookSim and their evaluation is skipped when they do not handle any traffic. This makes the processing cost of network modeling more proportional to the simulated traffic, increases simulation speed and helps to capitalize on the benefits of the above proposed mechanism which reduces the amount of simulated traffic.

2.3.3 BookSim-ZSim integration

The integration of BookSim into ZSim required a number of steps. BookSim was compiled as a separate library using a wrapper by Perez et al. [42], rather than as part of ZSim, which allows easier integration and faster compiling. The first phase of ZSim was extended to create network events for the interaction between cores, caches, and main memory controllers. These network events were given in this first phase a preliminary latency equal to the ZLL of that network event, which was calculated based on the source and destination of the created message, as well as the routing algorithm of the network. The network events were then carried in the second phase of ZSim and their latency was adjusted as described above, either via BookSim simulation or by analytically calculating it. The network simulation runs cycle by cycle only when there are outstanding packets, otherwise it is not engaged at all.

2.4 Modeling of high performance NoCs

A number of NoC router features, missing from previous BookSim works, were modeled in this work. This allows us to measure the impact of current state of the art NoCs on the performance of a system. As discussed in Section 2.2.3, these features are (i) bypassing of individual router control stages, used extensively in ShortPath as well as in FastTrackNoC [19, 40], (ii) Switch Traversal bypassing, used in FastTrackNoC, and (iii) DDR router datapaths, used in FastTrackNoC. The models of ShortPath and FastTrackNoC in BookSim employing the above features to offer both low latency and high network throughput are discussed next and the comparison of these models against their RTL counterparts is presented.



(a) FastTrackNoC - 8x8 network

(b) FastTrackNoC - 32x32 network

(c) ShortPath - 8x8 network

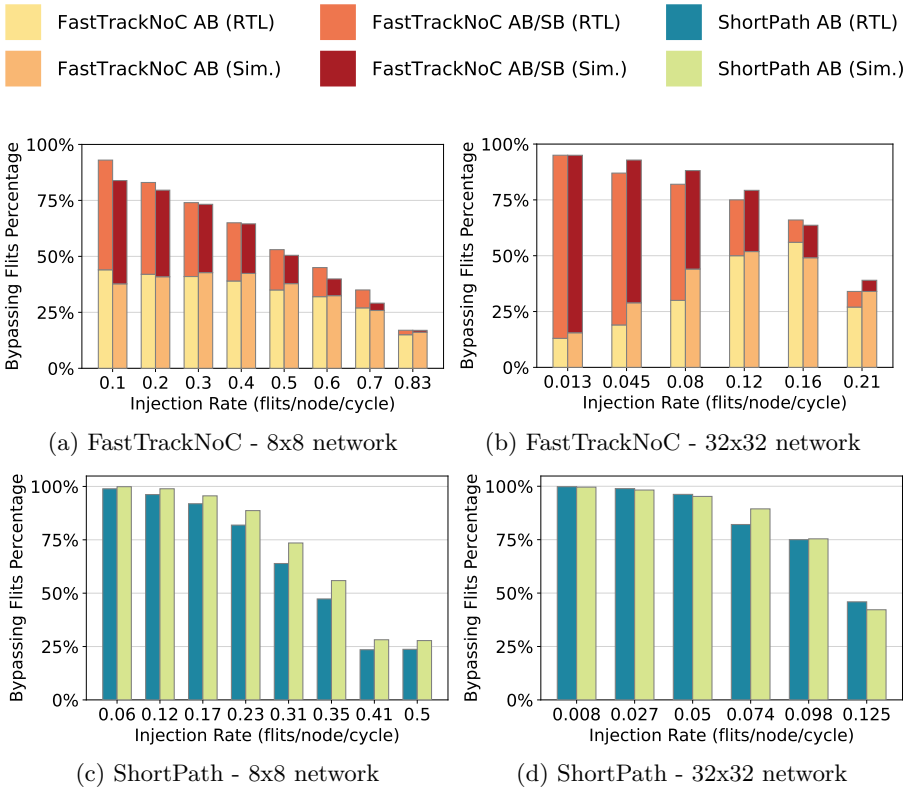
(d) ShortPath - 32x32 network

Figure 2.4: FastTrackNoC and ShortPath validation of Booksim models vs. RTL.

Previous BookSim models did not include a two-stage switch allocator (SA). So, for ShortPath, a two-stage, Separable Input-First Switch Allocator, and a SA Request Queue (SARQ) were modeled and integrated on BookSim's 4-stage router. In addition, the second SA stage (SA2) was merged with ST to produce a ShortPath model with the following stages: VA, SA1, SA2/ST, LT. Bypass checks were then added for VA and SA1 along with the option to skip the respective cycle(s) by moving bypassing flits to the proper queues. Finally, BookSim's router model was further adjusted to provide access to all flits in an input queue and to regulate insertions of incoming flits to the allocation queues.

The FastTrackNoC model was based on BookSim's 3-stage router with a speculative switch allocator (VA/SA, ST, LT). Its DDR datapath was modeled using a separate BookSim cycle for each clock edge. Then, flits could traverse datapath stages, i.e., ST, LT, every cycle, and control (allocation) would be performed in every odd cycle. Similar to the ShortPath model, bypassing logic was added to skip switch and VC allocation when possible. In addition, the option for bypassing the switch traversal stage was added for flits that satisfied the criteria, briefly, that is when they traverse a straight hop and arrive at input VC0 when it is empty.

The BookSim models of ShortPath and FastTrackNoC were validated against their RTL implementations for 2D mesh networks of size 8×8 and 32×32, considering 4 VCs routers and using uniform random traffic of 1- and 5-flit packets, split evenly. Experiments were performed by first simulating the networks for a warm-up phase of 1000 clock cycles and then for a measurement phase of 10000 clock cycles. Average packet latency was measured for packets



(a) FastTrackNoC - 8x8 network

(b) FastTrackNoC - 32x32 network

(c) ShortPath - 8x8 network

(d) ShortPath - 32x32 network

Figure 2.5: Validation of FastTrackNoC and ShortPath RTL vs. simulation models using BookSim (Sim.) on average latency, and percentage of flits performing Allocation Bypassing (AB), and Switch Traversal Bypassing (SB).

delivered during the measurement phase. Figure 2.4 shows their performance comparison measured in cycles rather than absolute time. It is worth noting that FastTrack and ShortPath cycles are different as explained in the next Section 2.5. In general, both the average packet latency and network throughput of the models are close to their RTL implementation for both networks. At low injection rates, FastTrackNoC latency is less than 3% off, while in medium injection rates, the error increases to 5-9% mainly due to a different arbitration logic and due to inaccuracies in the DDR modeling, because some router control functions are activated every cycle, rather than every other, which was in some cases too complex to model. Moreover, the FastTrackNoC model suffers less than 1% error in maximum throughput due to differences in the arbiter. On the other hand, the Shortpath model has up to 2% and 5% error in latency in low and medium-high injection rates, respectively, while its error in throughput is up to 2%.

The low error of the performance of the network models is confirmed by the low error in the percentage of activated bypassing cases compared to the RTL version of the networks when handling similar traffic at the same injection rate. The comparison in terms of bypassing percentages between our network models and their RTL counterparts is illustrated in Figure 2.5. On average, the error

of detecting bypass cases is 2-3% for FastTrackNoC and 1-7% for ShortPath. Detecting the specific type of bypassing for FastTrackNoC is quite accurate at low injection rates where the error is 6-16% but becomes less accurate for higher injection rates, where bypasses are less common, and the amount of switch traversal bypassing is overestimated by our model due to the above mentioned inaccuracies in DDR modeling. For ShortPath, the model is quite precise and within 1% error for a 32×32 network, but the number of bypasses is slightly underestimated in 8×8.

2.5 Experimental setup

The system configurations, simulation setups, and benchmarks used in our experiments are discussed next.

Most of our experiments model a 32-core system organized in tiles as illustrated in Figure 2.6 and configured with the parameters shown in Table 2.1. In a nutshell, out of order (OoO) cores are simulated with private L1 and L2 caches and a shared last level cache (LLC) sliced and distributed evenly across the tiles. The system uses the MESI cache coherence protocol and is connected to a DDR4 DRAM through a single memory controller. The only exception to the above system configuration is some of the design points explored in Section 2.1, which use a centralized, monolithic LLC, and/or smaller LLC, and/or slower DDR3 DRAM. In all experiments, a 128-bit NoC datapath is used, and 64 B cache lines, which set the packet size to 5-flit for data and one flit for control. With the exception of Section 2.6.3 a 3-stage baseline NoC design is used as described in Table 2.1. The operating frequency of the different NoC designs was retrieved via place & route implementations in 28 nm CMOS FDSOI 1.10 V standard cell technology. Finally, Cacti was used to determine the access times for the caches [47] and DRAMSim2 for modeling the memory system [44].

For our experiments, multi-threaded workloads OpenMP version of the

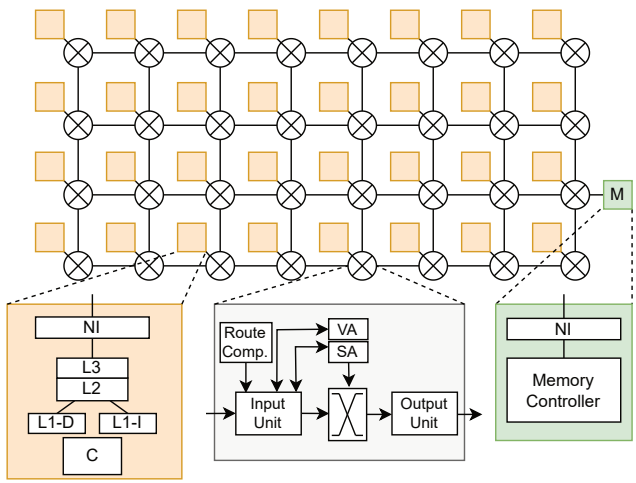


Figure 2.6: Multicore system organization used in most of our experiments.

Table 2.1: System and Network Configuration

System Configuration			
Cores	32 Cores, out of order, 3.2 GHz		
L1 Cache	Private, 32 kB, 4-way, 1 cycle access latency		
L2 Cache	Private, 128 kB, 8-way, 9 cycles access latency		
L3 Cache	Distrib. 16 MB, 16-way, 14 cycles access latency		
Memory	DDR4-3200 - 16 GB		
Network Parameters			
Design	Baseline [15]	ShortPath [40]	FastTrackNoC [19]
Frequency	2 GHz	2.5 GHz	1.6 GHz
Router Arch.	3-stages VA/SA, ST, LT	4-stages VA, SA1, SA2/ST, LT	2-stages ST, LT (VA/SA parallel to upstr. LT)
Topology	4×8 Mesh		
Flow Ctrl. VCs	4 VCs per port, credit-based flow control		
Other	Link: 128 bit, Buffer Size: 5-flit buffers, Routing: XY		

NAS Parallel Benchmarks [5, 46] class S were used and ran until completion executing up to 1 billion and on average 0.4 billion instructions per benchmark. There is the following exception to using the above workload. The comparison against gem5 required switching to the class *A* version of NAS benchmarks and exclude *dc* and *ua*, which were not supported by the gem5 setup; in this case, benchmarks ran for 1 billion instructions. In both the standalone ZSim and the BZSim configurations, out of order (OoO) core simulations with contention were selected, meaning both the bound and weave phases were activated. Gem5 setup was as optimistic as possible for gem5, using a multi-core system model of 32 TimingSimple (in-order) cores with X86 ISA and a Ruby-based cache hierarchy (as presented in Table 2.1) connected using the simple, point-to-point interconnection network, which does not model in network contention. System memory, modeled using DRAMSim2, was set to DDR4. The clock frequency of the modeled system was set to 3.2 GHz. A fast version of the gem5 simulator was compiled to reduce its simulation execution time by removing run-time debug support. Finally, simulation speed was measured on an Intel i9-12900K CPU machine with 64 GB DDR4-4000 memory running Ubuntu Linux 22.04 x86-64 with gcc 11.4. For the BZSim simulations, Pin 2.14 is also employed. Each benchmark was run separately to ensure precise calculation of its execution time.

2.6 Evaluation

First, the tradeoff between simulation speed and accuracy is explored for different BZSim windows and thresholds. Then, BZSim simulation speed is compared to other simulation alternatives, namely gem5 and ZSim. Finally, the impact of different NoC designs is measured using BZSim.

2.6.1 Exploring the simulation speed vs. accuracy tradeoff

Different points of the tradeoff between simulation speed and accuracy, exposed by BZSim, are explored on a 32-core system using as parameters the window size of CM1 and the threshold value used for deciding whether the latency of a packet will be produced via simulation or analytical calculation. Window sizes equal to the maximum ZLL as well as half and a quarter of that are explored. Threshold values of 1 to 5 are examined in addition to the option of not having any threshold, i.e., the latency of all traffic is analytically calculated.

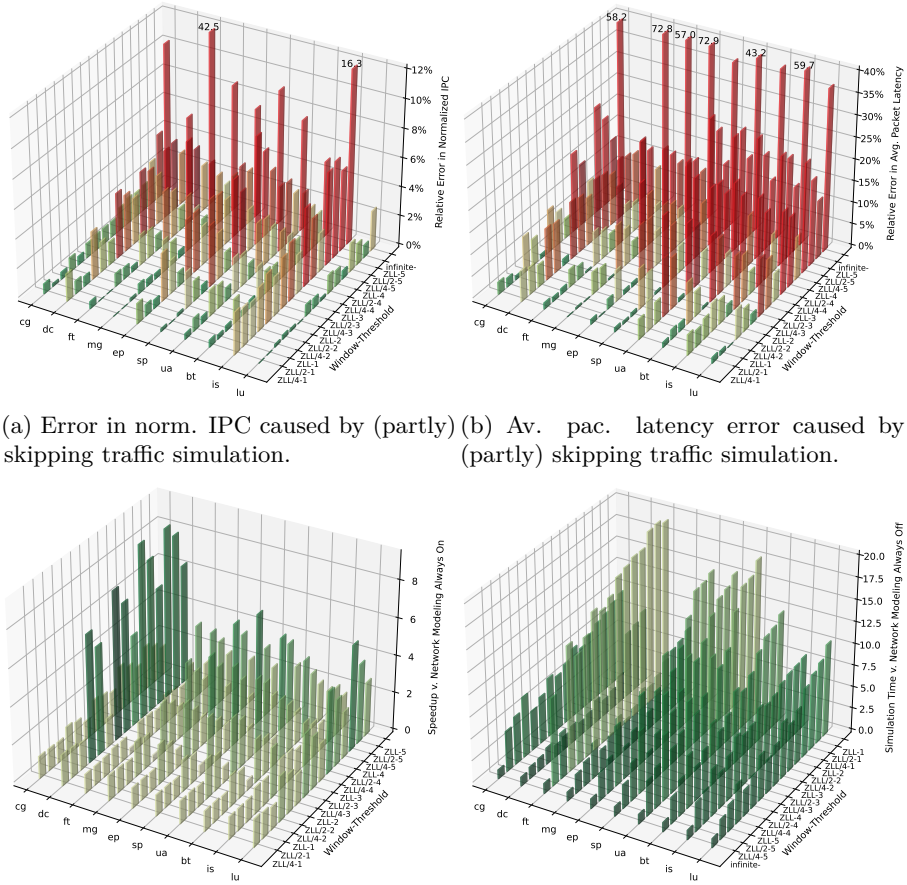
This analysis measures the relative error of skipping simulation for parts of the traffic on (i) the normalized IPC and (ii) the average packet latency compared to a baseline setup where all network traffic is always injected into network simulation. In addition, the improvement in simulation speed compared to the same baseline is evaluated as well as the increase in simulation time compared to not simulating the network at all. It is worth noting that the improvement in simulation time is proportional to the amount of traffic that skips network simulation, which is depicted in Figure 2.7.

Figure 2.8a depicts the error in normalized IPC compared to always simulating network traffic. This error is maximized when the threshold is disabled and traffic latency is always analytically calculated. In that case (threshold $= \infty$), the error in normalized IPC is between 3% and 42.5% and on average 12%. It can be observed that smaller threshold values and larger window sizes offer lower IPC error. For example, when any indication of contention sends a packet to simulation, i.e., threshold equal to 1, the average IPC error is less than 1%, and in that case on average 20% of the traffic skips simulation. The error is gradually increasing to 4% on average for a threshold of 5 where 65-85% of the traffic skips simulation.

A similar trend is observed in the error of the average packet latency, shown in Figure 2.8b. When the threshold is disabled, the error in average packet latency is between 14% and 73% and on average 49%. A threshold equal to 1 limits the average error in packet latency to lower than 3%, while thresholds 2, 3, 4, and 5 introduce an error of 5%, 10%, 14%, and 19% on average. An interesting observation is that the largest window size, equal to the maximum

window	threshold	cg	dc	ft	mg	ep	sp	ua	bt	is	lu	
-	infinite	100	100	100	100	100	100	100	100	100	100	100.0
ZLL	1	12	41	2	14	10	6	10	12	6	17	13.0
	2	24	83	8	29	39	22	34	30	14	36	31.9
	3	31	88	14	43	54	41	52	46	25	51	44.4
	4	44	93	23	55	68	56	64	58	37	62	55.9
	5	54	94	33	64	73	66	71	66	50	70	64.1
ZLL/2	1	20	50	3	19	16	12	19	20	9	26	19.3
	2	36	91	11	41	58	39	54	43	24	50	44.7
	3	42	95	22	63	73	63	67	62	39	68	59.3
	4	59	97	36	75	83	76	77	74	57	77	71.1
	5	70	98	49	81	86	84	85	81	73	83	78.9
ZLL/4	1	25	53	4	22	19	14	23	23	10	30	22.4
	2	43	94	14	49	69	49	59	52	28	57	51.5
	3	48	97	26	72	80	73	73	69	46	75	65.9
	4	68	98	42	84	87	86	82	81	69	84	78.1
	5	77	99	57	88	89	93	88	88	78	90	84.7

Figure 2.7: Percentage of packets that skip network simulation for different BZSim window sizes and thresholds.



(a) Error in norm. IPC caused by (partly) skipping traffic simulation.

(b) Av. pac. latency error caused by (partly) skipping traffic simulation.

(c) Speedup of simulation time achieved by (partly) skipping traffic simulation.

(d) Simulation Speed vs. Simulations without network modeling (ZSim). Note the points on the “Window - Threshold" axis are in reverse order compared to the other plots.

Figure 2.8: Exploring the impact of BZSim’s window size and threshold parameters to the (i) normalized IPC error, (ii) average packet latency error, (iii) speedup vs. simulating all network traffic, (iv) BZSim Simulation speed vs. not simulating any network traffic i.e., vs. original ZSim.

ZLL, offers an IPC error of only 1% and packet latency error of 4% even for threshold=2, while it skips simulation of roughly a third of the network traffic.

The maximum speedup achieved over a BZSim configuration that simulates all traffic is 2.5× to 9.3× and on average 4.5×. In these cases, BZSim, which selectively simulates about 20% of the network traffic and analytically calculates the latency of the rest, offers a simulation speed of 2-19 Million Instructions per Second (MIPS), which is 1.3× to 4.5× and on average 3× slower than a simulator without network modeling, i.e., the original ZSim. However, these fastest BZSim configurations may have an excessive error on normalized IPC and average packet latency. Trying to put a limit to the maximum error in

IPC and packet latency one can still achieve significant gains in simulation speed. For example, when limiting the maximum IPC and average packet latency errors to 2.5% and 5%, respectively, BZSim improves simulation speed compared to simulating all traffic by 1.4×, and is 9.9× slower than the original ZSim offering 2.4 MIPS on average. Relaxing the IPC and packet latency error limit to 3% and 10% offers an average simulation speed of 3.1 MIPS and a speedup of 1.8×, which is 7.7× slower than ZSim. Maintaining the maximum IPC error to 3% and increasing the packet latency error limit to 15% offers a speed of 3.9 MIPS, a speedup of 2.3× compared to always simulating network traffic, and is 6× slower than ZSim. Finally, relaxing the maximum acceptable errors to 5% for IPC and 20% for packet latency delivers an average simulation speed of 5.6 MIPS and speedup of 3.7× compared to a setup that simulates all traffic, while it is only 3.7× slower than ZSim, which in comparison suffers an IPC error of 12% on average (up to 42.5%) and does not model a network.

Although the accuracy of BZSim is not directly evaluated against real-world machines or other simulator alternatives, e.g., gem5, it can be stated that it depends primarily on the accuracy of its baseline system simulator (ZSim). The BookSim NoC models, on the other hand, offer near RTL accuracy. ZSim has been verified against real-world machines demonstrating good accuracy [45] and also compared to the accuracy of other simulators, such as gem5 and Sniper [2, 3] showing notable differences in absolute values, but similar trends in normalized results. BZSim performance results are on average 10% lower than the original ZSim, since adding NoC latency increases the memory access time.

2.6.2 Simulation Speed Comparison

Next, our BZSim simulation setup is compared against gem5 and ZSim. For that comparison, the experimental setup was adjusted to match the one available from gem5 when running the NAS Parallel Benchmarks (NPB). Given that the NAS class S benchmarks, used in all other experiments of this work, are not available in gem5, class A benchmarks running for 1 billion instructions were used. Another difference is that the interconnect of this particular gem5 system setup is simplified and hence quite optimistic for gem5 as it only models point-to-point connections between system components, rather than using a network modeled by Garnet or BookSim. Therefore, it is worth noting that according to Perez et al., this gem5 setup is roughly 1.5× and 5× faster than a gem5 setup that uses Garnet or BookSim, respectively, for modeling a network [42]. For these experiments, BZSim utilizes the same BookSim baseline NoC model described in Table 2.1 and decides when to invoke BookSim with a threshold of 4 and the window size that yields the fastest BZSim ($\frac{ZLL}{2}$ or $\frac{ZLL}{4}$). Finally, the original ZSim setup does not model a network and is equivalent to BZSim with network modeling disabled (threshold = ∞).

Figure 2.9 shows the comparison of BZSim, gem5, and ZSim simulation speed in Million Instructions per Second (MIPS). Gem5 is able to simulate 0.4-0.8 MIPS and on average 0.6 MIPS. BZSim is on average 9.5× faster than gem5 and offers a simulation speed of 0.5-29 MIPS and on average 5.9 MIPS. BZSim skips the simulation of 60% of the traffic on average and it is slower (less than 2 MIPS) on benchmarks with significant network congestion that

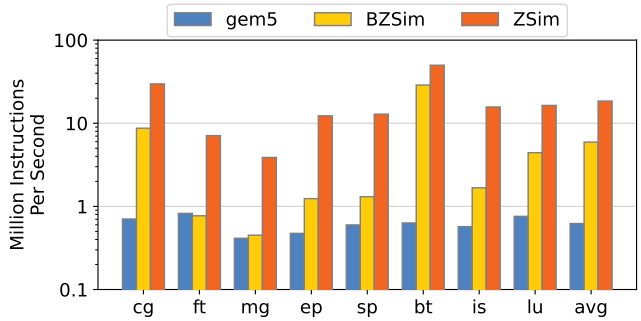


Figure 2.9: Simulation speed comparison between gem5, BZSim, and ZSim.

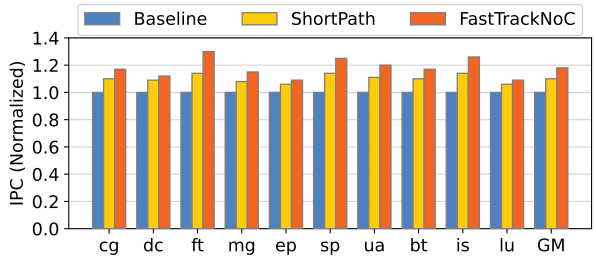
causes most traffic to be simulated. Finally, ZSim simulations are the fastest, roughly 3× faster than BZSim and 30× faster than gem5, simulating 3.9 to 50 MIPS and on average 18.5 MIPS. In summary, BZSim offers at least an order of magnitude faster simulation speed than alternatives that include network modeling (gem5) taking advantage of its ability to trade accuracy for speed.

2.6.3 Impact of NoC design on system performance

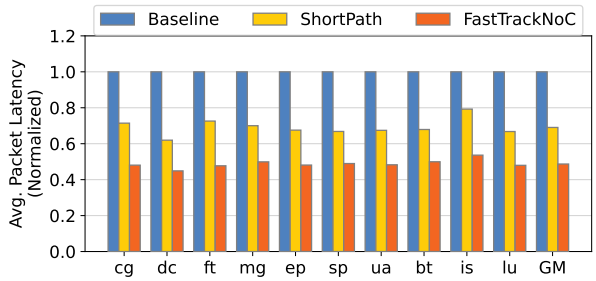
As discussed in Section 2.2.3, FastTrackNoC currently offers the highest network throughput and lowest packet latency using DDR datapaths and bypassing of both router control and switch traversal stages [19]. As discussed in literature [19], an 8×8 FastTrackNoC with uniform random synthetic traffic, offers 10% lower latency and 20% higher throughput than the second best ShortPath, which uses fine-grain bypassing of individual control stages [19]. Moreover, it achieves about 4× lower latency and 50% higher throughput than a typical 3-stage baseline NoC with no bypassing support. However, it is unclear how this affects the performance of a system. Next, we attempt to answer exactly this question. The impact of the above two NoC designs is measured on a 32-core system with the characteristics reported in Table 2.1.

Figure 2.10 shows the speedup of a system using FastTrackNoC or ShortPath over the same system using a baseline NoC as well as the normalized average packet latency of the three networks when running NAS benchmarks. As depicted in Figure 2.10a, the latency and throughput advantage of a FastTrackNoC offers 9-30% and on average 18% higher system performance compared to the baseline NoC. Furthermore, a system that uses the ShortPath NoC improves baseline system performance by up to 6-14% and on average 10%. Figure 2.10b shows the average packet latency of the networks. ShortPath latency is 62-79% and on average 69% of the baseline NoC latency, and FastTrackNoC latency is 45-54% and on average 49% of the baseline NoC latency.

The above results show the strong impact of NoC designs on the performance of a large multicore system and the need for fast tools to assess this impact.



(a) Normalized IPC



(b) Normalized average packet latency

Figure 2.10: Evaluation of a 32-core system with different NoC designs.

2.7 Conclusions

Network modeling is important for simulating future computer architectures, but it is also computationally intensive. This work was based on the insight that a significant fraction of the traffic in a multicore system has low contention and on the conjecture that analytically calculating the latency of such traffic, rather than deriving it via simulation, introduces relatively small inaccuracies in the simulation results. In this work, BookSim, a widely used and accurate network simulator, was integrated into ZSim, one of the fastest parallel microarchitectural simulators, to produce the proposed BZSim. Based on the above observation and conjecture, BZSim detects and skips the simulation of low contention traffic to offer a new knob for trading simulation accuracy for speed. It is able to speed up simulations by 2-4× with a maximum error in normalized IPC and packet latency of 3-5% and 10-20%, respectively. BZSim is about 9× faster than an optimistic gem5 setup, which uses a point-to-point simple network model, and 3× slower than ZSim, which does not model a network. Finally, BZSim was used to assess the impact on system performance of the fastest NoC designs, which were modeled in BZSim, showing that a 32-core system can get an average speedup of about 1.2× when using the fastest NoC instead of a baseline NoC.

Acknowledgment

This work was supported by the Swedish Foundation for Strategic Research (contract number CHI19-0048) under the PRIDE project.

Bibliography

[1] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "Garnet: A detailed on-chip network model inside a full-system simulator," in *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, 2009, pp. 33–42.

[2] A. Akram and L. Sawalha, "A comparison of x86 computer architecture simulators," 2016.

[3] A. Akram and L. Sawalha, "A survey of computer architecture simulation techniques and tools," *IEEE Access*, vol. 7, pp. 78 120–78 145, 2019.

[4] F. Alazemi, A. AziziMazreah, B. Bose, and L. Chen, "Routerless network-on-chip," in *IEEE Int'l Symp. on High Performance Computer Architecture (HPCA)*, 2018, pp. 492–503.

[5] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga, "The nas parallel benchmarks," *Int. J. High Perform. Comput. Appl.*, vol. 5, no. 3, p. 63–73, sep 1991.

[6] A. Bakhoda, J. Kim, and T. M. Aamodt, "Throughput-effective on-chip networks for manycore accelerators," in *MICRO-43*, 2010, pp. 421–432.

[7] J. Balfour and W. J. Dally, "Design tradeoffs for tiled cmp on-chip networks," in *Int. Conf. on Supercomputing*, 2006, pp. 187–198.

[8] D. U. Becker and W. J. Dally, "Allocator implementations for network-on-chip routers," in *Conf. on HPC Net., Stor. & Anal.*, ser. SC, 2009.

[9] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, p. 1–7, aug 2011. [Online]. Available: https://doi.org/10.1145/2024716.2024718

[10] S. Borkar, "How to stop interconnects from hindering the future of computing!" in *2013 Optical Interconnects Conf.*, May 2013, pp. 96–97.

[11] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11)*, 2011.

[12] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Noxim: An open, extensible and cycle-accurate network on chip simulator," in *Int'ernationa'l Conf. on Application-specific Systems, Architectures and Processors (ASAP)*, 2015, pp. 162–163.

[13] C. O. Chen, S. Park, T. Krishna, S. Subramanian, A. P. Chandrakasan, and L. Peh, "Smart: A single-cycle reconfigurable noc for soc applications," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2013, pp. 338–343.

[14] D. Chiou, D. Sunwoo, J. Kim, N. A. Patil, W. Reinhart, D. E. Johnson, J. Keefe, and H. Angepat, "Fpga-accelerated simulation technologies (fast): Fast, full-system, cycle-accurate simulators," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 40, 2007, p. 249–261.

[15] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, 2004.

[16] B. K. Daya, C.-H. O. Chen, S. Subramanian, W.-C. Kwon, S. Park, T. Krishna, J. Holt, A. P. Chandrakasan, and L.-S. Peh, "Scorpio: A 36-core research chip demonstrating snoopy coherence on a scalable mesh noc with in-network ordering," in *Int. Symp. on Computer Architecuture*, ser. ISCA, 2014, pp. 25–36.

[17] A. Ejaz, V. Papaefstathiou, and I. Sourdis, "DDRNoC: Dual data-rate network-on-chip," *ACM Trans. Archit. Code Optim.*, vol. 15, no. 2, 2018.

[18] A. Ejaz, V. Papaefstathiou, and I. Sourdis, "Highwaynoc: Approaching ideal noc performance with dual data rate routers," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 318–331, 2021.

[19] A. Ejaz and I. Sourdis, "Fasttracknoc: A noc with fasttrack router datapaths," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 971–985.

[20] A. Ejaz, V. Papaefstathiou, and I. Sourdis, "Freewaynoc: A ddr noc with pipeline bypassing," in *Int'l Symp. on Networks-on-Chip (NOCS)*, 2018.

[21] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Express cube topologies for on-chip interconnects," in *HPCA*, Feb 2009, pp. 163–174.

[22] G. Hamerly, E. Perelman, J. Lau, and B. Calder, "Simpoint 3.0: Faster and more flexible program phase analysis," *Journal of Instruction-Level Parallelism*, vol. 7, pp. 1–28, 09 2005.

[23] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally, "A detailed and flexible cycle-accurate network-on-chip simulator," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013, pp. 86–96.

[24] A. Khan, M. Vijayaraghavan, S. Boyd-Wickizer, and Arvind, "Fast and cycle-accurate modeling of a multicore processor," in *Proceedings of the 2012 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS '12)*, 2012, p. 178–187.

[25] A. E. Kiasari, Z. Lu, and A. Jantsch, "An analytical latency model for networks-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 113–123, 2013.

[26] H. Kim, A. Vitkovskiy, P. V. Gratz, and V. Soteriou, "Use it or lose it: Wear-out and lifetime in future chip multiprocessors," in *MICRO-46*, 2013, pp. 136–147.

[27] J. Kim, "Low-cost router microarchitecture for on-chip networks," in *42nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 42, 2009, p. 255–266.

[28] J. Kim, J. Balfour, and W. Dally, "Flattened butterfly topology for on-chip networks," in *MICRO-40*, 2007, pp. 172–182.

[29] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das, "A low latency router supporting adaptivity for on-chip interconnects," in *Design Automation Conf. (DAC)*, 2005, pp. 559–564.

[30] T. Krishna, J. Postman, C. Edmonds, L. S. Peh, and P. Chiang, "Swift: A swing-reduced interconnect for a token-based network-on-chip in 90nm cmos," in *IEEE ICCD*, 2010, pp. 439–446.

[31] T. Krishna, C.-H. O. Chen, W.-C. Kwon, and L.-S. Peh, "Smart: Single-cycle multihop traversals over a shared network on chip," *IEEE Micro*, vol. 34, no. 3, pp. 43–56, 2014.

[32] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Towards the ideal interconnection fabric," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, 2007.

[33] P. Lotfi-Kamran, B. Grot, and B. Falsafi, "Noc-out: Microarchitecting a scale-out processor," in *IEEE/ACM MICRO-45*, Dec 2012, pp. 177–187.

[34] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," in *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '05, 2005, p. 190–200.

[35] A. K. Mishra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. R. Das, "A case for dynamic frequency tuning in on-chip networks," in *MICRO-42*, 2009, pp. 292–303.

[36] S. Y. Narayana, S. K. Mandal, R. Ayoub, M. M. Islam, M. Kishinevsky, and U. Y. Ogras, "Fast analysis using finite queuing model for multilayer nocs," *IEEE Design & Test*, vol. 40, no. 6, pp. 112–124, 2023.

[37] C. Nicopoulos, V. Narayanan, and C. R. Das, *Network-on-Chip Architectures - A Holistic Design Exploration*, ser. Lecture Notes in Elect. Eng. Springer, 2010, vol. 45.

[38] L.-S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Int. Symp. on HPCA*, 2001.

[39] A. Psarras, S. Moisidis, C. Nicopoulos, and G. Dimitrakopoulos, "Networks-on-chip with double-data-rate links," *IEEE Trans. on Circuits and Systems*, vol. 64, no. 12, pp. 3103–3114, 2017.

[40] A. Psarras, I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos, "ShortPath: A Network-on-Chip Router with Fine-Grained Pipeline Bypassing," *IEEE ToC, 65, 10*, pp. 3136–3147, 2016.

[41] A. Psathakis, V. Papaefstathiou, N. Chrysos, F. Chaix, E. Vasilakis, D. Pnevmatikatos, and M. Katevenis, "A systematic evaluation of emerging mesh-like cmp nocs," in *ANCS*, 2015, pp. 159–170.

[42] I. Pérez, E. Vallejo, M. Moretó, and R. Beivide, "Bst: A booksim-based toolset to simulate nocs with single- and multi-hop bypass," in *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2020, pp. 47–57.

[43] S. Rao, S. Jeloka, R. Das, D. Blaauw, R. Dreslinski, and T. Mudge, "Vix: Virtual input crossbar for efficient switch allocation," in *Design Automation Conf. (DAC)*, 2014, pp. 103:1–103:6.

[44] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "Dramsim2: A cycle accurate memory system simulator," *IEEE Computer Architecture Letters*, vol. 10, no. 1, pp. 16–19, 2011.

[45] D. Sanchez and C. Kozyrakis, "Zsim: Fast and accurate microarchitectural simulation of thousand-core systems," in *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA '13)*, 2013, p. 475–486.

[46] SNU, "SNU NPB Suite," http://aces.snu.ac.kr/software/snu-npb/.

[47] S. Wilton and N. Jouppi, "Cacti: an enhanced cache access and cycle time model," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 5, pp. 677–688, 1996.

[48] R. E. Wunderlich, T. F. Wenisch, B. Falsafi, and J. C. Hoe, "Statistical sampling of microarchitecture simulation," *ACM Trans. Model. Comput. Simul.*, vol. 16, no. 3, p. 197–224, jul 2006.

Paper II

A Performance Analysis of Chiplet-Based Systems

Neethu Bal Mallya, **Panagiotis Strikos**, Bhavishya Goel, Ahsen Ejaz, and Ioannis Sourdis

2025 Design, Automation and Test in Europe Conference and Exhibition (DATE), Lyon, France, Mar 2025

Chapter 3

A Performance Analysis of Chiplet-Based Systems

Abstract

As the semiconductor industry struggles to keep Moore's law alive and integrate more functionality on a chip, multi-chiplet chips offer a lower cost alternative to large monolithic chips due to their higher yield. However, chiplet-based chips are naturally Non-Uniform Memory Access (NUMA) systems and therefore suffer from slow remote accesses. NUMA overheads are exacerbated by the limited throughput and higher latency of inter-chiplet communication. This paper offers a comprehensive analysis of chiplet-based systems with different design parameters measuring their performance overheads compared to traditional monolithic multicore designs and their scalability to system and chiplet size. Several design alternatives pertaining to the memory hierarchy, interconnects, and technology aspects are studied. Our analysis shows that although chiplet-based chips can cut (recurring engineering) costs to half, they may give away over a third of the monolithic performance. Part of this performance overhead can be regained with specific design choices.

3.1 Introduction

In the multicore era, integrating more resources on a chip is evermore important for the performance scaling of processors. In the past couple of decades, frequency scaling has been limited by power density, and therefore, delivering performance speedup relies primarily on fitting more cores on a chip. However, technology scaling has become more difficult, and large monolithic chips have low yields and, thus, excessive costs. Building chips out of multiple smaller chiplets is a cheaper, higher yield alternative [10, 15, 18, 28].

Die stacking technology has enabled multi-chiplet chips. It was first used for building 3D stacked DRAM chips such as High Bandwidth Memory (HBM) and bringing it closer to processing units, e.g., to a GPU [6] or a vector engine [21]. Later it was employed for disintegrating processors to multiple chiplets, e.g., AMD EPYC and RYZEN architectures, improving yield [18]. Currently, large chips, such as the AMD MI300 [1] and Intel Sapphire Rapids [20], are composed of many CPU and/or GPU chiplets, as well as HBM nodes combining high processing throughput with fast, high-bandwidth memory access.

Despite their improved yield, multi-chiplet chips come with performance overheads. Due to their large size, such systems inevitably use multiple non-uniform access memory nodes. Even early AMD EPYC and RYZEN chips, which provide DRAM access to their CPU chiplets via a single IO die, have a varying access latency by tens of nanoseconds depending on the accessed DRAM controller [18]. AMD MI300 and Intel Sapphire Rapids have even more complex, heterogeneous memory systems composed of multiple HBM nodes and external DRAM. Non-uniform Memory Access (NUMA) machines entail the performance pitfall of long latency remote accesses. Although in the past Cache Only Memory Architectures (COMA) [8] and Cache Coherent NUMA (CC-NUMA) [33] approaches improved data locality and performance of NUMA multi-socket machines, current multi-chiplet chips rely mainly on code optimizations to improve data placement when operating in a flat “HBM + external DDR" mode, or otherwise sacrifice HBM capacity to cache data.

Another performance overhead in multi-chiplet chips, which exacerbates the NUMA effects, is related to the inter-chiplet communication. As opposed to networks on monolithic chips, inter-chiplet connections suffer latency and bandwidth overheads. Exchanging messages with another chiplet requires traversing longer links via microbumps and a silicon interposer, adding extra latency. In addition, the number of available microbumps is limited by the chiplet size and their density constraints, putting a cap on available off-chiplet bandwidth.

Although the yield and cost benefits of multi-chiplet chips have been thoroughly analyzed [10], to the best of our knowledge, the performance with respect to their monolithic counterparts has not been studied. This work fills this gap by evaluating the aforementioned performance overheads of multi-chiplet chips compared to monolithic ones and analyzing the cost-performance trade-off they offer. We explore the following design points in this study: (i) system size, (ii) chiplet size, (iii) Network-on-Chip (NoC) bandwidth, (iv) Last-Level-Cache (LLC) organization, and (v) silicon interposer type (passive vs. active). We measure the performance overheads of chiplet-based chips varying the above design alternatives and analyze them based on the insight provided by several interconnect and memory system metrics.

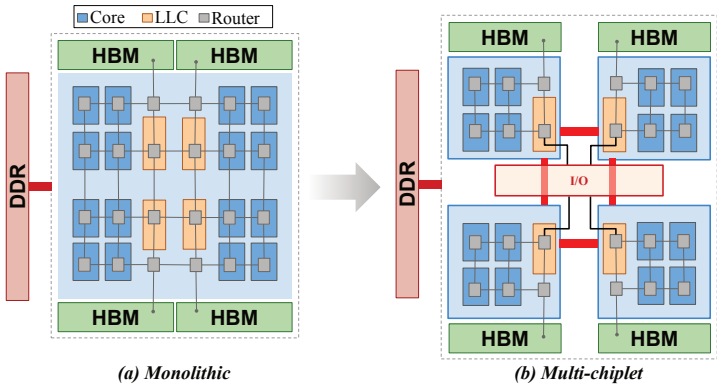


Figure 3.1: Monolithic vs. Multi-Chiplet chips.

Concisely, the contributions of this paper are the following:

- A microarchitectural simulation setup to model large-scale chiplet-based architectures, including detailed models of their memory system and interconnection network;
- The first thorough analysis of performance overheads and cost-performance trade-offs of chiplet-based chips in comparison to monolithic chips, showing that despite their large cost benefit, chiplet-based designs incur a significant impact on system performance;
- An analysis of various technological aspects that determine specific system parameters such as the length and delay of inter-chiplet links, microbumps budget, yield and cost of chiplet-based and monolithic chips;
- Some design choices are identified to regain some of the performance overheads of chiplet-based chips.

The remainder of this paper is organized as follows: Section 3.2 describes the design alternatives analyzed for chiplet-based architectures. Section 3.3 explains our experimental methodology. Section 3.4 presents our evaluation results. Finally, Section 3.5 summarizes our conclusions.

3.2 Overview of Chiplet-Based Architectures

The microarchitecture of the chiplet-based chips studied in this paper as well as the monolithic chips used as baselines, are described next. Without loss of generality, the multicore systems are organized in tiles composed of a core with its private L1 and L2 caches, a shared Last-Level Cache (LLC) partitioned in slices, each being closer to a subset of tiles, as well as HBM and external DDR DRAM interfaces. The above are interconnected via a Network-on-chip (NoC), which dedicates a network router for each tile, LLC slice, HBM node, and external DDR controller. The organization and floorplan of the chips illustrated in Figure 3.1 are inspired by the AMD Zen families [2, 4]. A monolithic chip uses a 2D mesh NoC, has its LLC slices in the middle columns of the 2D mesh, and the HBM and external DDR interfaces at the edges of the chip.

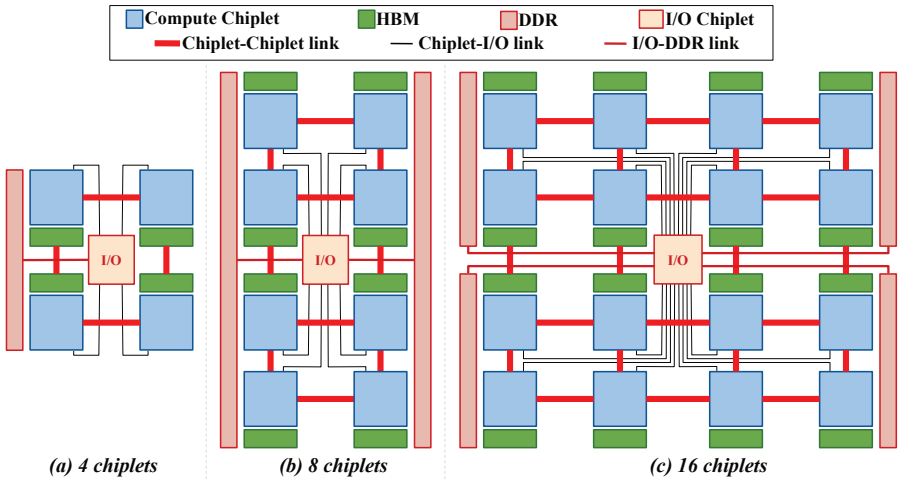


Figure 3.2: Different sizes of multi-chiplet chips.

The chiplet-based counterpart uses chiplets, which contain a subset of tiles, one or multiple LLC slices and HBM interfaces, while access to external DDR is provided via a separate IO chiplet. The NoC topology within the chiplet remains a 2D mesh, and inter-chiplet links are reduced to a number that can be supported by the microbumps budget of the chiplet, similar to previous work [11, 12].

Figure 3.2 illustrates the above multi-chiplet organization for different system sizes, i.e., different number of chiplets. It can be observed that the number of HBM nodes scales linearly to the number of chiplets, i.e., one HBM node per chiplet placed next to it. Moreover, the external DDR size and the number of channels also scale linearly to the number of chips.

In the rest of the section, details are provided for system aspects that affect the potential performance overheads of chiplet-based chips. In particular, it describes (i) the design of the interconnects with a focus on inter-chiplet communication and the choice of silicon interposer, (ii) the memory allocation that dictates data placement on the Non-Uniform Memory Access system, and (iii) the choice of the LLC organization. Finally, the yield and cost of the chiplet-based chips are estimated with respect to their monolithic counterparts.

3.2.1 Chiplet-based NoCs

One of the first design choices involves deciding whether to use a passive or active silicon interposer for integrating the chiplets. A passive silicon interposer is cheaper as it requires fewer fabrication steps and offers a higher yield, but it offers slower inter-chiplet links because it does not include buffers. An active interposer offers higher throughput because it includes active components to pipeline the links and potentially lower link latency. It may also include network routers offering more advanced topologies. Additionally, an active interposer benefits from lower clock skew/jitter due to repeaters and easier clock synchronization. Minimally active silicon interposers have been shown to have a small cost overhead compared to passive ones [13, 27]. Another design

choice studied in previous work is the placement of inter-chiplet links, showing the benefits of concentrating inter-chiplet links to a few edge NoC routers of a chiplet [11, 12].

In our performance analysis, passive silicon interposers as well as minimally active, i.e., with pipelined links, are explored. The total budget of microbumps per chiplet is estimated based on technology parameters, and inter-chiplet links are concentrated to fewer NoC edge routers, adjusting their width accordingly. Finally, various intra-chiplet NoC datapath widths are explored, offering different intra-chiplet communication bandwidths.

3.2.2 NUMA-aware Memory Allocation

The placement of data in DRAM (HBM and external DDR) is critical for the performance of a NUMA system. Modern operating systems widely support Non-Uniform Memory Access (NUMA) architectures through various mechanisms. For instance, operating systems like Linux [26], Windows [3], and FreeBSD [5] implement NUMA-aware scheduling algorithms that place processes and threads closer to the memory nodes. This approach helps to reduce access latency by optimizing memory locality. Additionally, these operating systems provide APIs that allow user applications to discover the NUMA topology, request memory from specific nodes, and set process affinity, enhancing performance for NUMA-enabled systems.

In this study, we use a Distance-aware Memory Allocation policy. This policy allocates physical memory pages to the memory node closest to the processor core that first accesses the memory. This approach can significantly improve performance by ensuring that memory is allocated in proximity to the accessing core, reducing the latency of memory access.

3.2.3 Last Level Cache Organization

In multi-chiplet systems, the Last-Level Cache (LLC) can be organized through two primary designs:

Sliced LLC: The Sliced LLC architecture, pioneered by Intel starting with the Sandy Bridge microarchitecture, distributes the LLC into multiple "slices" [32]. Each slice acts as an independent cache, but all the slices together form a single logical cache. The physical memory address determines the slice into which data is loaded, effectively distributing memory addresses across slices, and thereby enhancing effective memory bandwidth. In our study, we assign one LLC slice per chiplet and evenly divide the address space among the LLC slices, corresponding to the number of chiplets or High Bandwidth Memory (HBM) nodes associated with a chiplet. The addresses mapped to the external DDR are also divided and assigned into these slices. Accesses from core private caches mapped to the local LLC slice exhibit lower latency, whereas accesses mapped to remote LLC slices have to traverse inter-chiplet links, resulting in higher access latency.

Private LLC: Unlike the sliced LLC architecture where the logical LLC cache is distributed across chiplets, in the private LLC architecture, each chiplet is assigned its own private LLC cache, and the entire address range is mapped to that private LLC. As a result, all the accesses from the core private

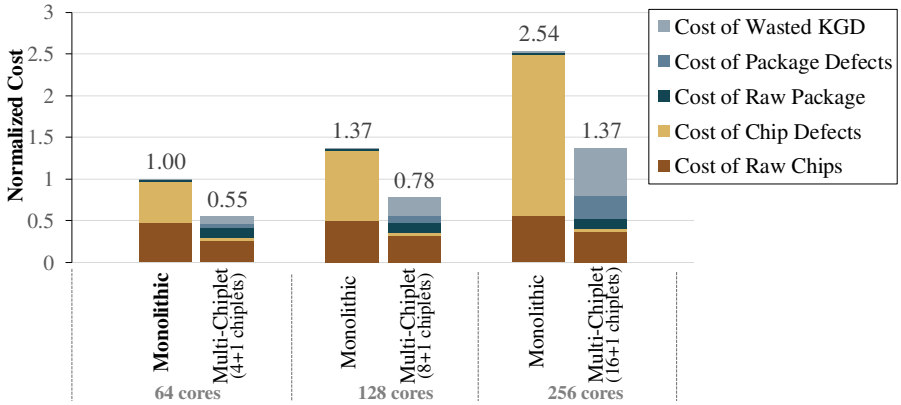


Figure 3.3: Cost analysis and comparison of RE cost of monolithic and multi-chiplet chips for different system sizes. Cost is normalized to that of the smallest monolithic.

caches first go to the local LLC, and only in the case of LLC miss may be required to traverse the inter-chiplet communication link (depending on where the data is mapped in the memory). This design results in reduced inter-chiplet communication overhead compared to the sliced LLC architecture. However, since the same address can now be present in different LLCs across chiplets, LLCs need to be kept coherent, requiring a complex inter-chiplet coherence mechanism. Despite these challenges, the Private LLC approach can offer performance benefits in scenarios where the coherence overhead is manageable or inter-chiplet communication latency is critical.

3.2.4 Yield and Cost

The performance analysis of the above multi-chiplet design alternatives needs to be complemented with an evaluation of their cost with respect to their monolithic alternative. This is performed using the Feng-Ma chiplet actuary model [10] with the parameters of the evaluated systems in our study.

More precisely, our work considers monolithic and multi-chiplet chips which are based on AMD EPYC microarchitecture composed of Zen4/Zen4c CPU chiplets manufactured at 5 nm and an IO chiplet at 14 nm. The chiplet area is estimated, considering 16-core Zen4 chiplets after scaling the L2 and L3 cache sizes to what is used in our performance evaluation, as depicted in Table 3.1. That results in a chiplet size of 66 mm^2, which is similar to the AMD Zen4 chiplets and slightly smaller than the AMD Zen4C chiplet. In addition, each multi-chiplet chip includes an IO chiplet of 400 mm^2, as estimated from AMD EPYC chips of similar technology. Finally, a passive interposer of 65 nm technology is considered.

Based on the above parameters, the above cost model was used to derive the costs of 64-, 128-, and 256-core systems, which are divided in the case of chiplet-based chips to 4+1, 8+1, and 16+1 chiplets, respectively, including the IO chiplet [10]. Figure 3.3 presents a breakdown of the detailed recursive engineering (RE) cost. The total cost is the sum of the following: (i) cost of raw chips which includes among others the cost of the silicon and the processing during the wafer fabrication, (ii) cost of raw package which covers the materials

that are necessary for assembling and packaging the chip as well as testing and verification, (iii) cost of wasted known good dies (KGD) that encapsulates the cost derived from dies that already have been tested to ensure their correct functionality, but still fail, (iv) cost of chip defects, that covers the cost of defects that occur during the wafer fabrication process, and (v) cost of package defects which includes costs related to the packaging process. The total cost is then normalized to that of the smallest monolithic chip.

Overall, the cost of chiplet-based chips is about 55% of that of monolithic chips. The gap between the monolithic and multi-chiplet costs seems to only slightly increase because of the conservative estimation of the model regarding the bonding and packaging multi-chiplet yield. Nevertheless, the model confirms the significant savings of chiplet-based approaches and puts our performance analysis in perspective.

3.3 Experimental Methodology

3.3.1 System Configuration

Our microarchitectural simulation offers detailed modeling of the memory subsystem and interconnection network, as explained in Section 3.3.2, and therefore is computationally intensive for large systems. In order to keep the simulation times of our experiments within affordable bounds (tens of hours per simulation point), the modeled systems are scaled down to a quarter of a real one. A full-scale AMD Zen4C chiplet contains 16 cores, as many considered in our cost analysis of Section 3.2.4[1]. As a consequence, our performance analysis considers chiplets scaled to be a quarter of a chiplet AMD Zen4C or Intel Sapphire Rapids chiplet and as such they contain a quarter of the number of cores and connect to a quarter of HBM channels, as shown in Table 3.1. Moreover, the L2 and L3 caches are undersized in order to put more pressure on the memory system and increase LLC misses per kilo instructions (MPKI), which is otherwise difficult to achieve when simulating systems for only a few billion instructions.

Based on the scaled down chiplet size ($16.5\,\mathrm{mm}^2$)[2], the microbump budget is calculated to be proportional to the number of cores it includes. In addition, the following parameters were taken into account for calculating the microbump budget: (i) a microbump pitch of $45\,\mu\mathrm{m}$, (ii) reserving 40% of the microbumps for power. Then, the number of microbumps available for data were allocated for (i) connecting to the HBM channels, (ii) one bidirectional link to the IO chiplet, (iii) multiple bidirectional links to the other CPU chiplets. Then, the width of the links to IO and CPU chiplets, as well as the total number of links to other CPU chiplets, were adjusted to fit the microbump budget. Finally, the latency of the inter-chiplet links was measured to be 2 or 3 (NoC) clock cycles considering the chiplet's dimensions and the latency of the links on the silicon interposer similar to [7, 28].

[1] A yield and cost analysis of a scaled down chiplet would not make sense as the size of the chiplets would be small, and so would be the size of the monolithic chip making it too small to break down into chiplets.

[2] Calculated based on Zen4 after scaling down L2 and L3 sizes proportional to the capacity indicated in Table 3.1.

Table 3.1: System Configuration[1]

System	
Chiplets	4 chiplets[1]
Cores and Caches	
Cores	4 cores[1] / chiplet, out-of-order, 3.2 GHz
TLB	I-TLB: 512-entry, 4-way, 1 cycle latency D-TLB: 512-entry, 4-way, 1 cycle latency
L1 Cache	L1-I: Private, 32KB, 4-way, 2 cycle access latency L1-D: Private, 32KB, 4-way, 2 cycle access latency
L2 Cache	Private, 256KB, 8-way, 4 cycle access latency
L3 Cache	Shared, 1MB/core, 16-way, 12 cycle access latency[2]
Main Memory	
HBM2	1 GB/chiplet, 2 GHz, 4 channels, 128 bits per channel, tCAS-tRCD-tRP: 14-14-14 ns
DDR4	4 GB, 3.2 GHz, 1 channel, 64 bits per channel, tCAS-tRCD-tRP: 22-22-22 ns
Network	
Intra-chiplet	2 GHz, 3-stage router (VA/SA, ST, LT), 2x3 Mesh, 4 VCs per port, credit-based flow control, 256 bit link for data, 154 bit link for control (coherence) traffic, 5 flit buffers, XY Routing [9]
Inter-chiplet	2 GHz, 3-stage router (VA/SA, ST, LT), 2x2 Mesh, passive interposer, 2 to 3 cycle link latency[3], 7 to 9 flit buffers [3]

[1] This configuration is the default setting. The parameter adjustments are detailed in the respective evaluation sections of the sensitivity studies.
[2] L3 access latency is 8 cycles for 2MB, 12 for 4MB, and 15 for 8MB.
[3] Depending on the maximum inter-chiplet link length [28].

3.3.2 Simulation Setup

BZSim simulator was used for our experiments [29], extended to model memory system and interconnects of chiplet-based chips. BZSim is based on ZSim simulator [25] integrated with BookSim2 [14] for cycle-accurate intra- and inter-chiplet network modeling, enhanced with a technique to detect and skip simulation of low contention traffic in order to speed up simulation times. BZSim offers microarchitectural simulations with detailed (cycle-accurate) interconnect modeling at an order of magnitude faster simulation speeds compared to GEM5, enabling multi-billion instruction experiments within reasonable times [29]. DRAMSim3 [16] was used for cycle-accurate DRAM modeling and CACTI [31] for estimating cache access times.

The system treats all HBM and external DDR memory as part of a unified flat address space. The virtual memory system was implemented based on HSCC [17]. The cores are configurable with translation lookaside buffers (TLBs) for both instructions and data, as well as with page table walkers (PTWs). Additionally, the memory management modules include a distance-aware allocation policy. This policy allocates pages to the HBM in the chiplet where they are first accessed. If pages are unavailable in the nearest HBM, they are allocated in the next neighboring HBM or in the external DDR.

Table 3.2: Workload Characteristics

Benchmark	Label	Input	LLC MPKI	Footprint (GB)	Assigned to Mixes mix-id$^{\#of instances}$
LLC MPKI 20-40					
pageRank[2]	PRL2	LDBC (100k)	37.41	0.84	1^3,2^3,3^2,4^3,5^2,6^3,7^3,8^1
mcf[1]	MCF	Default	34.01	0.45	2^2,6^3,8^2
graphColoring[2]	GCL2	LDBC (100k)	30.70	0.45	1^1,2^1,4^1,5^2,7^1,8^2
graphColoring[2]	GCL3	LDBC (10k)	21.26	0.09	1^2,2^1,3^2,6^2,8^1
Random Access Workload[3]	RAND	N=30, M=1000, chunk=1024	20.83	0.70	1^1,2^3,3^1,4^2,6^2,7^1,8^1
LLC MPKI 10-20					
connectedComp[2]	CCL3	LDBC (10k)	19.33	0.09	1^3,2^2,3^1,4^3,5^1,6^2,7^2,8^1
lbm[1]	LBM	Default	18.19	0.40	3^1,7^2,8^1
BFS[2]	BFSCR	CA RoadNet	17.25	0.64	1^1,2^1,3^1,4^2,5^3,7^2,8^1
fotonik3d[1]	FOTO	Default	17.07	0.59	1^1,4^1
pageRank[2]	PRL3	LDBC (10k)	13.96	0.09	2^1,4^1,5^1
xalancbmk[1]	XAL	Default	13.62	0.16	1^1,2^1,3^2,4^1,6^2,7^3,8^1
blender[1]	BLEN	Default	12.78	0.08	2^1,4^1,5^1
shortestPath[2]	SPCR	CA RoadNet	12.30	0.64	1^1,3^2,5^1,7^1,8^1
XSBench[4]	XSB	XXL	11.11	0.37	5^1,8^1
graphColoring[2]	GCCR	CA RoadNet	10.69	0.63	1^1,3^1,5^2,6^1,8^1
LLC MPKI 0-10					
parest[1]	PAR	Default	8.54	0.05	3^1
roms[1]	ROMS	Default	7.58	0.25	8^1
triangleCount[2]	TCL2	LDBC (100k)	6.24	0.55	6^1,8^1
graphColoring[2]	GCL1	LDBC (1000k)	5.92	0.29	1^1,4^1,7^1
pageRank[2]	PRKR	Knowledge Repo	4.56	0.30	3^1,5^1
omnetpp[1]	OMN	Default	4.53	0.16	5^1
BFS[2]	BFSL1	LDBC (1000k)	2.71	0.98	3^1

[1] SPEC CPU 2017 [22], [2] GraphBIG [19], [3] GUPS [24], [4] XSBench [30]

3.3.3 Workloads

We use mixes of multi-programmed workloads from the SPEC CPU2017 benchmark suite [22] (the eight with the highest MPKI), GraphBIG [19], Random access workload from the GUPS suite [24] and XSBench [30] in our experiments. For the SPEC CPU2017 and GraphBIG benchmarks, we use Simpoints [23] to select a representative slice of one billion instructions. We have chosen 22 different workloads, detailed in Table 3.2, and created random multi-programmed mixes of 16 applications designed to run on a system with 16 cores. Each mix of applications has a minimum total memory footprint of 7 GB and a geometric mean LLC MPKI of at least 11. To scale these mixes for systems with 32 or 64 cores, we replicate the 16-application mix twice for the 32-core system and four times for the 64-core system. All experiments run with an average of 125 million instructions per core warm-up period, where memory allocation is enabled, followed by an average of 250 million instructions per core of detailed simulation.

3.3.4 Evaluated Systems

We evaluate three distinct systems as follows:

1. **Chiplet-based System (CS)**: A multi-chiplet system with sliced LLC is the focus of our evaluation. The default configuration (depicted in Table 3.1) consists of 4 chiplets, each with 4 cores, integrated on a passive interposer with one LLC slice per chiplet, 256 bit NoC data-links, 4 HBM channels per chiplet, one link to IO chiplet, and one channel to external DDR. The above parameters change in the various sensitivity analyses of the evaluation. One variation of this design is to use chiplets with private, rather than sliced, LLC, denoted as **CP**.

2. **Monolithic (MN)**: A monolithic multicore matching the CS characteristics. Similarly, the default monolithic configuration is a 16-core system with sliced LLC in 4 parts and 256 bit NoC data-links, 16 HBM channels and one external DDR channel.

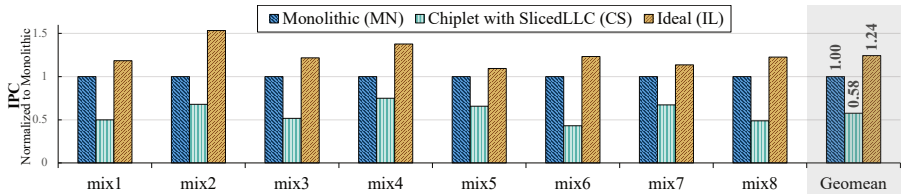
3. **Ideal (IL)**: An ideal chiplet-based system with the ideal scenario where an LLC miss is always served by the closest HBM channel, assuming the local HBM has infinite capacity, so memory allocation occurs solely within this local HBM. As a result, all memory requests remain local to the chiplet, eliminating the additional latency associated with accessing remote HBM or external DDR.

3.4 Performance Evaluation

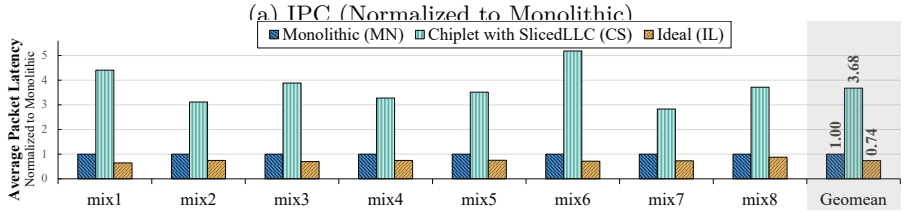
The performance of chiplet-based systems is evaluated and their overheads with respect to monolithic counterparts are measured. System performance is measured in terms of Instructions Per Cycle (IPC). The Average Memory Access Time (AMAT) is also measured and broken down to: (i) the access time for each cache level, (ii) the Network-on-Chip (NoC) latency between each level (L2-L3 and L3-DRAM), and (iii) the DRAM access time. Furthermore, the Average Packet Latency (APL) and the percentage of accesses to local and remote HBM nodes, as well as to external DDR are reported.

The performance evaluation is structured as follows: first, the default chiplet-based system is compared against the monolithic and ideal systems. Subsequently, a sensitivity analysis of the system size is conducted to examine how performance scales as the number of chiplets increases. Next, the impact of chiplet granularity is explored by analyzing different chiplet sizes (i.e., cores per chiplet) while keeping the system size fixed. Then, the performance of an alternative LLC organization for chiplet-based designs (private LLC per chiplet) is evaluated in comparison with the default sliced LLC. Next, a sensitivity analysis of the intra-chiplet NoC data bandwidth (datapath) is performed. Finally, the impact of passive versus minimally active interposer is measured.

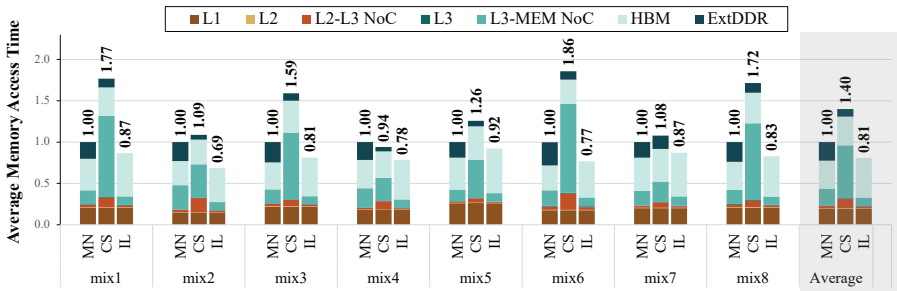
Multi-Chiplet vs. Monolithic vs. Ideal: Figure 3.4 shows, per mix of programs, the average performance (IPC), average packet latency, average memory access time, and the breakdown of DRAM accesses for the default configuration of a 4-chiplet system as well as for the equivalent 16-core monolithic and ideal systems. The chiplet-based system is able to maintain only



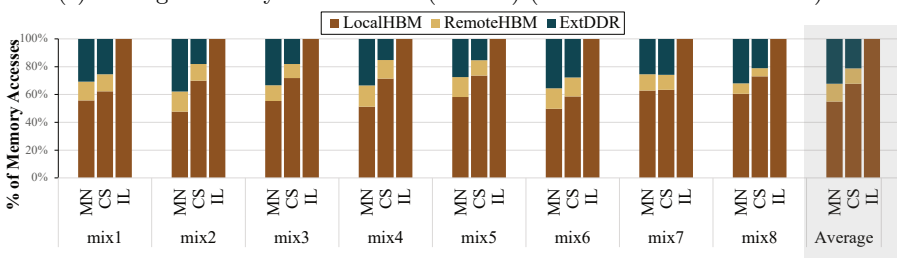
(a) IPC (Normalized to Monolithic)



(b) Average Packet Latency (Normalized to Monolithic)



(c) Average Memory Access Time (AMAT) (Normalized to Monolithic)



(d) Memory Accesses

Figure 3.4: Multi-Chiplet vs. Monolithic vs. Ideal

43%-75% of the monolithic performance and on average 58%, as shown in Figure 3.4a. Compared to the ideal system, which is 24% faster than the monolithic, and its LLC misses always go to the closest HBM, the chiplet-based system is 54% slower. The performance overhead of the chiplet-based system versus the monolithic is not explained by just observing AMAT, which is on average 40% higher than the monolithic. A more detailed look in Figure 3.4c reveals that the longer monolithic AMAT is due to slow external DDR accesses, rather than longer NoC and cache access latency, which is more performance critical and hence puts a heavier toll on chiplet-based performance. In fact, Figure 3.4b confirms the 3.7× longer packet latency of chiplet-based systems compared to monolithic. Finally, it is interesting to analyze the primary source of the performance overheads in the chiplet-based system, which is a fraction of accesses to data placed remotely. Figure 3.4d shows that on average 19% of

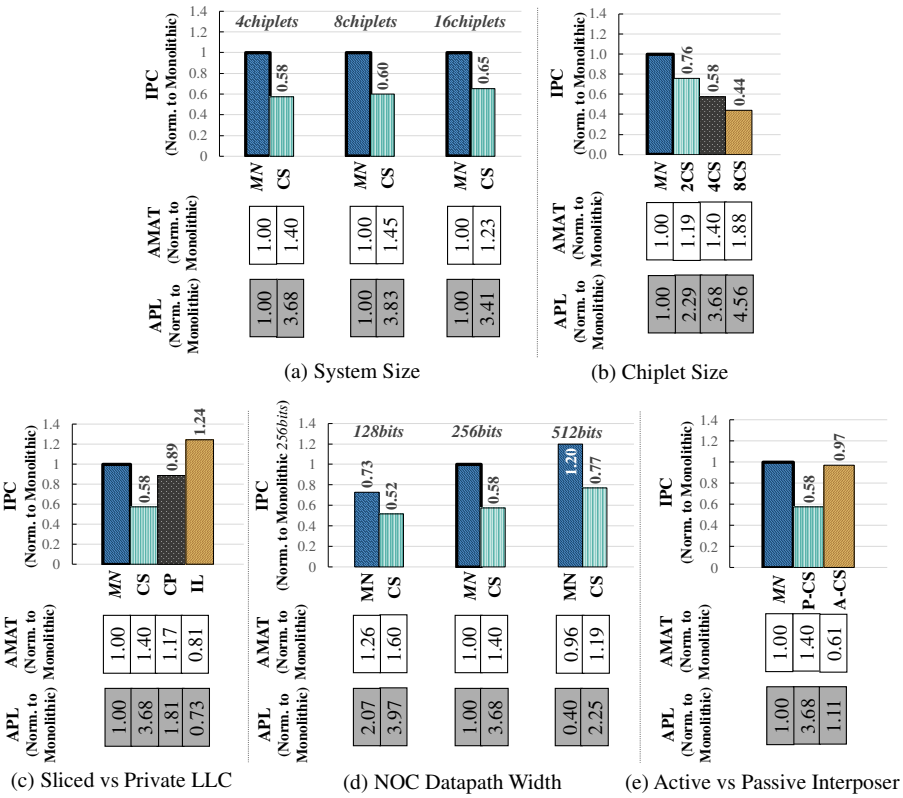


Figure 3.5: Sensitivity analyses* on system size, chiplet size, LLC organization, NoC datapath width, and interposer type.

* Geomean values of 8 mixes are shown, normalized to **MN**, as indicated by the thicker outline of the bars; **AMAT**: Average Memory Access Time; **APL**: Average Packet Latency; **MN**: Monolithic, **CS** or **CP**: Chiplet-based with Sliced or Private LLC, **IL**: Ideal, **P-CS** or **A-CS**: Chiplet-based with Passive or Active Interposer.

the data accesses are to the external DDR and 10% of the accesses are to a remote HBM node. For a chiplet-based system, as opposed to a monolithic one, all these accesses involve slow (due to limited bandwidth and longer latency) inter-chiplet communication.

Sensitivity Analysis on System Size: An interesting sensitivity analysis is with respect to the system size. Chiplet-based chips are meant to scale better to larger systems in terms of cost. However, it is unclear how their performance overheads change when system size increases while keeping the chiplet size constant. The performance of systems with 4, 8, and 16 CPU chiplets, i.e., 16, 32, and 64 cores, respectively, are evaluated. As shown in Figure 3.5(a), as the system size increases, the performance gap between chiplet-based systems and their respective monolithic of the same size remains stable or even slightly reduces, ranging from 58% to 65%. This is attributed primarily to the distance-aware data placement, which allows the distance of remote accesses to remain relatively stable.

Sensitivity Analysis on Chiplet Size: Next, a sensitivity analysis on

the chiplet size, i.e., the number of cores included in a chiplet, is performed for chiplets of 2 (2CS), 4 (4CS), and 8 cores (8CS) per chiplet on a 16-core system. Figure 3.5(b) shows the performance of these systems. As expected, performance reduces as the system is disintegrated to a larger number of chiplets. More precisely, 2, 4, and 8 chiplet systems offer 76%, 58%, and 44% of the monolithic IPC, respectively, which is reflected in their AMAT and APL measurements.

LLC Configuration Analysis - Sliced vs. Private: A chiplet-based system with sliced LLC would need to go to a remote chiplet to serve an L2 miss accessing a remote LLC slice if the memory address is mapped to the memory region of another chiplet. An interesting question arises as of the benefit of designing chiplets each with a private LLC. Such private LLC would store cache lines from the entire address space, regardless of whether the memory is mapped to local or remote DRAM (HBM and external DDR). Figure 3.5(c) depicts the results of this comparison. Chiplets with sliced LLC reduce IPC versus monolithic by 42%, while chiplets with private LLC reduce it by only 11%. A significant reduction in average packet latency of about 50% is achieved by using private LLC because, with this organization, L2 misses do not need to go off-chiplet, as opposed to LLC misses, which are fewer, and may need remote accesses. On the contrary, chiplets with sliced LLC may need remote accesses to serve L2 misses but not for LLC misses. This is also reflected in the AMAT, which is improved by 16% when using chiplets with private LLC compared to chiplets with sliced LLC.

Sensitivity Analysis on NoC Datapath Width: The performance impact of the NoC data-link bandwidth is analyzed for chiplet-based and monolithic systems. The intra-chiplet and monolithic NoC datapath varies from a quarter of a cache line (128 bits), to a full cache line (512 bits). It is worth noting that the width of inter-chiplet links remains constant as defined by the available microbump budget of the chiplets (64 bits). Figure 3.5(d) shows the IPC, AMAT and average packet latency of the three design points for chiplet-based and monolithic chips normalized to the 256-bit monolithic. It can be observed that wider (intra-chiplet) NoC links improve performance, although the gap with the respective monolithic does not follow a specific trend. Finally, as expected, AMAT improves for wider NoC datapaths, while at the same time, the gap between monolithic and chiplet-based average packet latency increases because the bottleneck of narrower inter-chiplet links is exacerbated.

Sensitivity Analysis on Active vs. Passive Interposer: The last design parameter explored in our study is the use of minimally active (A-CS) versus passive interposer (P-CS). As illustrated in Figure 3.5(e), a minimally active interposer increases both throughput and latency of inter-chiplet links because it can pipeline them. The performance impact of this is significant as it recovers most of the performance overhead on chiplet-based systems. That comes, however, at a higher system cost due to the lower yield of active interposers.

3.5 Conclusions

Semiconductor technology has difficulty scaling the integrated resources on a single die because monolithic chips have poor yield, leading to excessive costs. Multi-chiplet chips offer a cheaper alternative as they have a higher yield, but come with certain performance overheads stemming from their NUMA memory system and inter-chiplet interconnection bottlenecks. This paper analyzed these performance overheads. In particular, our study reveals that although chiplet-based chips reduce system costs by almost half compared to monolithic, they give away about a third of the monolithic performance. Our work further showed that part of this performance overhead can be regained with specific design choices. More specifically, designing chiplets with a private LLC improves performance by about 30%. Moreover, chiplet-based systems with active interposers are only 3% shy of the monolithic performance.

Acknowledgment

This work was supported by the Swedish Foundation for Strategic Research (contract number CHI19-0048) under the PRIDE project.

Bibliography

[1] Advanced Micro Devices, Inc., "AMD CDNA™ 3 Architecture," 2023. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/white-papers/amd-cdna-3-white-paper.pdf

[2] N. Beck, S. White, M. Paraschou, and S. Naffziger, "Zeppelin: An SoC for Multichip Architectures," in *2018 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2018, pp. 40–42.

[3] K. Bridge, "NUMA Support - Win32 Apps," May 2022, available: https://learn.microsoft.com/en-us/windows/win32/procthread/numa-support. Accessed: Oct. 11, 2024. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/procthread/numa-support

[4] T. Burd, N. Beck, S. White, M. Paraschou, N. Kalyanasundharam, G. Donley, A. Smith, L. Hewitt, and S. Naffziger, "Zeppelin: An SoC for Multichip Architectures," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 54, no. 1, pp. 133–143, Oct 2019.

[5] A. Chadd, "FreeBSD Manual Pages," Oct 2018. [Online]. Available: https://man.freebsd.org/cgi/man.cgi?query=numa&sektion=4&manpath=FreeBSD+14.0-RELEASE+and+Ports

[6] J. Choquette, W. Gandhi, O. Giroux, N. Stam, and R. Krashinsky, "NVIDIA A100 Tensor Core GPU: Performance and Innovation," *IEEE Micro*, vol. 41, no. 2, pp. 29–35, Mar 2021.

[7] A. Coskun, F. Eris, A. Joshi, A. B. Kahng, Y. Ma, and V. Srinivas, "A Cross-Layer Methodology for Design and Optimization of Networks in 2.5D Systems," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2018, pp. 1–8.

[8] F. Dahlgren and J. Torrellas, "Cache-Only Memory Architectures," *Computer*, vol. 32, no. 6, pp. 72–79, Jun 1999.

[9] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, USA: Morgan Kaufmann Publishers Inc., 2004.

[10] Y. Feng and K. Ma, "Chiplet Actuary: A Quantitative Cost Model and Multi-Chiplet Architecture Exploration," in *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC)*, Jul 2022, p. 121–126.

[11] Y. Feng, D. Xiang, and K. Ma, "A Scalable Methodology for Designing Efficient Interconnection Network of Chiplets," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb 2023, pp. 1059–1071.

[12] ——, "Heterogeneous Die-to-Die Interfaces: Enabling More Flexible Chiplet Interconnection Systems," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct 2023, p. 930–943.

[13] N. E. Jerger, A. Kannan, Z. Li, and G. H. Loh, "NoC Architectures for Silicon Interposer Systems: Why pay for more wires when you can get them (from your interposer) for free?" in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2014, pp. 458–470.

[14] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally, "A Detailed and Flexible Cycle-accurate Network-on-Chip Simulator," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Apr 2013, pp. 86–96.

[15] A. Kannan, N. E. Jerger, and G. H. Loh, "Enabling Interposer-based Disintegration of Multi-core Processors," in *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2015, pp. 546–558.

[16] S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. Jacob, "DRAMsim3: A Cycle-Accurate, Thermal-Capable DRAM Simulator," *IEEE Computer Architecture Letters (CAL)*, vol. 19, no. 2, pp. 106–109, Jul 2020.

[17] H. Liu, Y. Chen, X. Liao, H. Jin, B. He, L. Zheng, and R. Guo, "Hardware/Software Cooperative Caching for Hybrid DRAM/NVM Memory Architectures," in *Proceedings of the International Conference on Supercomputing (ICS)*, Jun 2017, pp. 26:1–26:10.

[18] S. Naffziger, N. Beck, T. Burd, K. Lepak, G. H. Loh, M. Subramony, and S. White, "Pioneering Chiplet Technology and Design for the AMD EPYC™ and Ryzen™ Processor Families : Industrial Product," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, Jun 2021, pp. 57–70.

[19] L. Nai, Y. Xia, I. G. Tanase, H. Kim, and C.-Y. Lin, "GraphBIG: Understanding Graph Computing in the Context of Industrial Solutions," in *SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2015, pp. 1–12.

[20] N. Nassif, A. O. Munch, C. L. Molnar, G. Pasdast, S. V. Lyer, Z. Yang, O. Mendoza, M. Huddart, S. Venkataraman, S. Kandula, R. Marom, A. M. Kern, B. Bowhill, D. R. Mulvihill, S. Nimmagadda, V. Kalidindi, J. Krause, M. M. Haq, R. Sharma, and K. Duda, "Sapphire Rapids: The Next-Generation Intel Xeon Scalable Processor," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2022, pp. 44–46.

[21] NEC CORPORATION, "SX-Aurora TSUBASA Architecture Guide Revision 1.1," 2018. [Online]. Available: https://sxauroratsubasa.sakura.ne.jp/documents/guide/pdfs/Aurora_ISA_guide.pdf

[22] R. Panda, S. Song, J. Dean, and L. K. John, "Wait of a Decade: Did SPEC CPU 2017 Broaden the Performance Horizon?" in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2018, pp. 271–282.

[23] E. Perelman, G. Hamerly, and B. Calder, "Picking Statistically Valid and Early Simulation Points," in *2003 12th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Sep 2003, pp. 244–255.

[24] S. J. Plimpton, R. Brightwell, C. Vaughan, K. Underwood, and M. Davis, "A Simple Synchronous Distributed-Memory Algorithm for the HPCC RandomAccess Benchmark," in *IEEE International Conference on Cluster Computing (Cluster 2006)*, Sep 2006, pp. 1–7.

[25] D. Sanchez and C. Kozyrakis, "ZSim: Fast and Accurate Microarchitectural Simulation of Thousand-core Systems," in *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA)*, Jun 2013, p. 475–486.

[26] K. Sarcar, "What is NUMA?" Nov 1999, available: https://www.kernel.org/doc/html/v5.4/vm/numa.html. Accessed: Oct. 11, 2024. [Online]. Available: https://www.kernel.org/doc/html/v5.4/vm/numa.html

[27] D. Stow, I. Akgun, and Y. Xie, "Investigation of Cost-Optimal Network-on-Chip for Passive and Active Interposer Systems," in *2019 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*, Jun 2019, pp. 1–8.

[28] D. Stow, Y. Xie, T. Siddiqua, and G. H. Loh, "Cost-Effective Design of Scalable High-performance Systems Using Active and Passive Interposers," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2017, pp. 728–735.

[29] P. Strikos, A. Ejaz, and I. Sourdis, "BZSim: Fast, Large-Scale Microarchitectural Simulation with Detailed Interconnect Modeling," in *2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, May 2024, pp. 167–178.

[30] J. R. Tramm, A. R. Siegel, T. Islam, and M. Schulz, "XSBench - The Development and Verification of a Performance Abstraction for Monte Carlo Reactor Analysis," in *Proceedings of the International Conference on Physics of Reactors (PHYSOR 2014)*, Sep 2014.

[31] S. J. Wilton and N. P. Jouppi, "CACTI: An Enhanced Cache Access and Cycle Time Model," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 31, no. 5, pp. 677–688, 1996.

[32] Y. Yarom, Q. Ge, F. Liu, R. B. Lee, and G. Heiser, "Mapping the Intel Last-Level Cache," *IACR Cryptology ePrint Archive*, Jan 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:1281871

[33] Z. Zhang and J. Torrellas, "Reducing Remote Conflict Misses: NUMA with Remote Cache versus COMA," in *Proceedings Third International Symposium on High-Performance Computer Architecture (HPCA)*, Feb 1997, pp. 272–281.

Appendix A
Additional Results

This appendix provides additional results on the cost analysis of various design options for chiplet-based systems. This supplementary material offers further insights and supports the findings presented in **Paper II**.

A Cost Analysis

As detailed in Section 3.2.4, this thesis evaluates the associated costs using the Feng-Ma chiplet actuary model [10], incorporating the parameters of the evaluated systems. Figure 3.6 presents a detailed breakdown of the recursive engineering (RE) cost for the different design options of chiplet-based systems and their monolithic counterpart. The total RE cost comprises:

(i) **Cost of raw chips**, which includes, among others, the cost of the silicon and the processing involved in wafer fabrication.
(ii) **Cost of chip defects**, which includes the cost of defects that occur during the wafer fabrication process.
(iii) **Cost of raw package**, which covers the materials necessary for assembling and packaging the chip, as well as testing and verification.
(iv) **Cost of package defects**, which includes the costs arising from defects during the packaging process.
(v) **Cost of wasted Known Good Dies (KGD)**, which encapsulates the cost of dies that have already been tested to ensure their correct functionality but still fail.

System Size: Using the cost model and parameters mentioned in Section 3.2.4, we derived the costs of 64-, 128-, and 256-core systems. In chiplet-based systems, these cores are partitioned into 4+1, 8+1, and 16+1 configurations, respectively, including an IO chiplet in each case. Figure 3.6a presents the cost values normalized to the monolithic of the smallest system. The results indicate that the chiplet-based designs reduce costs to approximately 55% of their monolithic counterparts. The cost gap between monolithic and multi-chiplet systems remains relatively stable or increases slightly due to the model's conservative bonding and packaging yield estimates. Nevertheless, the results

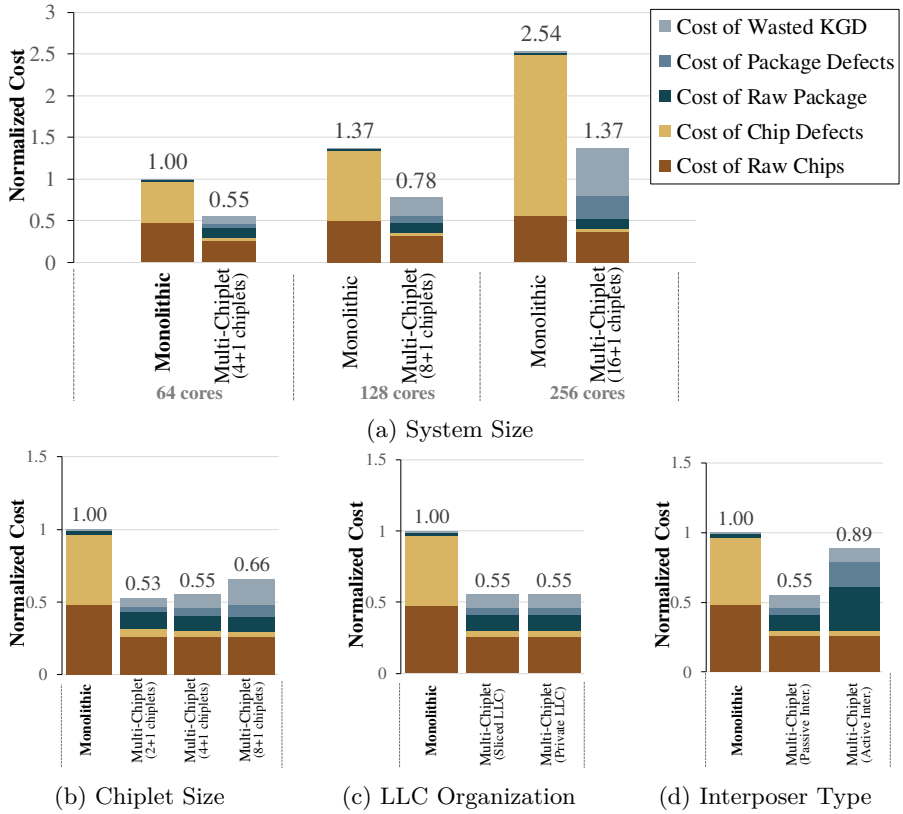


Figure 3.6: Cost analysis and comparison of RE costs between monolithic and various chiplet-based systems, with costs normalized to the monolithic.

highlight the substantial cost savings of chiplet-based designs over monolithic designs.

Chiplet Size: We examined the impact of chiplet size, which refers to the number of cores per chiplet, by partitioning a 64-core system into 2+1, 4+1, and 8+1 chiplet configurations, each including an IO chiplet. Figure 3.6b presents the cost values normalized to the monolithic design. The results show that while the first chiplet-based system has a reasonable cost, increasing the number of chiplets causes a non-linear rise in overall system cost. This can be attributed to the increased package defects and wasted KGD as chiplet size increases, reducing yield and raising costs.

LLC Organization: Another key design choice was the LLC organization in the chiplet-based systems. We evaluated a 64-core system partitioned into a 4+1 chiplet configuration. Figure 3.6c presents the cost values normalized to the monolithic design. While the LLC organization does not directly influence the cost, Section 3.4 demonstrates that a private LLC significantly outperforms a sliced LLC in terms of system performance.

Interposer Type: The last design choice was the different types of silicon interposers for multi-chiplet systems, evaluated on a 64-core system partitioned into a 4+1 chiplet configuration. Figure 3.6d presents the cost values normalized to the monolithic design. Unlike passive interposers, active interposers incorporate active logic to pipeline network links, thus improving the network's latency and throughput. While active interposers enhance system performance (as demonstrated in Section 3.4), they come at a 61% higher cost than passive interposers, making their cost comparable to monolithic designs.

Overall, the analysis highlights the cost trade-offs in chiplet-based designs and the significant cost savings they can achieve.

Paper III

PRISM: Reducing Performance Overhead of Multi-chiplet Chip Interconnects

Panagiotis Strikos, Ahsen Ejaz, and Ioannis Sourdis

Under Submission

Chapter 4

PRISM: Reducing Performance Overhead of Multi-chiplet Chip Interconnects

Abstract

Chiplet-based designs are a more cost-effective way to build large chips compared to monolithic dies, but introduce communication overheads. Crossing the chiplet boundaries has bandwidth limitations and latency overheads. More precisely, the microbump budget of a chiplet is tight as it is limited by its size and density allowed by the technology, while connecting via a silicon interposer, especially a passive one, adds extra delay. This paper introduces PRISM, a new network for multi-chiplet chips, which alleviates these overheads. It describes a new topology which offers reduced hop count for inter-chiplet traffic and is free of network deadlocks. In addition, it utilizes more effectively the chiplet-to-chiplet bandwidth by compressing flits selectively only when congestion is detected, thereby minimizing compression latency costs. PRISM is evaluated in a 256-core system with 16 chiplets improving inter-chiplet packet latency by up to 5.8× and overall system performance up to 17.4%.

4.1 Introduction

In an era where the demand for compute power increases rapidly, integrating more resources on a chip is essential for the performance scaling of processors. However, technology scaling becomes increasingly challenging. Moore's law has diminishing returns, and integrating more devices on a monolithic chip suffers excessive costs. Building chips out of multiple smaller chiplets is a cheaper, higher yield alternative [13, 21, 27, 35], which, in addition, is well-suited to the integration of heterogeneous components. For over a decade, die stacking technology has enabled disintegrating processors to multiple chiplets, e.g., AMD EPYC and RYZEN architectures [27]. Currently, large chips, such as the AMD MI300 [1] and Intel Sapphire Rapids [29], are composed of many CPU and/or GPU chiplets, as well as HBM nodes combining high processing throughput with fast, high-bandwidth memory access.

Despite their improved yield, multi-chiplet chips come with various performance bottlenecks [24]. Inter-chiplet interconnects suffer latency and bandwidth overheads compared to their monolithic counterparts. In particular, the number of available microbumps on a chiplet is limited by its size and its density constraints, putting a cap on available off-chiplet bandwidth. Furthermore, chiplet-to-chiplet (C2C) communication requires traversing longer links via microbumps and a silicon interposer, adding extra latency. Using a passive silicon interposer is often selected due to its lower cost, but makes the aforementioned latency overheads more severe. Finally, regular monolithic chip network topologies, such as 2D mesh, do not fit well to chiplet-based chips, so avoiding network deadlocks requires special attention.

Although many Network-on-Chip (NoC) topologies [3, 11, 12, 17, 21, 34] as well as deadlock avoidance mechanisms [4–6, 15, 23, 37, 38, 40–42, 44] have been proposed for multi-chiplet chips, most of them do not apply to systems with passive silicon interposers. The absence of active components on the silicon interposer makes it more challenging to design a scalable, efficient, and deadlock-free network solution. Some existing approaches demonstrated the benefits of fewer wider C2C links versus more and narrower links [14]. However, apart from that, very little of prior work has focused on the latency and bandwidth limitations of C2C communication.

This paper introduces PRISM, a new network solution for multi-chiplet chips. The design focuses on systems integrated on passive silicon interposer, although many of its concepts are applicable to systems with active interposers, too. A new network topology is introduced which minimizes inter-chiplet latency. It employs dedicated edge routers to handle inter-chiplet traffic and within a chiplet connects them with each other directly via peripheral bypass links. Thereby PRISM reduces hop count of inter-chiplet traffic that crosses a chiplet and prevents unnecessary interference with intra-chiplet traffic. Moreover, a new deadlock free, deterministic routing algorithm is created for the PRISM topology. Finally, edge routers are equipped with compression support to utilize more effectively the limited C2C bandwidth. Individual inter-chiplet flits are considered for compression and selected only when congestion is detected or when they are not critical for performance, e.g., write-back traffic, thereby avoiding unnecessary compression delays.

In summary, PRISM makes the following contributions:

- A new topology for multi-chiplet chips with passive silicon interposer is introduced, which improves packet latency by reducing hop-count of inter-chiplet traffic and by avoiding interference with intra-chiplet traffic.
- A deterministic deadlock-free routing algorithm for the above topology is developed.
- PRISM is the first to apply compression for C2C communication on multi-chiplet chips. Its edge routers are designed to selectively compress inter-chiplet traffic based on their latency criticality and router congestion thereby increasing effective C2C bandwidth with reduced latency overheads.
- PRISM is evaluated on a system with 16 chiplets and 256 cores in total demonstrating up to 5.8× reduction in inter-chiplet packet latency and up to 17.4% improvement in system performance.

The remainder of this paper is organized as follows: Section 4.2 discusses related work. Section 4.3 presents the PRISM design, covering the network topology, the XY-Bypass routing algorithm with deadlock avoidance, and the compression-capable edge router. Section 4.4 describes the experimental methodology. Section 4.5 provides our evaluation results, and Section 4.6 summarizes our conclusions.

4.2 Related Work

This section presents existing network topologies and network deadlock prevention mechanisms for multi-chiplet chips, as well as previous compression techniques for NoCs.

4.2.0.1 Network Topologies for multi-chiplet chips

Several works have explored topologies for multi-chiplet systems considering active interposers [3, 17, 21, 34], which can embed repeaters and routing logic but are significantly more expensive than their passive counterparts [35]. On the contrary, this work focuses on passive interposers, which make network design more challenging.

Feng et al. [14] showed that grouping the edge interfaces of 2D-mesh chiplets enables high-radix inter-chiplet topologies, which reduce network diameter and deliver up to 2× saturation throughput and 45% lower latency than a conventional 2D-mesh multi-chiplet network. PRISM follows a similar approach using fewer C2C links per chiplet edge than the dimension of its internal 2D-mesh. The same authors later proposed Ring Road [12], an intra-chiplet NoC to replace router-to-router channels with router-less rings, achieving 1.7–2× bandwidth-per-area over 2D-mesh and directing cross-chip traffic onto dedicated outermost ring channels so that transit packets do not degrade local NoC performance.

Targeting passive interposers, Network-on-Die (NoD) [11] moves all interchiplet routers off the compute chiplets onto a standalone routing chiplet placed at the centre of the interposer. This system is conceptually similar to

one with a dedicated I/O die such as the AMD EPYC [27], although NoD is a pure routing die rather than a combined I/O and memory hub. However, interconnecting all chiplets through a single central routing die does not scale well with the number of chiplets in a system as it requires longer C2C wires and more complex routing die. As a consequence, NoD's single-cycle C2C link assumption cannot hold for larger chiplet arrays, especially on a passive silicon interposer. Coskun et al. [7] addressed this by pipelining the links through intermediate chiplets. In contrast, PRISM topology allows long C2C connections, after exiting the source chiplet and before entering the destination chiplet, to use exclusively edge routers and their bypass links. In doing so, hop count is reduced and interference with intra-chiplet traffic is avoided in intermediate chiplets.

4.2.0.2 Network deadlock avoidance in multi-chiplet chips

A common deadlock avoidance strategy is turn restriction: forbidden turns at boundary routers break channel dependency cycles [41], with subsequent work automating restriction selection [4–6] or adding escape paths for blocked packets [40]. Remote Control [23] controls when packets may leave their chiplet rather than which turns they may take. Both strategies reduce throughput and hurt performance. More precisely, forbidden turns decrease path diversity, while injection gating throttles traffic when the network is congested.

Zhang et al. [44] propose deadlock-free routing for chiplets on organic substrates, which have quite different characteristics and wiring constraints than silicon interposers [16]. This algorithm is therefore not directly applicable to chips based on silicon interposers.

An alternative class of approaches avoids deadlock through *virtual-channel* (VC) or *virtual-network* (VN) separation. Holsmark et al. [15] show that hierarchical NoC topologies can be deadlock-free if different VC classes are assigned at hierarchy boundaries. DeFT and ReD propose VN separation for 2.5-D chiplet systems [37, 38]. Two VNs and three simple routing rules suffice to prevent dependency cycles while preserving full path diversity. Yu et al. [42] propose another VN-based scheme where VN0 is exclusively reserved for inbound and intra-chiplet packets while any packet may use VN1; cyclic channel dependencies are permitted to exist in VN1, and deadlock is avoided because inbound packets can always escape via VN0. However, all these schemes target *active* interposers, where on-interposer routers participate in VN assignment and enforce transition rules. PRISM adjusts the VN-separation concept of ReD [38] to fit the passive interposer constraints. Since the interposer cannot host any routing logic, additional (edge) routers are placed at the chiplet boundaries and are employed to handle C2C communication and VN enforcements to guarantee deadlock avoidance.

4.2.0.3 Compression mechanisms for NoCs

Das et al. [9] were among the first to study in-network compression, showing that compressing cache traffic at the network interface (NI) reduces average network latency and power. Their design shows a maximum speedup of around 6%, even at an optimistic 4 GHz NoC clock.

Several designs improve NoC bandwidth by compressing all network traffic, but add latency overheads even at low network load. NoΔ [43] applies BDI compression per packet at every NI using parallel compressors with single-cycle latency, but does not measure gains in system performance. Niwa et al. [30] place a compressor at each router input and perform compression in parallel with switch allocation. FlitZip [10] compresses injected traffic at flit granularity using BDI with a 1-byte base. Multiple compressed flits are packed together and metadata is stored in the header flit.

Selective compression avoids penalising uncongested traffic. Jin et al. [20] track recurring values per flow in a shared table and compress only packets on congested paths, but the decision is made once at the source NI. Packets that skip compression have no further opportunity to be compressed in the network. SCNoCs [18] predicts compression ratio using a per-NI history table and propagates a global congestion threshold to all routers via a H-tree broadcast through a separate dedicated subnetwork. DISCO [39] makes compression decisions per packet and overlaps (de)compression with queuing of packets. It shares one compressor per router forcing traffic to be serialized. It also requires double buffering and extra hardware for zero-bubble padding when compressed flits are separated to be compatible with wormhole routing.

All these approaches target monolithic chips and place compressors at every router or NI. PRISM targets multi-chiplet chips and applies selective compression only at the C2C links which are a bottleneck. In doing so, PRISM maximizes the benefit-to-area-overhead ratio while avoiding latency costs on uncongested traffic. Unlike other works, a flit that skips compression at the source chiplet still has the opportunity to be compressed at subsequent intermediate chiplets.

4.3 Design

PRISM is an interconnection network for multi-chiplet chips integrated on a passive silicon interposer designed to reduce latency and bandwidth overheads of chiplet-to-chiplet (C2C) communication. As illustrated in Figure 4.1, chiplets are organized in a 2D mesh and interconnected with each other through bidirectional C2C links whose width is determined by the available microbump budget. Specialized edge routers at each chiplet boundary bridge the intra- and inter-chiplet networks. On-chiplet bypass links connect edge routers within a chiplet, allowing inter-chiplet traffic to cross intermediate chiplets without entering their internal network reducing hop count and avoiding interference with intra-chiplet traffic. Furthermore, edge routers support packet compression to increase effective C2C bandwidth utilization. The following subsections describe the PRISM topology, routing algorithm, and edge router design.

4.3.1 Network Topology

The chiplets are arranged in a 2D mesh and interconnected through a passive interposer, as shown on the left side of Figure 4.1. Each chiplet is connected to its four neighbors (except those on the edges of the overall mesh) through two bidirectional chiplet-to-chiplet (C2C) links. Each chiplet contains its own internal 2D mesh network, as depicted on the right side of Figure 4.1. Without

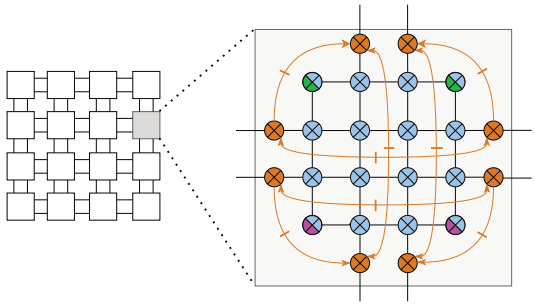


Figure 4.1: PRISM topology. At the chiplet-mesh level (left), chiplets are arranged in a mesh and connected to their neighbors through two bidirectional C2C links per side. Within each chiplet (right), an internal 2D mesh is augmented with edge routers on each side, connected to one another through bypass links. Routers are color-coded by the components they serve: cores (blue), LLC banks (magenta), and HBM controllers (green).

loss of generality, each non-edge router interconnects a core, while the corner routers utilize unused ports to also connect to HBM and LLC nodes. In the perimeter of each chiplet there are two edge routers on each side, which are directly connected to the corresponding edge routers of the neighboring chiplet through C2C links. Each edge router is also connected to the internal mesh of its chiplet through what we denote as a local link, and to one or two other edge routers of the chiplet through pipelined bypass links. The purpose of these bypass links is to allow packets whose destination lies beyond the current chiplet to travel directly from one edge router to the other, without entering the internal network of a chiplet that is not their destination. It should be noted that not all bypass links are bidirectional; this is determined by the constraints of the inter-chiplet XY routing scheme described next.

4.3.2 XY-Bypass Routing and Deadlock Avoidance

Even when each chiplet's internal routing is individually deadlock free, combining intra- and inter-chiplet networks can introduce circular dependencies in the global channel dependency graph, leading to deadlock [38, 41]. Figure 4.2 illustrates such a scenario: each chiplet uses XY routing, so intra-chiplet routing is deadlock free, yet a circular buffer dependency when crossing multiple chiplets would cause the system to deadlock in the absence of VN separation.

Deadlocks in such cases may arise from the fact that YX turns are formed by routing in two network levels (intra- and inter-chiplet), each having its own (XY) routing. Simply stated, a packet uses XY within the source or destination chiplet, but may cause a YX turn in between. In the example, the purple packet traverses the network in the Y direction (North to South) reaching its destination router. There, in order to reach its destination chiplet, it uses XY routing again, but the fact that its destination is on the West makes it take a turn from North to West. This turn is forbidden by traditional XY routing, but necessary in a hierarchical version of the algorithm. The introduction of

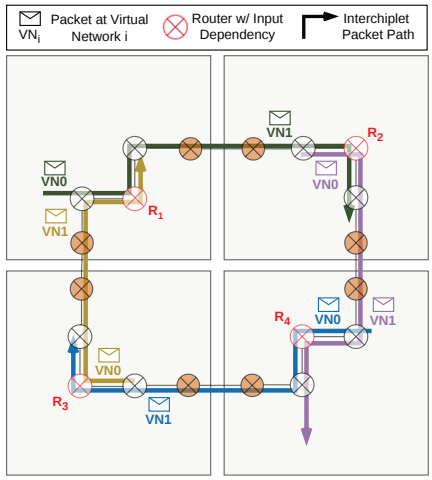


Figure 4.2: Potential deadlock in a multi-chiplet network using XY routing without VN separation. Flits from four inter-chiplet packets occupy buffers in the inputs of routers R_1–R_4, forming a circular dependency. VN separation breaks the cycle: on any chiplet that is the source for one packet and the destination for another, those packets are assigned to different virtual networks and cannot occupy the same bufferspace, eliminating the circular dependency.

multiple VNs and the rules described next break these circular dependencies.

PRISM adapts an existing two-virtual-network (VN) deadlock-avoidance scheme from ReD [38], originally developed for 2.5D chiplet-on-active-interposer networks, where deadlocks arise from the interaction between on-chip traffic and the vertical links connecting each chip layer to the interposer routing layer. That model is mapped to our topology as follows:

- The intra-chiplet mesh corresponds to the on-chiplet routing layer.
- Edge routers, though physically on the chiplet die, correspond to on-interposer routers.
- Bypass links and C2C links correspond to the interposer vertical connections.
- Local links between an edge router and its adjacent intra-chiplet router correspond to a vertical chiplet-to-interposer link.

The deadlock-freedom guarantee of ReD therefore still holds in our version. Employing two virtual networks (VN0 and VN1) and the following three rules suffice to break any cyclic dependencies:

[I] Transition from VN1 to VN0 is forbidden. Only transitions from VN0 to VN1 are allowed.

[II] Inter-chiplet packets should only use VN0 before exiting the source chiplet's edge router.

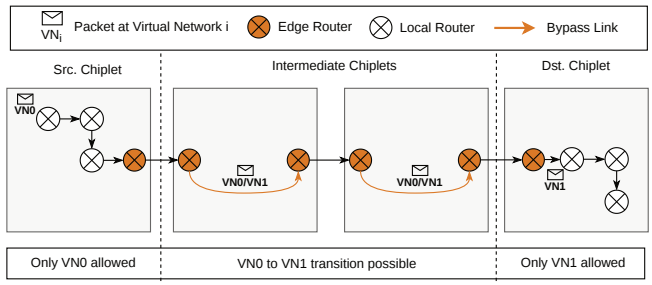


Figure 4.3: XY-Bypass routing example. A packet is first routed within its source chiplet in VN0 to the appropriate edge router. In intermediate chiplets, bypass links (orange arcs) carry the packet from the incoming to the outgoing edge router without entering the internal mesh; the VN0→VN1 transition may occur at any point during this inter-chiplet journey. Upon entering the destination chiplet, the packet is already in VN1 and follows XY routing to its final destination.

[III] Inter-chiplet packets should only use VN1 after entering the destination chiplet's internal mesh.

Rules 2 and 3 together enforce that a packet must be in VN0 while routing within its source chiplet, and must be in VN1 by the time it enters its destination chiplet's internal mesh. The transition may happen at any point in between: at the source edge router, on a bypassing or C2C link, or at the destination edge router. Combined with Rule 1, each inter-chiplet packet crosses the VN boundary exactly once, preventing any cyclic dependency between intra- and inter-chiplet traffic. Each virtual network is allocated a minimum of two virtual channels, which is the smallest number that prevents head-of-line blocking within a VN.

Packets are routed using XY-Bypass routing: a two-level scheme that applies XY dimension-order routing both within each chiplet and across the chiplet mesh. Bypass links at edge routers enable packets to skip intermediate chiplets that are not their destination. Figure 4.3 illustrates an example inter-chiplet packet traversal.

Intra-chiplet packets follow standard XY routing within the chiplet mesh and are free to use either VN, as they never traverse an inter-chiplet port, but still have to adhere to rule 1.

Inter-chiplet packets are routed in three stages. In the first stage, the packet is routed in the source chiplet in VN0 towards the appropriate edge router using XY routing. The edge router is selected based on whether the packet's path at the chiplet-mesh level requires a turn. If the packet travels straight, moving along only one axis at the chiplet level, the closest edge router on the corresponding side is selected. If the packet must turn at the chiplet-mesh level, the edge router is selected based on the downstream bypass link. Since each edge router's bypass links connect to fixed directions, only the edge router that has a bypass link in the required turn direction can route the packet correctly. In the second stage while the packet is traversing through intermediate chiplets,

it uses bypass links to pass from the incoming edge router to the outgoing edge router without entering each chiplet's internal mesh. During this inter-chiplet journey, the packet may transition from VN0 to VN1 at any point. Finally, in the third stage, the packet enters the internal mesh of the destination chiplet in VN1 and follows XY routing to its final destination.

Having established a deadlock-free multi-chiplet network, the following section introduces compression at the edge routers to improve latency and throughput on the inter-chiplet links.

4.3.3 Edge Router

Without loss of generality, all routers in the PRISM network are 3-stage routers [8] with next route computation, speculative switch allocation and split switch traversal (ST) and link traversal (LT), introducing in total 3 cycles delay per hop. Edge routers handle inter-chiplet traffic and are additionally enhanced with flit compression support. This adds another pipeline stage to the edge routers, increasing hop latency according to the compression latency. As discussed next, two variants of edge routers were designed and studied: one that compresses all inter-chiplet traffic (Always-Compress, AC), and one that dynamically selects which flits to compress (Selective-Compress, SC). It should be noted that only multi-flit packets are eligible for compression, and single-flit packets are always forwarded uncompressed.

In edge routers, incoming flits are intercepted before reaching the input buffers and fed into the compression unit. The head flit is always forwarded uncompressed. Base-Delta Immediate (BDI) compression [31] is applied to each body flit. We use a modified BDI variant that adds 1-byte and 2-byte repeat encodings beyond the standard 8-byte repeat. These additions increase compression density at the flit granularity. Additionally, we omit the three base-8-delta variants (B8D1, B8D2, B8D4). The base-8 encodings offer limited to no compression benefit at the 16-byte flit granularity, as the 8-byte base alone consumes half the flit before any deltas. Moreover, restricting the scheme to eight total encodings allows the encoding type to be stored in only 3 bits within each flit. Compression begins as soon as the first body flit arrives without waiting for the full packet. Each flit is encoded with its own base and delta values which are carried within the flit so the decompressor can reconstruct every flit independently. Body flits are repacketized as they exit the compressor: each input flit contributes a fraction equal to the compression ratio toward a full-width output flit, and a new flit is emitted once enough compressed data has accumulated to fill one. If the last body flit exits the compressor without the accumulated data reaching a full flit, the final output flit is padded. The head flit is held at the edge router until the first body flit's compression result is known, since the outcome must be recorded in the header so the destination can determine whether to decompress. If the compressed body data fits within the head flit's unused payload bytes, it is packed there rather than emitted as a separate flit. The head flit continues absorbing subsequent compressed body data as long as space remains. In the most favorable case where each flit contains a single repeated byte, the entire packet collapses to a single flit. The compressed flits are placed back into their original virtual channels, after which they proceed through the router's pipeline.

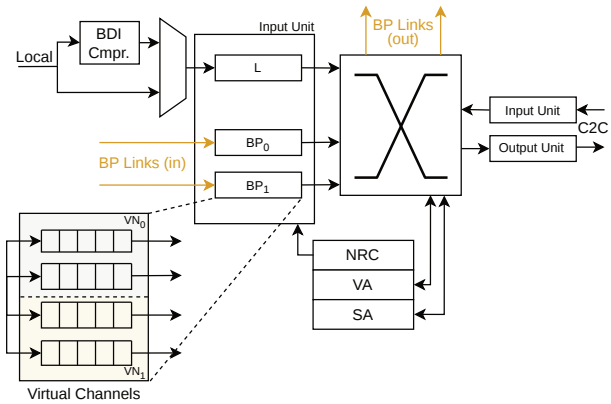


Figure 4.4: Always-compress edge router datapath. A single BDI compressor is placed on the local input compressing all incoming flits. Bypass links connect to other on-chiplet edge routers. C2C inputs carry traffic arriving from neighboring chiplets. Bypassing traffic is already compressed and does not need to go through the compressor again.

Decompression is applied at the network interface (NI) of the destination router. Beyond the first edge router it enters, an inter-chiplet flit travels compressed through its entire path, including any bypass links and the internal mesh of the destination chiplet, and is decompressed by the NI only when it reaches its final destination. Since any router in the system may receive a compressed inter-chiplet flit, every router's NI is equipped with a BDI decompressor.

The two compression modes described below share this datapath. They differ in where compressors are placed and in whether compression can be dynamically decided per flit.

4.3.3.1 Always-Compress (AC) edge router

In the AC edge router, a single BDI compressor is placed on the local input of the edge router, as shown in Figure 4.4. Every eligible flit originating within the chiplet is compressed unconditionally before being placed onto the inter-chiplet link. Bypassing packets (packets transiting through the chiplet on their way to a further destination) enter and leave through the bypass links and are never seen by the compressor, passing through the router unmodified. The bypassing and C2C inputs carry no compressor; only locally-originating traffic is eligible for compression in AC.

AC delivers the highest possible effective throughput on the inter-chiplet links and is bound to improve latency under high congestion. However, compression latency is always incurred, even on an otherwise idle link and this can increase end-to-end latency under light load.

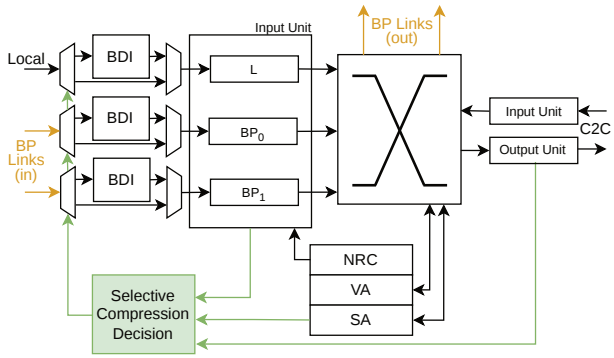


Figure 4.5: Selective-compression edge router datapath. BDI compressors are placed only on the local input and bypass inputs (two or three compressors per edge router). Bypass links connect to other on-chiplet edge routers. C2C inputs carry traffic arriving from a neighboring chiplet. Already compressed flits arriving in bypass inputs skip the compressor. The rest are evaluated by the selective compression mechanism.

4.3.3.2 Selectively-Compress (SC) edge router

Compressing a flit when the output C2C link is idle increases latency because it has to go through the compressor and packetizer, instead of directly using the free C2C link. SC avoids this by enabling compression only when the inter-chiplet link is congested. This hides the compression latency by overlapping it with the queuing delay a flit will encounter in the edge router.

SC places one BDI compressor on all non-C2C inputs: the local input and the one or two bypassing inputs, as shown in Figure 4.5. C2C inputs carry traffic arriving from a neighboring chiplet and have no compressor. When the head flit of an eligible packet arrives at an edge router, the router evaluates four congestion indicators before deciding whether to compress:

[I] **Buffer Occupancy (BO)**: the total number of flits waiting across all local and bypass input buffers meets or exceeds a threshold t_{BO}, indicating that one of them might use the link soon and therefore there is enough time to compress the incoming flit.

[II] **Link Busy (LB)**: the C2C link was used recently and has not yet been idle for long enough to be considered free.

[III] **Compressor Busy (CB)**: at least one compressor pipeline across any input currently has a flit in-flight, meaning it will occupy the C2C link imminently.

[IV] **Output Stall (OS)**: all downstream VC buffers are full or no virtual channel is available on the C2C output port, so the packet cannot depart uncompressed regardless.

If any condition holds, the packet is compressed; otherwise it bypasses the compressor and is forwarded immediately. Under low load, when the link is

free, buffers are empty, and the compressor is idle, packets skip compression entirely. Under high load, at least one of the conditions is met and compression reduces the number of flits competing for the bottleneck link.

Placing compressors on the bypassing inputs has an additional benefit for multi-hop traffic. A packet in transit that was chosen by the selective mechanism to skip compression at the source chiplet because the link was idle at the time will have another opportunity to be compressed at a subsequent edge router if the conditions indicating congestion are met. SC can therefore compress a packet at any edge router along its path, not only at the source chiplet.

4.4 Experimental Methodology

4.4.1 System Configuration

We evaluate a 256-core multi-chiplet system comprising 16 chiplets arranged in a 4×4 mesh on a passive silicon interposer. Each chiplet contains 16 out-of-order cores and is connected directly to a local HBM2 node; all HBM2 nodes are part of a global flat address space. Memory is allocated using a chiplet-local-first policy: pages are mapped to the HBM node on the chiplet of the requesting core and fall back to a remote HBM node only when the local one is full. Table 4.1 summarizes the full system configuration.

The inter-chiplet link width is determined by the microbump budget available per chiplet. The number of microbumps is proportional to the chiplet area, which in turn depends on the number of cores, cache capacity, and manufacturing process node. Assuming a microbump pitch of 45 µm and reserving 40% of bumps for power delivery, the remaining signal bumps are allocated to the HBM2 channels, and to the inter-chiplet links. For a 16-core chiplet at 5 nm, this budget yields 128-bit bidirectional inter-chiplet links [24].

A chip-to-chip link latency of 2 cycles is considered, derived from the physical dimensions of the chiplets and the propagation delay of links on a passive silicon interposer [24, 35]. The analysis in [24] reports a range of 2–3 NoC clock cycles depending on the maximum link length; we adopt the lower bound, which corresponds to the shorter inter-chiplet distances in our 4×4 16-chiplet layout.

4.4.1.1 Evaluated Designs

All evaluated systems share the same multi-chiplet topology: a mesh of chiplets interconnected through edge routers with bypass links, using two virtual networks for deadlock avoidance as described in Section 4.3. The only dimension that varies across systems is the compression policy applied at the edge routers:

- **Baseline (BL):** Edge routers forward inter-chiplet packets without compression. This serves as the reference point for evaluating the impact of compression on latency and throughput.

- **Always Compress (AC):** BDI compression is applied to every inter-chiplet packet at the source edge router and decompressed at the desti-

Table 4.1: System Configuration

System	
Chiplets	16 chiplets, 4×4 inter-chiplet mesh
Cores and Caches	
Cores	256 total (16/chiplet), out-of-order, 3.2 GHz
TLB	I-TLB: 512-entry, 4-way, 1 cycle latency D-TLB: 512-entry, 4-way, 1 cycle latency
L1 Cache	L1-I: Private, 32KB, 4-way, 3 cycle access latency L1-D: Private, 32KB, 4-way, 3 cycle access latency
L2 Cache	Private, 256KB, 8-way, 4 cycle access latency
L3 Cache	Private per chiplet, 2 MB/core, 16-way, 25 cycle access latency
Main Memory	
HBM2	4 GB/chiplet, 2 GHz, 8 channels, 128 bits per channel, tCAS-tRCD-tRP: 14-14-14 ns
Network	
Intra-chiplet	2 GHz, 3-stage router (VA/SA, ST, LT), 4×4 core mesh with 8 dedicated edge routers (2/side) and bypass links, 2 VNs, 2 VCs per VN, credit-based flow control, 128-bit link, 5 flit buffers, XY-bypass Routing
Inter-chiplet	2 GHz, 3-stage router (VA/SA, ST, LT), 4×4 chiplet mesh, 2 cycles C2C link latency, 128-bit link, 9 flit buffers [1]
Edge Compr.	BDI at edge routers; 1 compressor/router (AC: local input) or 3 compressors/router (SC: local + 2 bypass inputs); 1-cycle compression [43], 2-cycle packetization, 1-cycle decompression
Packaging	
Interposer	Passive silicon interposer

[1] Increased only for edge routers to accommodate round-trip delay of the C2C link.

nation network interface. This maximizes bandwidth savings but incurs compression overhead on every packet regardless of network load.

- **Selective Compress (SC):** Under low load, packets are selected by the mechanism in place to be forwarded uncompressed; BDI compression is activated only where it can reduce queuing delay.

4.4.2 Simulation Setup

4.4.2.1 Standalone Network Simulations

Standalone Network evaluation uses BookSim2 with synthetic traffic. The detailed parameters of the simulated network can be found in Table 4.1.

For network modeling, BookSim2 [19] was extended to support our multi-chiplet NoC architecture. The network model was augmented with the chiplet mesh topology, edge routers, and bypass links. The XY-bypass routing function and the virtual network assignment logic enforcing the three deadlock-avoidance rules were implemented directly in the routing layer. To evaluate NoC compression, we added a compression router model at the edge routers implementing the three configurations studied in this work: BL, AC, and SC.

Synthetic traffic consists of equal fractions of 5-flit data packets and 1-flit control packets. We evaluate three traffic patterns: uniform random (UR), hybrid uniform random (Hybrid UR), and transpose. Hybrid UR is a variant of UR that biases traffic toward intra-chiplet destinations to alleviate some of the inter-chiplet congestion. In particular, 80% of Hybrid UR traffic is intra-chiplet, with destinations selected using UR within the source chiplet, and the remaining 20% of the traffic is inter-chiplet, with destinations selected using UR across all other chiplets. This is based on the average observed C2C traffic in our system-level experiments which is about 20% of the total traffic.

4.4.2.2 System-level Simulations

System-level experiments are conducted using BZSim [36], a microarchitectural simulator that couples ZSim [33] with the extended BookSim2 network model for cycle-accurate intra- and inter-chiplet network simulation, and applies a low-contention traffic skipping technique to accelerate simulation without sacrificing accuracy. The simulator delivers detailed interconnect modeling at speeds an order of magnitude faster than GEM5, enabling the simulation of multi-billion instruction experiments within reasonable runtimes. HBM timing is modeled with DRAMSim3 [22] and cache access latencies are estimated using CACTI [26].

To obtain per-packet compression ratios during system-level simulation, we extend BZSim with a compressibility instrumentation layer. For each cache line traversing the hierarchy, BDI is applied to the line's actual data read from the application's virtual address space via Pin. The resulting compression ratio is propagated with the request through the cache levels and, upon entering the NoC, is stamped onto every flit of the outgoing packet. Edge routers use this ratio to determine the compressed byte count for each inter-chiplet flit.

We select nine workloads from four suites: SPEC OMP2012 [25], GraphBIG [28], GAP [2], and Splash-3 [32], covering irregular graph analytics and structured scientific codes. The primary selection criterion is *remotely served LLC MPKI* (rMPKI): the number of LLC misses per kilo instructions that must be satisfied by a remote chiplet's HBM, crossing at least one inter-chiplet link. A high rMPKI stresses the network and allows us to observe the impact of compression. Workload inputs, LLC MPKI, and rMPKI are summarized in Table 4.2. All benchmarks are multi-threaded and one host thread is mapped to one of the 256 simulated cores. The benchmarks are also instrumented to simulate only their parallel region and run for 2 billion instructions.

4.5 Evaluation Results

4.5.1 Standalone Network Performance

For the evaluation of the network using synthetic traffic, a configurable compression ratio was used. Given that we simulate 5-flit packets, we select ratios of 2.5×, 1.67×, and 1.25× that correspond to a 5-flit packet being compressed to 2, 3, or 4 flits, respectively. These ratios represent high, medium, and low compressibility. Figure 4.6 presents the average packet latency as a function of injection rate for the three traffic patterns and compression ratios.

Table 4.2: Workload Characteristics

Benchmark	Input	LLC MPKI	LLC rMPKI
graphColoring[2]	LDBC 1000k	100	69
dbc[1]	Default	68	63
fmm[4]	n=65K, p=256	71	47
fma3d[1]	Default	135	34
pr[3]	Scale 25, 10 trials	50	31
gbig-bfs[2]	CA Road Network	23	21
sssp[3]	Scale 25, 10 trials	27	18
radix[4]	n=262K, r=1K, p=256	22	16
gap-bfs[3]	Scale 25, 10 trials	25	15

[1] SPEC OMP2012 [25], [2] GraphBIG [28], [3] GAP Benchmark Suite [2], [4] Splash-3 [32]

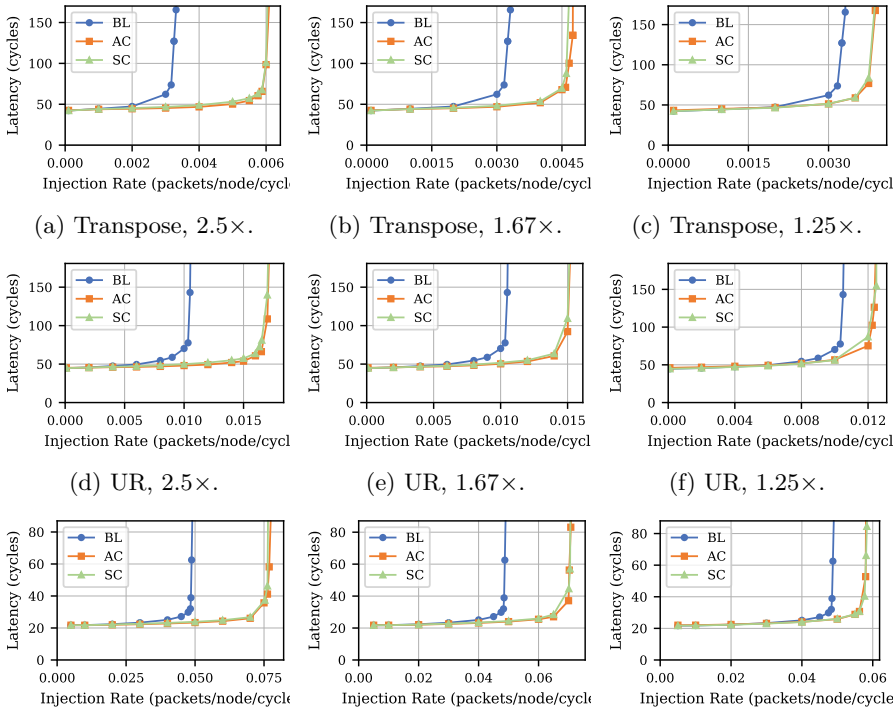
At zero load, AC incurs a small latency overhead compared to the baseline: less than 1% for the high and medium compression ratios, and approximately 2% for the low compression ratio. This overhead stems from the compression, packetization, and decompression delays applied at the edge router even on an uncongested network where compression cannot be hidden behind queuing delays. It is expected that these overheads would be higher for slower compression algorithms. On the other hand, SC matches baseline zero-load latency. In the absence of congestion, SC allows packets to skip compression at the edge router and to proceed directly to the chip-to-chip link.

Compression yields significant throughput gains across all traffic patterns and compression ratios. For transpose traffic, AC and SC achieve 1.2×, 1.5×, and 1.9× higher peak throughput at compression ratios of 1.25×, 1.67×, and 2.5×, respectively. Uniform random and Hybrid Uniform Random show the same throughput benefits, in particular, they are 1.2× and 1.5× faster at low and medium compressibility, and 1.7× at high compressibility. The larger gain under transpose is due to the heavier inter-chiplet traffic load placed in specific links. This increases the performance impact of PRISM. In all cases SC matches the peak throughput of AC. This confirms that selectively skipping compression on lightly loaded links does not sacrifice network throughput.

4.5.2 System Performance

Figure 4.7a shows the speedup of a PRISM AC and SC system over a baseline with the same topology, but no C2C compression. AC achieves 8.1% higher performance on average, ranging from 2% (gbig-bfs) to 13.7% (fma3d). SC improves system performance by 10.0% on average, ranging from 2.7% (gbig-bfs) to 17.4% (fmm). SC outperforms AC on eight of the nine workloads used.

For AC, the achieved speedup depends primarily on three workload-specific factors. First is the percentage of packets that cross chiplet boundaries, which ranges from 11% to 33% and on average 18% as shown in Figure 4.7b. Second, the congestion these packets cause to the inter-chiplet links in the baseline



(a) Transpose, 2.5×. (b) Transpose, 1.67×. (c) Transpose, 1.25×.

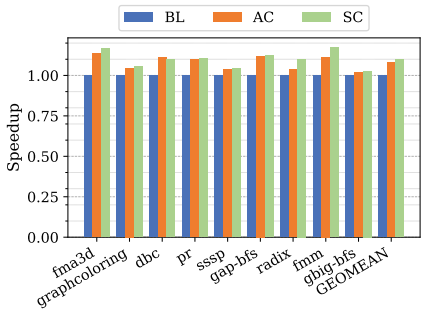
(d) UR, 2.5×. (e) UR, 1.67×. (f) UR, 1.25×.

(g) Hybrid UR, 2.5×. (h) Hybrid UR, 1.67×. (i) Hybrid UR, 1.25×.

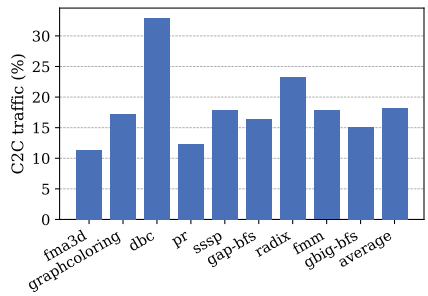
Figure 4.6: Average packet latency vs. injection rate for transpose (top), UR (middle), and Hybrid UR (bottom) traffic under high (2.5×), medium (1.67×), and low (1.25×) compression ratios (left to right).

system. This is reflected in the average packet latency of inter-chiplet traffic depicted in Figure 4.8b and ranges from approximately 85 to 620 cycles. It can be observed that most benchmarks with high inter-chiplet packet latency also have very high intra-chiplet packet latency as shown in Figure 4.8a. Third factor is the compressibility of that inter-chiplet traffic shown in Figure 4.7c, which ranges from 1.12× to 4.89× and is measured only on flits that go through the compressor. SC depends on the same above factors as well as its effectiveness to enable C2C compression only when needed. Figure 4.7d shows the percentage of flits that are not selected by SC for compression out of the total number of flits eligible for compression, which is from 18% to 44%. Note that a flit that skips compression at one edge router may be reconsidered for compression at a subsequent bypass link. This percentage indicates that the energy savings come without performance penalties and, in most cases, actually improve performance.

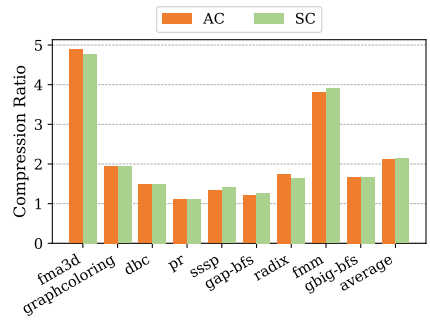
The two highest speedups under both schemes are observed in `fma3d` and `fmm`, which combine the highest compression ratios in the dataset (4.89× and 3.80× respectively) with saturated baseline inter-chiplet links, i.e., 489 and 487 cycles baseline inter-chiplet APL. The high ratios dissolve the congestion almost entirely under AC: inter-chiplet APL for `fma3d` falls by 81%, and for `fmm` by 78%, yielding 13.7% and 11.5% performance gains, respectively. It is worth



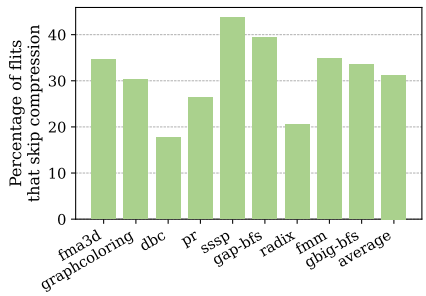
(a) Speedup of AC and SC over the baseline.



(b) Inter-chiplet traffic as a percentage of total number of flits.



(c) Effective compression ratio on eligible inter-chiplet flits.

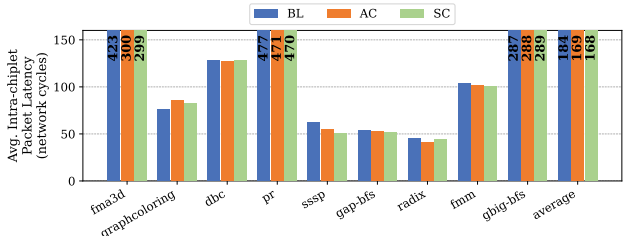


(d) Percentage of flits that skip compression at an SC edge router out of the total flits considered.

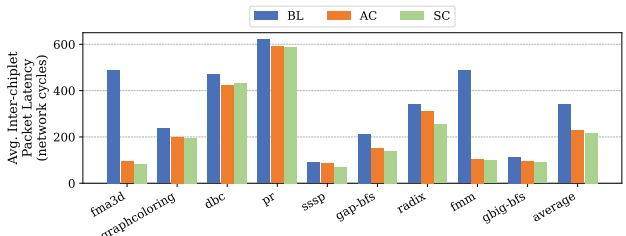
Figure 4.7: System performance, inter-chiplet traffic percentage, compression ratio, and percentage of flits that skip compression under SC across all workloads.

noting that `fma3d` exhibits high instruction imbalance, which is the relative variation in instructions retired across chiplets. This creates imbalanced load across different paths. In SC, the less congested paths benefit from skipping compression which is about 35% of the total eligible flits, while the more congested links maintain the advantages of compression utilizing better C2C bandwidth and improving inter-chiplet APL by an additional 10% versus AC, raising speedup to 17% compared to the baseline. `fmm` similarly benefits from selective compression because 35% of its flits eligible for compression skip compression lowering inter-chiplet APL by 6% increasing performance over baseline from 11.5% to 17.4%.

AC offers 10–12% speedup in `dbc`, `pr`, and `gap-bfs` with compression ratios of (1.12–1.48×). `dbc` generates the highest percentage of C2C traffic out of all benchmarks (32.9%). Under these circumstances, even the modest flit compression yields significant bandwidth savings lowering inter-chiplet APL by 10%. SC's skip threshold proves to be too lenient on these near-saturated C2C links: 18% of eligible flits skip compression in SC allowing uncompressed traffic to consume bandwidth that AC would save. This raises inter-chiplet



(a) Average packet latency for intra-chiplet packets.



(b) Average packet latency for inter-chiplet packets.

Figure 4.8: Average intra- and inter-chiplet packet latency across all workloads.

APL by 2% compared to AC (8% lower than baseline) offering 10% speedup over baseline versus 11% for AC.

`pr` has a high number of rMPKI and very congested C2C links (inter-chiplet APL is 621 cycles the highest in the dataset) exhibiting similar characteristics with `fma3d`. However, its lower compression ratio and more balanced workload does not allow the same performance gains. Still AC delivers 10% speedup and SC 10.5% speedup allowing 26% of eligible flits to skip compression with negligible impact in the latency of packets crossing the already congested C2C links.

`gap-bfs` has an rMPKI of 15, and a compression ratio of 1.20×, lower than other benchmarks with similar speedups, but even moderate compression alleviates the already congested C2C links, reducing APL by 30% and offering 12% speedup. In SC, 40% of the eligible flits skip compression, and the compressed ones have slightly improved compression ratio of 1.27×. This however does not have significant impact on inter-chiplet APL, and combined with the fact that intra-chiplet APL is already low, the SC speedup is 12.5% compared to the baseline, only 0.3% higher than AC.

The remaining workloads, `sssp`, `radix`, and `gbig-bfs`, deliver the lowest speedup of 2.0–3.9% in AC for different reasons. `sssp` has a low percentage of remotely served LLC misses so the inter-chiplet links are not heavily loaded in the baseline design (89 cycles APL). Still its moderate compression ratio of 1.33 reduces inter-chiplet APL by 5% resulting in a 3.9% speedup. In SC, 43% of the flits avoid going through the compressor and the associated latency overheads. This increases speedup for SC (4.3% vs. baseline) compared to AC (3.9% vs. baseline).

`radix` has the second-highest percentage of C2C traffic (23.3%) and 1.74× compression ratio. Given its moderate workload imbalance and the fact that more than half of its inter-chiplet traffic consists of non-compressible single-flit

control packets, AC's gains are limited to 3.8%. SC allows 20% of the flits to skip compression reducing compression costs in less congested paths, which yields 10% speedup over baseline. This is the largest improvement of SC over AC among all benchmarks.

Finally, `graphColoring` has rMPKI of 69, the highest one measured. It has a moderate 2× compression ratio. However, most of the traffic crossing C2C links are single flit packets. AC alleviates some of the congestion in C2C links and achieves a 15% inter-chiplet APL reduction which offers a 4.6% speedup. In SC, 30% of the flits skip compression, but the fact that the inter-chiplet traffic is only 17% of the total traffic allows only an improvement of 2% in inter-chiplet APL and a speedup of 5.7% compared to the baseline.

4.6 Conclusions

This paper described PRISM, a new network for multi-chiplet chips built on a passive silicon interposer designed to reduce C2C latency and bandwidth overheads. PRISM introduced a new topology which lowers hop count and latency of inter-chiplet traffic while avoiding network deadlocks. In addition, it utilizes more effectively the chiplet-to-chiplet bandwidth by compressing flits selectively only when congestion is detected, thereby minimizing compression latency costs. In doing so, PRISM on a 256-core system composed of 16 chiplets improves inter-chiplet packet latency by up to 5.8× and, under selective compression, overall system performance by up to 17.4% and 10.0% on average. In our experiments, 11–33% of the network traffic crosses chiplet boundaries and is compressed with compression ratio of 1.1–4.9×. Selective compression skips compression in 18–44% of the considered cases and improves system performance compared to always compressing by up to 6.5%.

Bibliography

[1] I. Advanced Micro Devices, "AMD CDNA 3 White Paper," 2023. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/white-papers/amd-cdna-3-white-paper.pdf

[2] S. Beamer, K. Asanović, and D. Patterson, "The gap benchmark suite," *arXiv*, 2015.

[3] S. Bharadwaj, J. Yin, B. Beckmann, and T. Krishna, "Kite: A family of heterogeneous interposer topologies enabled via accurate interconnect modeling," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[4] W. Cao, S. Jiang, and L. Huang, "Fast turn restriction algorithm to build deadlock-free modular chiplet integration systems," in *2022 IEEE 4th International Conference on Circuits and Systems (ICCS)*, 2022, pp. 27–32.

[5] Z. Cao, W. Guo, Z. Wan, P. Li, Q. Liu, C. Wang, and Y. Shao, "ETRS: Efficient turn restrictions setting method for boundary routers in chiplet-based systems," *The Journal of Supercomputing*, vol. 80, no. 14, pp. 20 488–20 517, 2024.

[6] Z. Cao, Z. Wan, P. Li, Q. Liu, C. Wang, and Y. Shao, "LBDR: A load-balanced deadlock-free routing strategy for chiplet systems," *Integration*, vol. 96, p. 102149, 2024.

[7] A. Coskun, F. Eris, A. Joshi, A. B. Kahng, Y. Ma, A. Narayan, and V. Srinivas, "Cross-layer co-optimization of network design and chiplet placement in 2.5-D systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 5183–5196, 2020.

[8] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[9] R. Das, A. K. Mishra, C. Nicopoulos, D. Park, V. Narayanan, R. Iyer, M. S. Yousif, and C. R. Das, "Performance and power optimization through data compression in network-on-chip architectures," in *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, 2008, pp. 215–225.

[10] D. Deb, R. M.K., and J. Jose, "Flitzip: Effective packet compression for noc in multiprocessor system-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 1, pp. 117–128, 2022.

[11] M. Ebrahimi, A. Y. Weldezion, and M. Daneshtalab, "NoD: Network-on-die as a standalone NoC for heterogeneous many-core systems in 2.5D ICs," in *2017 19th International Symposium on Computer Architecture and Digital Systems (CADS)*, 2017, pp. 1–6.

[12] Y. Feng, W. Li, and K. Ma, "Ring road: A scalable polar-coordinate-based 2D network-on-chip architecture," in *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2024, pp. 871–884.

[13] Y. Feng and K. Ma, "Chiplet Actuary: A Quantitative Cost Model and Multi-Chiplet Architecture Exploration," in *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC)*, Jul 2022, p. 121–126.

[14] Y. Feng, D. Xiang, and K. Ma, "A scalable methodology for designing efficient interconnection network of chiplets," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 1059–1071.

[15] R. Holsmark, S. Kumar, M. Palesi, and A. Mejia, "Hira: A methodology for deadlock free routing in hierarchical networks on chip," in *2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, 2009, pp. 2–11.

[16] P. Iff, M. Besta, and T. Hoefler, "Foldedhexatorus: An inter-chiplet interconnect topology for chiplet-based systems using organic and glass substrates," 2025. [Online]. Available: https://arxiv.org/abs/2504.19878

[17] N. E. Jerger, A. Kannan, Z. Li, and G. H. Loh, "Noc architectures for silicon interposer systems: Why pay for more wires when you can get them (from your interposer) for free?" in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, 2014, pp. 458–470.

[18] F. Jiang, C. Li, L. Chen, J. Liu, W. Zhang, and J. Xu, "SCNoCs: An adaptive heterogeneous multi-NoC with selective compression and power gating," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2024.

[19] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally, "A detailed and flexible cycle-accurate network-on-chip simulator," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013, pp. 86–96.

[20] Y. Jin, K. H. Yum, and E. J. Kim, "Adaptive data compression for high-performance low-power on-chip networks," in *2008 41st IEEE/ACM International Symposium on Microarchitecture*, 2008, pp. 354–363.

[21] A. Kannan, N. E. Jerger, and G. H. Loh, "Enabling Interposer-based Disintegration of Multi-core Processors," in *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2015, pp. 546–558.

[22] S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. Jacob, "DRAMsim3: A cycle-accurate, thermal-capable DRAM simulator," *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 106–109, 2020.

[23] P. Majumder, S. Kim, J. Huang, K. H. Yum, and E. J. Kim, "Remote control: A simple deadlock avoidance scheme for modular systems-on-chip," *IEEE Transactions on Computers*, vol. 70, no. 11, pp. 1928–1941, 2021.

[24] N. B. Mallya, P. Strikos, B. Goel, A. Ejaz, and I. Sourdis, "A performance analysis of chiplet-based systems," in *2025 Design, Automation & Test in Europe Conference (DATE)*, 2025, pp. 1–7.

[25] M. S. Müller, J. Baron, W. C. Brantley, H. Feng, D. Hackenberg, R. Henschel, G. Jost, D. Molka, C. Parrott, J. Robichaux, P. Shelepugin, M. van Waveren, B. Whitney, and K. Kumaran, "Spec omp2012 – an application benchmark suite for parallel systems using openmp," in *Proceedings of the 8th International Conference on OpenMP in a Heterogeneous World*, ser. IWOMP'12. Berlin, Heidelberg: Springer-Verlag, 2012, p. 223–236.

[26] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: A tool to model large caches," HP Laboratories, Tech. Rep. HPL-2009-85, 2009.

[27] S. Naffziger, N. Beck, T. Burd, K. Lepak, G. H. Loh, M. Subramony, and S. White, "Pioneering Chiplet Technology and Design for the AMD EPYC™ and Ryzen™ Processor Families : Industrial Product," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, Jun 2021, pp. 57–70.

[28] L. Nai, Y. Xia, I. G. Tanase, H. Kim, and C.-Y. Lin, "Graphbig: understanding graph computing in the context of industrial solutions," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2807591.2807626

[29] N. Nassif, A. O. Munch, C. L. Molnar, G. Pasdast, S. V. Lyer, Z. Yang, O. Mendoza, M. Huddart, S. Venkataraman, S. Kandula, R. Marom, A. M. Kern, B. Bowhill, D. R. Mulvihill, S. Nimmagadda, V. Kalidindi, J. Krause, M. M. Haq, R. Sharma, and K. Duda, "Sapphire Rapids: The Next-Generation Intel Xeon Scalable Processor," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2022, pp. 44–46.

[30] N. Niwa, Y. Shikama, H. Amano, and M. Koibuchi, "A case for low-latency network-on-chip using compression routers," in *2021 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2021, pp. 134–139.

[31] G. Pekhimenko, V. Seshadri, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "Base-delta-immediate compression: Practical data compression for on-chip caches," in *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*, 2012, pp. 377–388.

[32] C. Sakalis, C. Leonardsson, S. Kaxiras, and A. Ros, "Splash-3: A properly synchronized benchmark suite for contemporary research," in *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2016, pp. 101–111.

[33] D. Sanchez and C. Kozyrakis, "ZSim: Fast and accurate microarchitectural simulation of thousand-core systems," in *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA)*, 2013, pp. 475–486.

[34] H. Shabani and X. Guo, "Cluscross: a new topology for silicon interposer-based network-on-chip," in *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, ser. NOCS '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3313231.3352363

[35] D. Stow, Y. Xie, T. Siddiqua, and G. H. Loh, "Cost-Effective Design of Scalable High-performance Systems Using Active and Passive Interposers," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2017, pp. 728–735.

[36] P. Strikos, A. Ejaz, and I. Sourdis, "BZSim: Fast, Large-scale Microarchitectural Simulation with Detailed Interconnect Modeling," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, May 2024.

[37] E. Taheri, S. Pasricha, and M. Nikdast, "DeFT: A deadlock-free and fault-tolerant routing algorithm for 2.5D chiplet networks," in *Design, Automation and Test in Europe Conference (DATE)*, 2022, pp. 1047–1052.

[38] ——, "ReD: A reliable and deadlock-free routing for 2.5-D chiplet-based interposer networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 12, pp. 4599–4612, 2024.

[39] Y. Wang, H. Li, Y. Han, and X. Li, "A low overhead in-network data compressor for the memory hierarchy of chip multiprocessors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 6, pp. 1265–1277, 2018.

[40] Y. Wu, L. Wang, X. Wang, J. Han, J. Zhu, H. Jiang, S. Yin, S. Wei, and L. Liu, "Upward packet popup for deadlock freedom in modular chiplet-based systems," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 986–1000.

[41] J. Yin, Z. Lin, O. Kayiran, M. Poremba, M. S. B. Altaf, N. Enright Jerger, and G. H. Loh, "Modular routing design for chiplet-based systems," in *Proceedings of the 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018, pp. 726–738.

[42] D. Yu, A. Li, N. Jing, J. Jiang, W. Sheng, and Q. Wang, "VDA: A simple but efficient virtual-channel-based deadlock avoidance scheme for scalable chiplet networks," in *Proceedings of the Great Lakes Symposium on VLSI 2024*. Association for Computing Machinery, 2024, pp. 357–363.

[43] J. Zhan, M. Poremba, Y. Xu, and Y. Xie, "Noδ: Leveraging delta compression for end-to-end memory access in noc based multicores," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014, pp. 586–591.

[44] X. Zhang, M. Wang, Y. Zhang, T. Lu, and Z. Yu, "A scalable deadlock-free static routing algorithm for chiplet-based systems," in *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*, 2023, pp. 1350–1357.