DISSERTATION

# DESIGN AND OPTIMIZATION OF EFFICIENT, FAULT-TOLERANT AND SECURE 2.5D CHIPLET SYSTEMS

Submitted by

Ebad Taheri

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2024

Doctoral Committee:

Advisor: Mahdi Nikdast
Co-Advisor: Sudeep Pasricha

Yashwant K. Malaiya
Anura P. Jayasumana

# ABSTRACT

## DESIGN AND OPTIMIZATION OF EFFICIENT, FAULT-TOLERANT AND SECURE 2.5D CHIPLET SYSTEMS

In response to the burgeoning demand for high-performance computing systems, this Ph.D. dissertation investigates the pivotal challenges surrounding Networks-on-Chip (NoCs) within the framework of 2.5D and 3D integration technologies, with a primary objective of enhancing the efficiency, fault tolerance, and security of forthcoming computing system architectures. The inherent limitations in bandwidth and reliability at the boundary of chiplets in 2.5D chiplet systems engender significant challenges in traffic management, latency, and energy efficiency. Furthermore, the interconnected global network on an interposer, linking multiple chiplets, necessitates high-bandwidth, low-latency communication to accommodate the substantial traffic generated by numerous cores across diverse chiplets. This Ph.D. dissertation emphasizes various design aspects of NoCs, such as latency, energy efficiency, fault tolerance, and security. It explores the design of 3D NoCs leveraging Through-Silicon Vias (TSVs) for vertical communication. To address reliability concerns and fabrication costs associated with high TSV density, Partially Connected 3D NoC (PC-3DNoC) has been proposed. An adaptive congestion-aware TSV link selection algorithm is introduced to manage traffic load and optimize communication, resulting in reduced latency and improved energy efficiency. For 2.5D chiplet systems, a novel deadlock-free and fault-tolerant routing algorithm is presented. The fault-tolerant algorithm enhances redundancy in vertical link selection and offers improved network reachability with reduced latency compared to existing solutions, even in the presence of faults. Furthermore, to address the energy consumption concerns of silicon-photonic-based 2.5D networks, a reconfigurable power-efficient and congestion-aware silicon-photonic-based 2.5D Interposer network is proposed. The proposed photonic interposer utilizes phase change materials (PCMs) for dynamic reconfiguration and power gating of the pho-

tonic network, leading to lower latency and improved energy efficiency. Additionally, the research investigates the integration of optical computation and communication into 2.5D chiplet platforms for domain-specific machine learning (ML) processing. This approach aims to overcome limitations in computation density and communication speeds faced by traditional accelerators, paving the way for sustainable and scalable ML hardware. Furthermore, this dissertation proposes a 2.5D chiplet-based architecture utilizing a silicon-photonic-based interposer, which tackles the limitations of conventional bus-based communication by employing a novel switch-based network, achieving significant energy efficiency improvements for high-bandwidth, low-latency data movement in machine learning accelerators. The switch-based network employs our proposed optical switch based on Mach–Zehnder Interferometer (MZI) devices with a dividing state to facilitate broadcast and optimize communication for ML workloads. Finally, the dissertation explores security considerations in 2.5D chiplet systems with diverse, potentially untrusted chiplets. To address this, a secure routing framework for Network-on-Interposer is presented. The proposed secure framework protects the system against distributed denial-of-service (DDoS) attacks by concealing predictable routing paths. It leverages multi-objective optimization to balance efficiency and reliability for the NoI. The proposed contributions in this dissertation help advance the field of chip-scale interconnection networks by proposing novel techniques for improved performance, reliability, and power efficiency in 3D and 2.5D NoC architectures. These advancements hold promise for the design of future high-performance computing systems, particularly in the areas of machine learning and other computationally intensive applications.

# ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Prof. Mahdi Nikdast, for his invaluable guidance, unwavering support, and endless patience throughout the journey of this dissertation. His expertise, encouragement, and constructive feedback have been instrumental in shaping this work. Whenever deadlines loomed, he devoted his time to ensure our progress. Beyond the technical realm, Prof. Nikdast extended his support to me during personal hardships, such as the challenging period of pandemic-induced separation from my wife. His unwavering commitment began from my first email to him in Iran, my home country, and continues to this day, five years later. His dedication to our growth and success is truly commendable.

I am also grateful to my co-advisor, Prof. Sudeep Pasricha, for his insightful input, mentorship, and dedication to helping me navigate the complexities of my research. His perspective and encouragement have been immensely valuable in refining my ideas and methodologies. Prof. Pasricha generously devoted his time to assist me in every step of this project, and I consider myself fortunate to have such a knowledgeable and supportive mentor.

I extend my heartfelt thanks to the members of my committee, Prof. Anura P. Jayasumana and Prof. Yashwant K. Malaiya, for their time, expertise, and constructive criticism. Their diverse perspectives and thoughtful insights have greatly enriched the quality of this dissertation.

Special gratitude goes to my mentor and collaborator outside Colorado State University. Prof. Ahmad Patooghy, my master's degree advisor, shaped my mindset and encouraged me to pursue a Ph.D. I also want to express my appreciation to Prof. Rayan Kim, who provided significant assistance and collaboration during the first two years of my Ph.D. project. Additionally, I am thankful to Prof. Nader Sehatbaksh and his student, Pooya Aghanoury, at UCLA for their collaboration, as well as Prof. Kaveh Rahbardar Mojaver for their assistance in my Ph.D. project.

To my collaborators and labmates, including Amin Mahdian, Kamil Khan, Febin Sunny, Alex Qi, and Asif Mirza, I express my gratitude for your camaraderie, support, and intellectual exchange. I have learned a great deal from each of you and am thankful for the ideas we have built

together. Additionally, I am grateful to my colleagues at EPIC lab and ECSyD lab. Your friendship and collaboration have enriched this journey, making it both enjoyable and rewarding.

I owe a debt of gratitude to my wife, Mydeh, for her unwavering love, understanding, and patience. Her encouragement, support, and sacrifices have been the bedrock of my success, and I am profoundly grateful for her presence in my life.

Last but not least, I want to thank my parents for their unconditional love, support, and belief in me. Their encouragement and sacrifices have been a source of strength and motivation throughout this endeavor.

This dissertation would not have been possible without the support and encouragement of all those mentioned above, as well as many others who have contributed in ways both seen and unseen. Thank you all for being part of this transformative journey.

LIST OF RESEARCH PUBLICATIONS

1. **E. Taheri**, M. A. Mahdian, S. Pasricha, and M. Nikdast, "SwInt: A Non-Blocking Switch-Based Silicon Photonic Interposer Network for 2.5D Machine Learning Accelerators", IEEE Journal on Emerging and Selected Topics in Circuits and Systems.

2. **E. Taheri**, S. Pasricha, and M. Nikdast, "ReD: A Reliable and Deadlock-Free Routing Algorithm for 2.5D Chiplet Networks", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

3. **E. Taheri**, P. aghanoury, S. Pasricha, M. Nikdast, and N. Sehatbakhsh, "SCRIPT: A Multi-Objective Routing Framework for Securing Chiplet Systems against Distributed DoS Attacks", Proceedings of the Great Lakes Symposium on VLSI.

4. M. A. Mahdian, **E. Taheri**, K. Rahbardar Mojaver, and M. Nikdast, “Photonic Physically Unclonable Functions using Ring-Assisted Contra-Directional Couplers”, IEEE/Optica Optical Fiber Communication (OFC) Conference, 2024.

5. F Sunny, **E. Taheri**, M Nikdast, and S Pasricha, "Silicon Photonic 2.5 D Interposer Networks for Overcoming Communication Bottlenecks in Scale-out Machine Learning Hardware Accelerators", IEEE VLSI Test Symposium, 2024.

6. **E. Taheri**, M. A. Mahdian, S. Pasricha, and M. Nikdast, "TRINE: A Tree-Based Silicon Photonic Interposer Network for Energy-Efficient 2.5 D Machine Learning Acceleration", Proceedings of the 16th International Workshop on Network on Chip, 2023.

7. M. A. Mahdian, **E. Taheri**, and M. Nikdast, "Pars: A power-aware and reliable control plane for silicon photonic switch fabrics", International Conference on Photonics in Switching and Computing (PSC), 2023.

8. **E. Taheri**, R. G. Kim, and M. Nikdast, "AdEle+: An Adaptive Congestion-and-Energy-Aware Elevator Selection for Partially Connected 3D Networks-on-Chip", IEEE Transactions on Computers, 2023.

9. F Sunny, **E. Taheri**, M Nikdast, and S Pasricha, "Machine Learning Accelerators in 2.5 D Chiplet Platforms with Silicon Photonics", IEEE/ACM Design, Automation and Test in Europe (DATE) Conference and Exhibition, 2023.

10. **E. Taheri**, S. Pasricha, and M. Nikdast, "ReSiPI: A Reconfigurable Silicon-Photonic 2.5D Chiplet Network with PCMs for Energy-Efficient Interposer Communication", IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2022.

11. **E. Taheri**, S. Pasricha, and M. Nikdast, "DeFT: A Deadlock-Free and Fault-Tolerant Routing Algorithm for 2.5D Chiplet Systems", IEEE/ACM Design, Automation and Test in Europe (DATE) Conference and Exhibition, 2022.

12. F. Sunny, **E. Taheri**, M. Nikdast, and S. Pasricha, "A Survey on Silicon Photonics for Deep Learning," ACM Journal on Emerging Technologies in Computing Systems (JETC) , 2021.

13. **E. Taheri**, R. G. Kim, and M. Nikdast, "AdEle: An Adaptive Congestion-and-Energy-Aware Elevator Selection for Partially Connected 3D NoCs," IEEE/ACM Design Automation Conference (DAC), 2021.

14. A. Mirza, S. Manafi Avari, **E. Taheri**, S. Pasricha, and M. Nikdast, "Opportunities for Cross-Layer Design in High-Performance Computing Systems with Integrated Silicon Photonic Networks", IEEE/ACM Design, Automation and Test in Europe (DATE) Conference and Exhibition, 2020.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# Chapter 1

# Introduction

The relentless demand for high computational power has been a driving force in the semiconductor industry, leading to the development of innovative chip designs focused on scalability and performance enhancement. One notable solution that emerged from this pursuit is the integration of multiple cores onto a single chip, marking the era of multicore chips and system-on-chips (SoCs), where numerous processing/memory cores are integrated onto the same chip. However, utilizing numerous processing cores in chips with traditional communication infrastructures poses significant challenges. NoCs have been introduced as a suitable communication platform for inter-core communications within chips. The possibility of fabricating three-dimensional integrated circuits and consequently creating three-dimensional on-chip networks has been a significant achievement in chip fabrication, particularly in addressing the challenges of traditional chips, especially in terms of latency. However, three-dimensional integration leads to a substantial increase in power density in chips. The high power density in these networks creates hot spots, posing a threat to the reliability of chips. Furthermore, chiplet systems, as an innovative technology, improve scalability and fabrication cost by disintegrating large-scale chiplets and connecting them through a global interposer. 3D and 2.5D chiplets both pave the way for a high density of cores on the same chip, aiming for high computational power and energy efficiency, especially with recent advances in machine learning applications, where there is a high demand for parallel processing and high computational power with minimized energy consumption to address sustainability. However, there are significant energy efficiency, fault tolerance, and security challenges in providing efficient communication for such new technologies. Therefore, designing and optimizing efficient NoCs for these innovative technologies is of great importance. Silicon photonic technology, in which on-chip communication is enabled by exchanging data using optical signals, is also a great candidate to be used in global interposer networks, offering high bandwidth and low latency for relatively longer distances on the chip.

This chapter introduces multicore chips, SoCs, NoCs, 3D, and 2.5D integration. Moreover, it discusses how silicon photonics can be useful in enhancing the energy efficiency and performance of 2.5D chiplet systems. It also briefly discusses energy efficiency, fault tolerance, and security in these networks, and briefly outlines the objectives of this dissertation.

## 1.1 Multicore and system-on-chip

Multicore chips represent a paradigm shift in chip architecture, where several processing cores are integrated on the same chip to leverage parallel processing capabilities. For instance, IBM Cyclops-64 chip includes 160 processing cores integrated on the same chip [2]. By incorporating multiple cores, multicore chips enable concurrent execution of tasks, thereby significantly boosting computational throughput and efficiency. The concept of multicore chips extends beyond homogeneous architectures to embrace heterogeneity, exemplified by the advent of System-on-Chip (SoC) designs. In SoC implementations, diverse processing units such as central processing units (CPUs), graphics processing units (GPUs), caches, and specialized accelerators are integrated onto a single chip. This heterogeneous architecture facilitates efficient utilization of resources by allocating specific tasks to the most suitable processing element, thus optimizing performance for various workloads. The integration of heterogeneous cores within SoCs introduces a new era of computational efficiency and versatility. By leveraging the strengths of different processing units, SoCs empower applications to harness a broader spectrum of computational capabilities, ranging from general-purpose computing tasks handled by CPUs to parallelized data processing facilitated by GPUs and accelerators. Furthermore, the scalability inherent in multicore and heterogeneous chip designs aligns with the ever-growing demands of modern computing applications. As workloads become increasingly complex and diverse, the ability to seamlessly scale computational resources through multicore and heterogeneous architectures becomes paramount in meeting performance requirements while managing power consumption and cost constraints.

However, assuming a large number of cores on a single chip, an effective communication paradigm should be designed to achieve high performance.

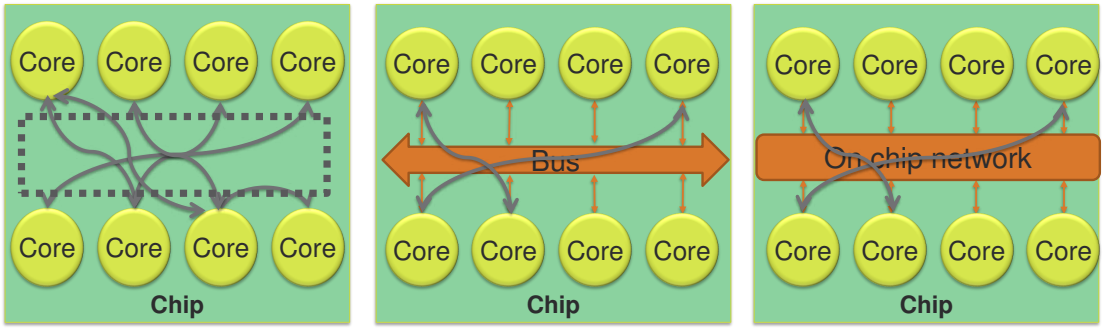



**Figure 1.1:** On chip communication paradigm: (a) In point-to-point communication each core is directly connected to the other core using a physical link, (b) In Bus-based communicaiton the bus is shared between cores, and (c) In Network-on-chip, simultaneous communication between core can be handled.

## 1.2 On-chip communication

Point-to-point communication, where there exists a dedicated physical connection between two cores, is not scalable. Assuming $N$ cores on a chip, providing communication between all cores would require $N \times (N-1)$ direct connections (almost equal to $N^2$ in a large-scale chip). Consequently, point-to-point communication proves impractical for current and future SoCs. The primary concern stems from limitations in the number of metal layers available on a chip's manufacturing technology.

To alleviate such limitations, traditional bus-based communication architectures have been introduced, where one or multiple buses are shared among all cores. However, bus-based communication suffers from scalability issues, both in terms of latency and performance. When a bus is allocated for communication between two cores, other cores sharing the same bus must wait until the communication is complete before proceeding with their own communication tasks. This waiting introduces latency into the communication process, and as the number of cores sharing the same bus increases, so does the latency imposed on the communication packets.

To address these scalability concerns in on-chip communication, Networks-on-Chip (NoCs) have been proposed. NoCs offer a more efficient and scalable communication paradigm compared to traditional point-to-point and bus-based architectures. These communication paradigms are compared in Figure 1.1.

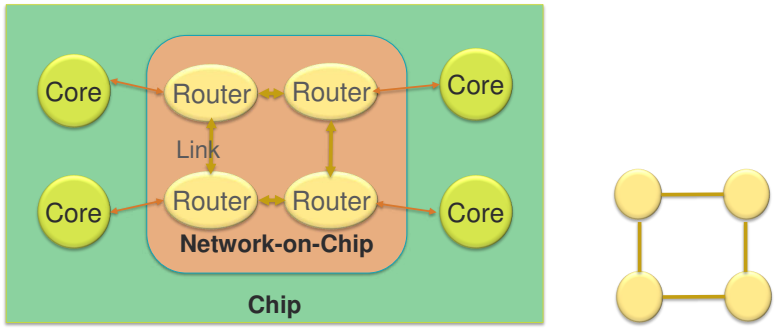



**Figure 1.2:** (a) a Networks-on-Chip with four cores, and (b) the corresponding topology of this network.

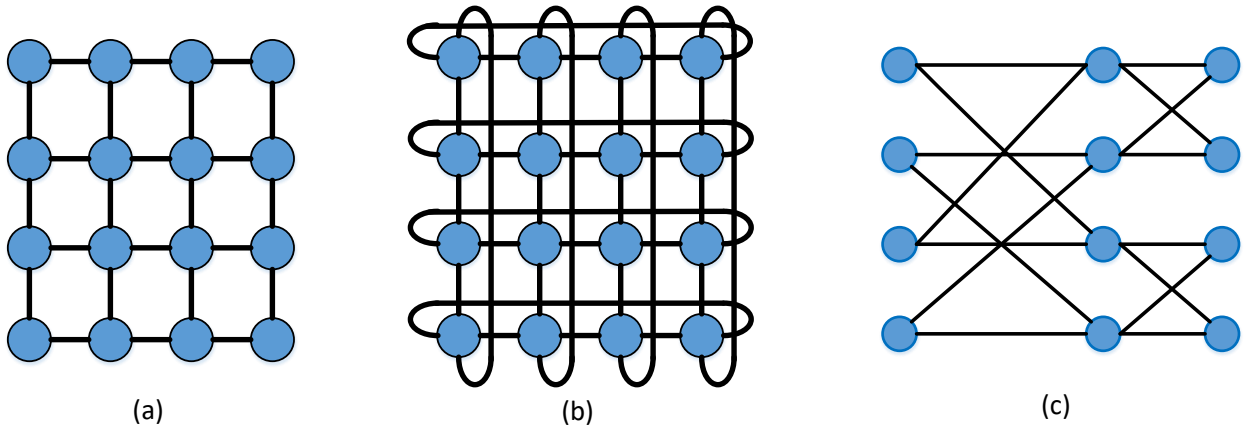



**Figure 1.3:** Network-on-Chip topology examples: (a) mesh, (b) torus, and (c) butterfly.

## 1.3 Networks-on-Chip

In comparison with conventional communication paradigms such as point-to-point and bus-based communication, NoCs provide efficient communication for on chip systems because they present high performance, low power consumption, and more scalability. NoCs provide communication for processing elements employing routers which are connected using interconnects. NoC has emerged as a prevalent solution to enable scalable on-chip communication in manycore systems [3]. The Figure 1.2 illustrates a NoCs architecture featuring four cores, along with the corresponding topology depicting the interconnection scheme between these cores.

### 1.3.1 Topology

NoC architectures provide versatile communication flexibility through a variety of topologies, which dictate how routers are interconnected. These topologies, exemplified by mesh, torus, and butterfly configurations in Figure 1.3, significantly influence network performance based on specific application requirements. The mesh topology establishes a grid-like interconnection, enabling horizontal and vertical communication among cores. Mesh is favored for its simplicity, as its planar topology reduces fabrication costs and ensures uniform link lengths, facilitating low worst-case link delay for high-frequency operation. However, it suffers from a large diameter and average distance between nodes, particularly at the network's edges where connectivity is lower. To address this, the torus topology forms a toroidal structure, linking outermost cores, thus enhancing communication efficiency and reducing latency. Nonetheless, torus configurations may incur longer link delays and increased complexity. Meanwhile, the butterfly topology organizes cores hierarchically, enabling parallel communication paths resembling a butterfly's wings. These diverse topologies present trade-offs in scalability, latency, and power consumption, allowing designers to customize NoC architectures to meet specific SoC design needs. Additionally, NoC topologies, categorized into standard and application-specific configurations, determine router and core interconnections. Standard topologies ensure connectivity between all routers, while optimized configurations match traffic patterns for improved performance and reduced costs. Mesh and torus configurations are prevalent in standard setups, offering simplicity and efficiency, while specialized configurations cater to specific application requirements, enhancing performance at the expense of planar properties or complexity.

### 1.3.2 Router architecture

A router in an NoC, as shown in Figure 1.4, comprises the following main components:

- Buffers: These buffers are of the first-in-first-out (FIFO) type and are used to store packets passing through the router. Routers can have input buffers, output buffers, or both. For example, an input-buffered router stores incoming data first and then, after routing mechanisms,

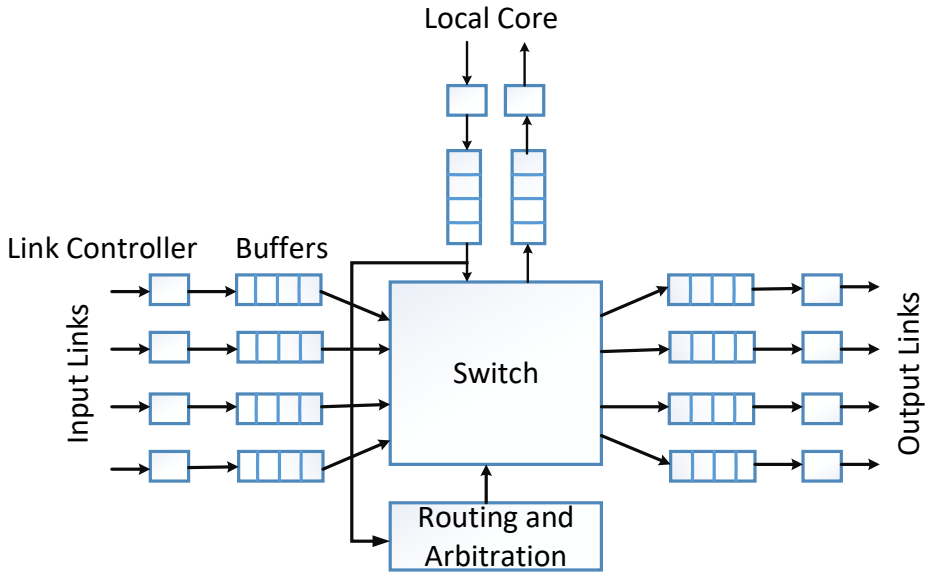



**Figure 1.4:** NoC router architecture.

switch allocation, and link assignment, sends the data to the output link. In the depicted model in Figure 1.4, both input and output channels have buffers.

- Switch: This unit is responsible for connecting input buffers or links of the router to the corresponding output buffers or links.

- Routing and Arbitration Unit: This unit implements routing algorithms, uses the output link for an input packet, and sets the crossbar accordingly. If multiple packets simultaneously request the use of the same output link, this unit arbitrates between them. If the link is busy, the input packet waits in the input buffer, and eventually, after the link becomes available again, it is routed.

- Link Controller: The flow of packets in the physical channel between neighboring routers is managed by the link controller unit. A link controller is placed on both sides of a channel to transmit flow control units.

### 1.3.3 Switching

Data communication in NoC involves dividing data intended for transmission between cores into packets, which are further divided into smaller units called flits. Each packet is then divided into even smaller units called fits, determining the width of the communication channel, typically considered equal to a flit. Switching can be categorized based on temporal behavior and packet storage methods [4]:

**Circuit-switched Routing**

This method involves reserving a path (circuit) from the source to the destination before data transmission. All packets of a message utilize the same path. A route-request is sent from the source to the destination to establish the circuit, allocating channels for packet transmission. Circuit channels remain dedicated to the source-destination pair until data transmission ends, after which resources are released. However, circuit construction overhead and difficulty in finding circuits during network congestion are drawbacks.

**Packet-based Routing**

This method does not require route establishment before data transmission. Routing is performed separately for each packet, potentially leading to different routes for different packets of the same message. Packet-based routing includes three types:

- Store-and-Forward: Upon reaching a node, the entire packet is stored in the buffer, and routing decisions are made based on decoded routing information. If the next router can accommodate all packets, the packet is forwarded without complete reception. This method ensures accurate routing but may lead to higher latency.

- Direct Virtual Cut-Through: Routing information is placed in the header of the first flit. Only the first flit is stored, and routing decisions are made based on this information, enabling faster transmission compared to store-and-forward routing.

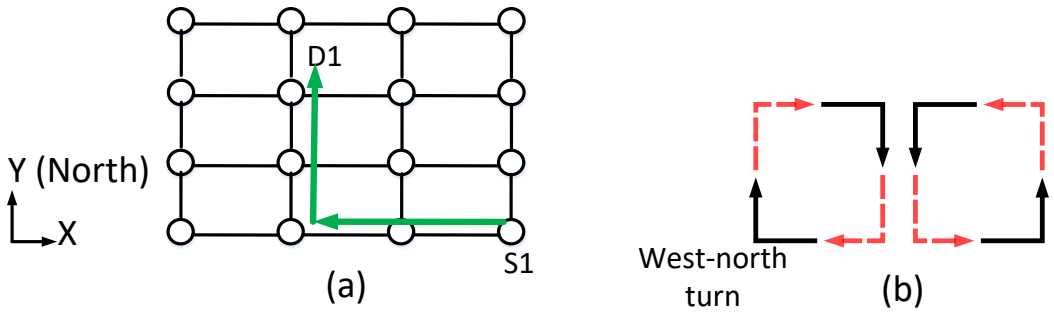



**Figure 1.5:** Routing example: (a) XY routing. (b) turn model of XY routing

- Wormhole switching involves breaking data into small packets, each with a control header for routing. These packets can be transmitted through different paths, reducing buffer usage compared to direct virtual cut-through. Wormhole switching is widely regarded as the most popular and efficient switching method in NoC, and it will be the primary switching method discussed throughout the remainder of this dissertation. However, it's important to note that the likelihood of deadlocks, or infinite cyclic dependencies in the network, increases significantly with this method, necessitating further discussion and the exploration of mitigation strategies in subsequent chapters.

### 1.3.4 Routing

Routing in NoC architectures dictates how packets traverse intermediate routers from a source router to a destination router. Figure 1.5 provides an illustration of an XY routing example, showcasing the routing path taken by data packets through a mesh topology. It also depicts the potential turns resulting from XY routing. For instance, when routing from S1 to D1, the west-north turn is taken, as shown in the figure. However, since YX routing is not implemented, the dashed-red turns are not taken, effectively breaking a cycle. This cycle-breaking mechanism is crucial for avoiding deadlocks, a topic that will be discussed in more detail later in this section.

Routing strategies are categorized based on several criteria. Firstly, routing can be either distributed or centralized. In distributed routing, each router makes decisions locally based on packet information, while in centralized routing, decisions are made for the entire network from specific

locations. Additionally, routing can be single-destination or multi-destination, allowing one sender to transmit data to multiple receivers. Furthermore, routing can be oblivious, with no knowledge of network conditions, or adaptive, adjusting routing paths based on network conditions, known as adaptive routing. In adaptive routing, decision-making can occur locally at each node. Routers play a crucial role in routing by determining how to forward inputs to outputs using specific algorithms. Each router utilizes routing information contained in the packet header to route the packet to the appropriate output. Despite these processes, routing in NoC architectures can encounter challenges leading to packet loss or delays.

The three main concerns in routing are deadlock, livelock, and starvation. Deadlock occurs when packets wait for each other to proceed, resulting in cyclical dependencies and network saturation. Livelock arises when packets circulate without reaching their destination. Starvation can occur when packets with lower priorities never reach their destination due to resource allocation to higher-priority packets. One of the most important aspects of designing a deadlock-free routing algorithm is the demand for cycle breaking. The cyclic dependency between buffer of routers results in infinite waiting between channels and deteriorates network performance. Deadlock mitigation methods are broadly divided into two categories: turn model-based and virtual channel-based routing algorithms.

### 1.3.5 Deadlock recovery

Deadlock recovery in NoC architectures refers to the process of detecting and resolving deadlock situations where multiple routers are blocked, unable to proceed due to circular dependencies in buffer allocation. While deadlock recovery mechanisms can help revive a system from such situation, it is critical to avoid deadlock altogether rather than relying solely on recovery strategies.

Deadlocks can bring communication to a grinding halt, rendering the entire system non-functional. Therefore, it is imperative to prioritize the prevention of deadlock rather than relying solely on recovery strategies.

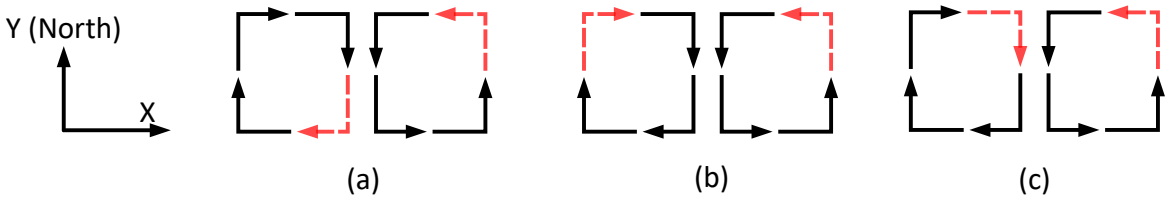



**Figure 1.6:** Deadlock free turn model: (a) west-first turn model, (b) north-last turn model, and (c) negative-first turn model.

Deadlock recovery mechanisms, while useful as a last resort, come with inherent drawbacks. They introduce complexity and overhead to the system, which can degrade performance and efficiency. Additionally, the time taken for deadlock recovery can lead to significant delays, particularly in time-sensitive applications, impacting system responsiveness and reliability.

Moreover, relying on recovery strategies alone does not address the root cause of deadlock. It merely resolves immediate deadlock situations without guaranteeing prevention of future occurrences. This can result in a vicious cycle of deadlock detection and recovery, leading to system instability and inefficiency.

Therefore, the focus should be on implementing deadlock avoidance strategies as a fundamental aspect of NoC design. By prioritizing deadlock avoidance, designers can create more robust and reliable systems that operate seamlessly without the need for frequent recovery interventions. This approach ensures the continuous and uninterrupted flow of data within the NoC, safeguarding system performance and functionality.

### 1.3.6 Deadlock avoidance

#### Turn Model

As we discussed the turns in XY routing, resulted in deadlock-free routing since the resulting turns cannot make a cycle. The turn model algorithm determines that one or more turns during packet routing are not allowed, therefore cutting off cyclic dependencies between channels. Three famous models have been described for this method, which are shown in Figure 1.6:

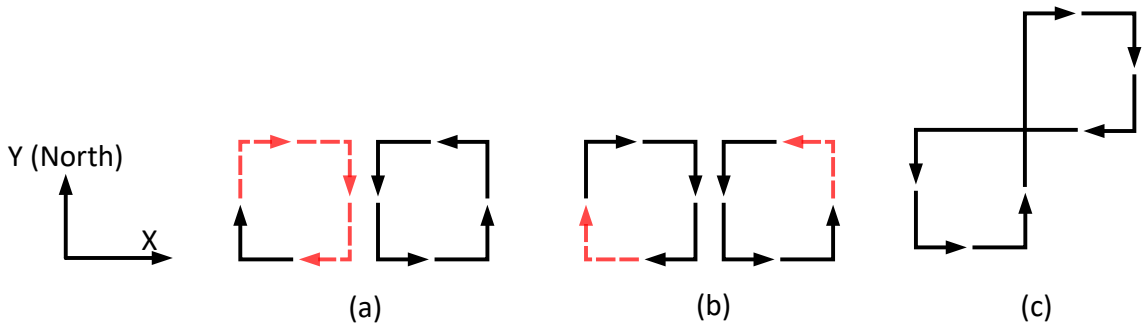



**Figure 1.7:** Deadlock turn model examples: (a) the turns in the right-hand cycle create cyclic dependency, (b) the combination of the turns in both cycles can create cyclic dependency, and (c) an example of the combination from the avoided turn in b resulted in a deadlock cycle

- West-first Turn Model: In this turn model, all the turn towards the west are prohibited. while this rule breaks any possible cyclic dependency, a packet that moves westward cannot be routed adaptively.
- North-Last Turn Model: In this turn model, similarly, a packet that moves northward cannot be routed adaptively routed. In other words, after moving northward, it cannot turn anymore.
- Negative-first Turn Model: In this turn model, a packet that moves in the positive direction cannot turn in the negative direction.

As it can be seen, by only avoiding one turn among the four turns in each cycle, deadlock can be avoided. However, there are cases where avoiding one turn per cycle cannot break the cyclic dependency, as shown in Figure 1.7

### Virtual Channels

Flow control mechanisms between two routers are such that packets between two routers is stored in the input and output buffers of each channel (link), and passage through the physical channel only occurs at certain time clocks. Since the buffers are FIFO, the packets that enter first is prepared for transmission to the next path before others; therefore, when a packet occupies the buffers of a channel, another packet, even when physically free, cannot access the channel. In this situation, by adding a buffer next to the buffers of each channel, it is possible to use the physical

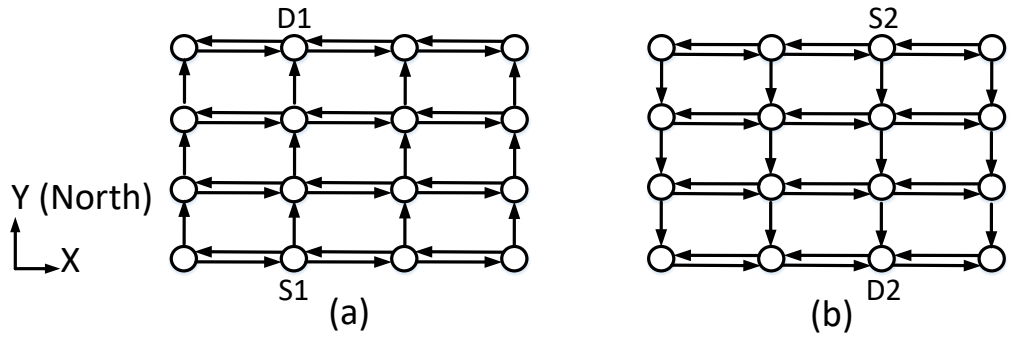



**Figure 1.8:** Sub network to achieve deadlock freedom while offering fully adaptive routing: (a) adaptive routing from south to north (e.g., S1 to D1, and (b) adaptive routing from north to south (e.g., S2 to D2.)

channel, effectively sharing the physical channel among virtual channels. In this case, it is possible to save on hardware overhead by adding virtual channels. By adding virtual channels, the flow of each virtual channel is separated, and different turns can be added to virtual channels. In this case, the turn model is examined separately for each virtual channel, and it is possible to transfer information from one channel to another as long as cyclic dependencies between virtual channels are not created. For example, if there are two virtual channels, it is possible to send a packet from the first channel to the second, provided that a packet from the second channel is not sent to the first. Therefore, virtual channels not only improve network performance but also potentially can break the cyclic dependencies and prevent deadlock at the cost of hardware overhead due to the addition of buffers. For instance, two subnetworks presented in Figure 1.8 can prevent deadlock and offer fully adaptive routing, if one virtual channel is assigned to each sub network. Therefore, following the routing rules (avoided/allowed turns) in each sub-network, deadlock freedom is guaranteed.

### 1.3.7 3D Networks-on-Chip

Conventional two-dimensional NoC (2D NoCs) suffer from low scalability to support low-latency communication of a large number of on-chip processing cores. To address this challenge, three-dimensional NoCs (3D NoCs) have been proposed to offer high-performance communication for modern computers [5]. With the advances in three-dimensional (3D) integration technologies, systems with stacked dies are interconnected using Through-Silicon Vias (TSVs), further improv-

ing NoC scalability, integration density, and system heterogeneity [6–8]. From a network perspective, 3D topologies reduce the average inter-node distance and save energy consumption [5]. However, connecting the layers of 3D chips necessitates the use of inefficient vertical link technologies. The most promising vertical link technology is TSV, although it introduces high cost and low reliable links. Partially connected NoCs (PCNoCs) have been proposed to alleviate the high cost of vertical links by utilizing a smaller number of TSVs. In 3D NoCs, each vertical link (a.k.a. elevator) includes tens or even hundreds of TSV wires. Therefore, removing some elevators greatly lightens the fabrication cost of the NoCs while imposing a small latency overhead. However, as some links are eliminated, the network topology becomes an irregular one, which makes the routing process more complex. Elevators are shared among routers, and each router selects an elevator to route the packet to another layer.

The challenges of 3D NoCs include high fabrication costs, power density issues, and limited cooling conductivity, leading to thermal hotspots and reliability concerns [9, 10]. To mitigate these challenges, 2.5D NoCs (a.k.a. chiplet systems), offer a compelling alternative. In a 2.5D NoCs, chiplets are integrated on an interposer, enabling inter-chiplet communication while maintaining modularity. This approach not only addresses the drawbacks of 3D integration but also facilitates improved manufacturing yield by disintegrating large chips into smaller, interconnected chiplets. With the continuous demand for high compute performance and parallelism in data-intensive applications, 2.5D chiplet systems present a scalable solution, allowing for easier integration of predesigned chiplets and speeding up time-to-market for electronics designs.

### 1.3.8 2.5D Networks-on-Chip (chiplet systems)

The continuous expansion of data- and compute-intensive applications, such as big data analytics and deep learning, has spurred the development of large-scale chips with heightened compute performance and extensive parallelism. These chips, featuring numerous processing cores ranging from several tens to hundreds, provide the computational prowess necessary for emerging applications but are hindered by low manufacturing yields owing to their substantial chip size. In response

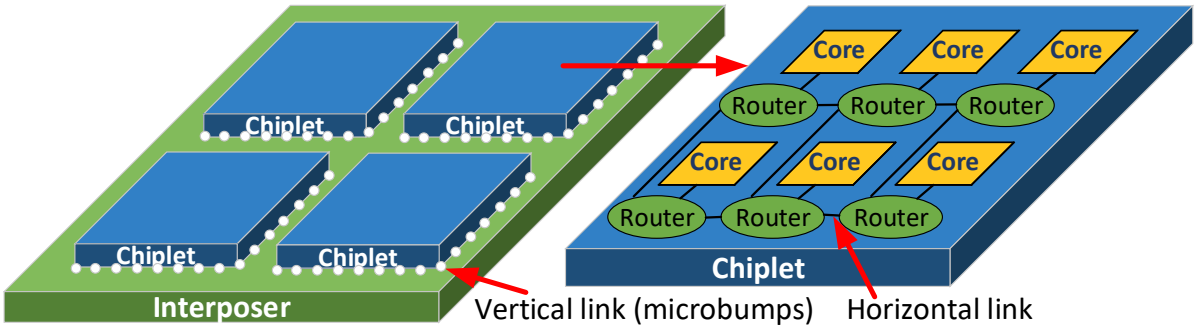



**Figure 1.9:** An example of a 2.5D chiplet system with four chiplets connected through an interposer.

to this challenge, 2.5D chiplet systems have emerged, where a large chip is fragmented into several smaller chiplets interconnected via an inter-chiplet interposer network, substantially enhancing collective manufacturing yield. Illustrated in Figure 1.9, a typical 2.5D chiplet system integrates multiple chiplets on an interposer, facilitating inter-chiplet communication. Each chiplet typically incorporates multiple processing cores interconnected by an intra-chiplet mesh-based NoC. This modular approach allows for the assembly of chiplets into new systems, effectively addressing design challenges, particularly time to market considerations, in the electronics domain.

Moreover, the 2.5D integrated approach offers a modular solution by housing several chiplets on an interposer that supports inter-chiplet communication. Similar to 3D SoCs, such modular integration enables heterogeneous and independent design of each chiplet. Furthermore, the design cycle for chiplet systems is notably shortened compared to traditional monolithic designs, as off-the-shelf chiplets can be readily integrated on an interposer to create chiplet systems with varying computation capabilities tailored for different applications. Consequently, chiplet-based systems can harness diverse manufacturing processes and technologies for each chiplet, amplifying the overall flexibility and efficiency of the system architecture. Given the escalating demands for high bandwidth and low latency in modern computing tasks, particularly in data- and compute-intensive applications, the adoption of optical communication on the interposer emerges as a compelling solution to meet these stringent requirements effectively.

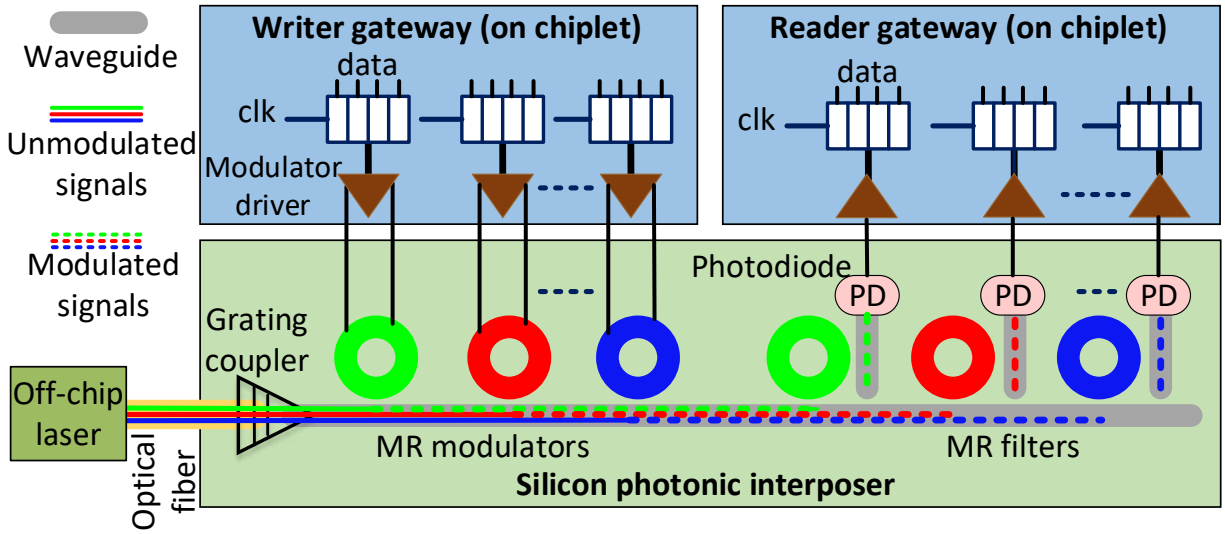



**Figure 1.10:** Data transmission on a silicon photonic interposer.

## 1.4 Silicon Photonic Networks-on-Chip

Electronic interconnects in most of the designs (e.g designs with long interconnects) results in insufficient energy efficiency, high latency and low bandwidth. In such conventional designs, energy efficiency is limited within the distance of smaller than 300mm [11]. Moreover, some effects like skin effect deteriorate into higher losses; and wiring density results in losses because of resistance and capacitance of wires. As a result, conventional electronic interconnect is not a promising solution for the future of NoCs [12].

Silicon photonic communication offers a paradigm shift in on-chip data transfer, particularly advantageous for scenarios demanding high bandwidth and low energy consumption. While conventional metallic-based NoCs suffice for intra-chiplet communication within small-scale designs, silicon photonic solutions excel in inter-chiplet communication, especially on platforms like interposers. Leveraging silicon photonic technology on an interposer enables the provision of high-bandwidth communication among chiplets over relatively long distances on the interposer itself. This capability is particularly pertinent in domains such as machine learning accelerators, where vast amounts of data need to be rapidly shuttled between different processing units. Our proposed silicon photonic interposers, detailed later in subsequent chapters, aim to address these communication challenges efficiently.

In contrast to metallic-based interconnects, silicon photonic communication offers several key advantages. Firstly, it allows for significantly higher bandwidth transmission over longer distances while consuming minimal power. This efficiency is particularly crucial in modern computing architectures, where the demand for data transfer rates continues to escalate. Furthermore, the integration of silicon photonic components onto existing semiconductor fabrication processes facilitates seamless adoption without requiring substantial infrastructure changes. These factors collectively position silicon photonic communication as a compelling solution for future-generation computing systems, especially in scenarios necessitating high-speed, low-latency data exchange among disparate chiplets.

Fig. 1.10 shows an example of data transmission between two chiplets that are placed on a silicon-photonic interposer. On the interposer, some modulators, filters, and photodiodes (PDs) are used to perform electro-optical and opto-electrical data conversions. In this dissertation, we consider microring resonator (MR) devices [13, 14], for modulators and filters, due to their area and power efficiency. We define a gateway as an electronic circuit on a chiplet, which controls the modulators (i.e., writer gateway in Fig. 1.10) and PDs (i.e., reader gateway in Fig. 1.10) on the interposer. Moreover, gateways receive/send data from/to the routers on the same chiplet. As shown in Fig. 1.10, optical signals with different wavelengths are generated in an off-chip laser source (green, red, and blue wavelengths). The optical signals are then coupled to the waveguide on the photonic interposer using an optical fiber and a grating coupler. At the writer gateway, MRs modulate electronic data on the optical signals and, at the reader gateway, each MR filters its corresponding optical signal to be detected by the PD. Note that each MR device is designed to resonate at (i.e., couple with) a specific wavelength. As a result, several wavelengths can transmit bits of data at the same time, over the same waveguide; this technique is called WDM.

## 1.5 NoC design challenges in 2.5D and 3D chip technologies

### 1.5.1 Network latency and traffic distribution

Network latency is one of the primary design challenges for NoCs. As the number of cores and processing elements in a chip increases, so does the amount of data traffic between them. This traffic must be managed efficiently to prevent delays and ensure that each core can access the data it needs in a timely manner.

### 1.5.2 Fault tolerance

Faults in NoC architectures are categorized into three main types based on their duration and impact: permanent faults, transient faults, and periodic faults. Permanent faults persist in the system until repaired, while transient faults occur under specific conditions and last for a limited period. Periodic faults, such as thermal faults, recur at regular intervals and may render the affected component temporarily unusable until rectified. These faults can affect the performance and reliability of NoC systems, which are susceptible to various failure mechanisms influenced by environmental conditions and process variations.

Radiation is a significant source of faults, originating from cosmic neutrons and alpha particles, which can induce Single Event Upsets (SEUs) or Single Event Transients (SETs) in memory elements or logic gates. Electrostatic discharge can break electrical bonds in electronic components, leading to dielectric, PN junction, or interconnect breakdowns, exacerbated by increased heat generation and leakage current. Aging effects, accelerated by heat, can degrade CMOS hardware performance and increase failure rates, particularly in interconnects. Electrical migration, driven by high current density, causes thinning of metal contacts over time, leading to timing failures or circuit malfunction. Heat-induced faults, especially prevalent in submicron technologies and three-dimensional on-chip networks, result from temperature-induced variations in transistor characteristics, affecting speed, power consumption, and reliability.

To cope with faults, three main methods are employed: fault avoidance, fault hiding, and fault tolerance. Fault avoidance aims to prevent fault progression through testing, and quality control

methods. Fault hiding reduces the severity or number of fault effects, often by incorporating redundancy in the system. Fault tolerance enables the system to continue operating despite faults, achieved through fault masking, detection and relocation, or routing algorithms that minimize exposure to faults. These coping methods are essential for ensuring the reliable performance of NoC architectures in the face of potential faults and failures.

Fault-Tolerant Routing algorithms ensure network reliability in the face of faults, selecting alternative paths to avoid disruptions caused by faults. These algorithms employ techniques such as fault regions and turn models to mitigate the impact of faults on network performance and reliability.

### 1.5.3 Hardware security

Hardware security focuses on protecting computing systems from physical attacks and vulnerabilities in the underlying hardware components. In this disseration we specifically focus on the security of interposers in 2.5D chiplet systems. This is crucial because the interposer is the central communication hub for all chiplets, and if compromised, could bring the entire system down. Security is especially important in chiplet systems compared to traditional monolithic chips due to several reasons. First, chiplet systems often integrate components from various sources, some of which might be untrusted. These untrusted chiplets could potentially launch Denial-of-Service (DoS) attacks by flooding the interposer with traffic, making it unavailable for legitimate communication. Second, chiplet systems rely on vertical links for communication between chiplets stacked on top of the interposer. These vertical links are expensive and critical for system performance, making them prime targets for attackers.

### 1.5.4 Energy efficiency

Energy efficiency is another critical concern for chip designers, especially for NoCs. As the number of cores on a chip increases, so does the power required to transmit data between them. NoC designers must carefully consider energy consumption when designing communication architectures to ensure optimal performance without excessive power demands. Energy efficiency

is paramount in 2.5D chiplet systems. These systems integrate multiple chiplets to overcome the limitations of large-scale chip fabrication. While this approach offers benefits, it also leads to a higher traffic load on the interposer network. To handle this increased traffic, solutions like silicon photonic communication have been proposed for high-bandwidth inter-chiplet communication. However, silicon photonics, despite its advantages, suffers from high power consumption in interposer networks. Therefore, designing energy-efficient 2.5D chiplet systems is critical to minimize power usage and ensure system reliability, even with the increased traffic demands.

### 1.5.5 Scalability

Scalability is another crucial aspect of NoC design. As the number of cores on a chip, and potentially the number of chiplets in a system, continues to increase in the future, NoCs must be able to adapt and handle this growth. Designers employ various techniques to achieve scalability. One approach is using hierarchical topologies, where smaller networks are combined to form larger ones, such as integrating 2.5D and 3D networks. however, dynamic routing algorithms and reconfigurable architectures that can adjust to changes in the network's configuration are necessary to be designed. Additionally, NoC virtualization allows combining multiple NoCs into a single virtual network, enabling scaling across multiple chips or even entire systems.

## 1.6 Dissertation motivation

The motivation for this dissertation stems from the urgent need to tackle key challenges in NoC architectures, specifically focusing on 3D and 2.5D topologies and interposer designs, while addressing latency, energy efficiency, fault tolerance, and security concerns. This research endeavors to overcome the limitations of 3D NoCs, including high fabrication costs and reliability issues related to TSV technology, by introducing an adaptive congestion- and energy-aware elevator-selection algorithm to mitigate latency and enhance energy efficiency. Furthermore, the dissertation aims to enhance the reliability and reduce latency in 2.5D chiplet systems through the development of fault-tolerant routing algorithms. The increasing significance of heterogeneous 2.5D

integration, facilitating seamless chiplet integration to reduce design time and costs, underscores the need for ensuring the security and reliability of the Network-on-Interposer (NoI). Addressing this imperative, the dissertation also seeks to develop a secure routing algorithm to safeguard chiplet systems against Denial-of-Service (DDoS) attacks, an aspect yet to be adequately addressed in current research. Additionally, the dissertation aims to contribute novel solutions to further enhance the efficiency of chiplet systems, considering silicon photonic NoIs. The proposed photonic NoI with PCM for energy-efficient interposer communication, aims to mitigate power consumption in interposer-based photonic networks, providing lower latency, reduced power consumption, and enhanced energy minimization compared to existing photonic networks. Furthermore, 2.5D accelerators requires efficient interposer networks for low-latency and energy-efficient communication, facilitating parallel Multiply and Accumulation (MAC) operations.

## 1.7 Dissertation outline

The dissertation explores various aspects of interconnect architectures for emerging 3D and 2.5D topologies. Chapter 2 presents an adaptive congestion-and-energy-aware elevator selections framework for Partially Connected 3D NoCs (PC-3DNoCs), proposing an algorithm for efficient elevator selection. Chapter 3 delves into a reliable and deadlock-free routing approach for 2.5D Chiplet-based Interposer Networks, introducing a routing algorithm to ensure reliable communication in active interposers. In Chapter 4, a reconfigurable Silicon-Photonic 2.5D Chiplet Network with Phase Change Materials (PCMs) is outlined, emphasizing energy-efficient communication through the ReSiPI architecture. Chapter 5 extends the discussion to machine learning accelerators in 2.5D chiplet platforms, leveraging ReSiPI-based Silicon Photonic Interposers for improved performance. Chapter 6 introduces a non-blocking switch-based silicon photonic interposer network for 2.5D machine learning accelerators, showcasing the SwInt architecture for efficient data transfer. Chapter 7 introduces a multi-objective routing framework designed to secure chiplet systems against distributed denial of service (DDoS) attacks. Through these chapters, the dissertation aims to provide comprehensive insights into advanced interconnect technologies and their applica-

tions in next-generation chiplet-based systems while outlining avenues for future research. Finally, Chapter 8 summarizes this dissertation and provides some suggestions for future work.

# Chapter 2

# Adaptive Congestion-and-Energy-Aware Elevator Selection for Partially Connected 3D NoCs

Unlike conventional 2D Networks-on-Chip (NoCs), 3D NoCs offer a scalable and energy efficient on-chip communication. Vertical die stacking of 3D NoCs is enabled using inter-layer Through-Silicon Via (TSV) links. However, TSV technology suffers from low reliability and high fabrication costs. To mitigate these costs, Partially Connected 3D NoCs (PC-3DNoCs), which use fewer vertical TSV links, have been introduced. However, with fewer vertical links (a.k.a. elevators), elevator-less routers will have to send their traffic to nearby elevators for inter-layer traffic, increasing the traffic load and congestion at these elevators and potentially reducing performance. Therefore, it is important that elevator-less routers choose elevators that balance the traffic load among the available elevators. To address this problem, this chapter presents an adaptive congestion- and energy-aware elevator-selection algorithm, called AdEle+. AdEle+ employs an offline multi-objective simulated-annealing-based optimization to find good elevator subsets for routers. In addition, during high traffic loads, AdEle+ uses an adaptive and online elevator selection algorithm to select an elevator from the elevator subset to dynamically manage traffic congestion on elevators. Moreover, in low congestion circumstances, AdEle+ switches to a minimal distance selection to improve energy efficiency. Compared to state-of-the-art selection algorithms under various PC-3DNoC configurations and traffic patterns, AdEle+ reduces the average latency by 9.5% on average and up to 11.2% while reducing the hardware overhead by 10.1% due to its efficient online selection in the routers.

## 2.1 Introduction

To create vertical links in TSV-based 3D NoCs, multiple TSVs are grouped into a bundle. However, due to the large TSV interconnect pitch and keep-out-zone requirements [15], these TSV

bundles result in large area overhead. Moreover, TSVs are particularly susceptible to electromigration and capacitive crosstalk-induced issues [16,17]. Therefore, it is very costly to have TSV-based vertical links at every router [6,7]. To address these challenges, Partially Connected 3D NoCs (PC-3DNoCs) with TSV-based vertical links at only some routers have been proposed [1, 7, 18]. In addition to reduced fabrication costs, the PC-3DNoC paradigm can be utilized to handle missing vertical links due to TSV-based faults [19] and to improve manufacturing yields.

Since PC-3DNoCs remove some of the vertical links (a.k.a. elevators), the remaining elevators must be shared among multiple routers. A well-known routing solution in PC-3DNoCs, Elevator-First routing [1], naïvely select the nearest elevator without considering the overall network traffic, potentially creating traffic hotspots at certain elevators and increasing the network latency [7]. To mitigate the effects of these traffic hotspots, we can balance the traffic across all elevators using an adaptive routing technique that selects elevators based on elevator utilization. For example, Congestion-aware Dynamic elevator Assignment (CDA) [20] uses global traffic information to improve the elevator load distribution during runtime.

Elevator-selection algorithms in PC-3DNoCs can be broadly classified into design-time and runtime approaches. For example, in [1, 21], each router is assigned one elevator for all vertical traffic at design time and to optimize network performance (e.g., the network latency). On the other hand, online approaches like [22] monitor traffic and select elevators during runtime to help balance the elevator load. Since design-time approaches do most of the calculations offline, only simple look-up tables are required, oftentimes resulting in simpler and faster implementations compared to an online approach. However, these offline approaches [1,21,23–25] rely only on the traffic information available during design time and cannot adapt to new runtime traffic scenarios or changing system conditions, e.g., faults. Although online solutions can help adjust to these changes, they can impose large overhead. For example, in CDA [20, 22], gathering global traffic information from distant routers and determining the selection at runtime impose significant hardware and latency overhead.

This chapter addresses the elevator-selection problem in PC-3DNoC routing techniques by developing, a novel, congestion- and energy-aware adaptive elevator-selection scheme called AdEle+. The main contributions of AdEle+ are summarized in the following:

- We propose AdEle+, a two-stage elevator-selection approach that takes advantage of both design-time and runtime benefits: AdEle+ integrates a design-time elevator-set optimization and a runtime elevator-selection policy to balance traffic with minimal overhead.
- We utilize a Multi-Objective Simulated-Annealing-based optimization algorithm (AMOSA [26]) offiline for the design-time elevator-set optimization. AMOSA chooses, for each router, an optimized subset of elevators that will be used during runtime selection to simplify the online elevator selection.
- We utilize a Multi-Objective Simulated-Annealing-based optimization algorithm (AMOSA [26]) offiline for the design-time elevator-set optimization. AMOSA chooses, for each router, an optimized subset of elevators that will be used during runtime selection to simplify the online elevator selection.
- We develop a low-cost runtime elevator selection that has two modes: one for high traffic loads using local traffic information and a simple modified round-robin technique to improve the elevator load; and another for low traffic loads using a distance-based elevator selection that considers the elevator distance to *both* source and destination.
- An algorithm is presented to find the optimized threshold for switching between the different runtime elevator selection modes.

Our simulation results, under different synthetic and real-application traffics with various PC-3DNoC configurations, show the promise of AdEle+ compared to state-of-the-art. For instance, AdEle+ improves the network latency by 10%, on average, and by up to 13.9%. AdEle+ also reduces the energy consumption in low traffic scenarios. Moreover, due to the local traffic monitoring of AdEle+, hardware overhead is reduced, compared to CDA selection, in which global traffic information is shared among routers.

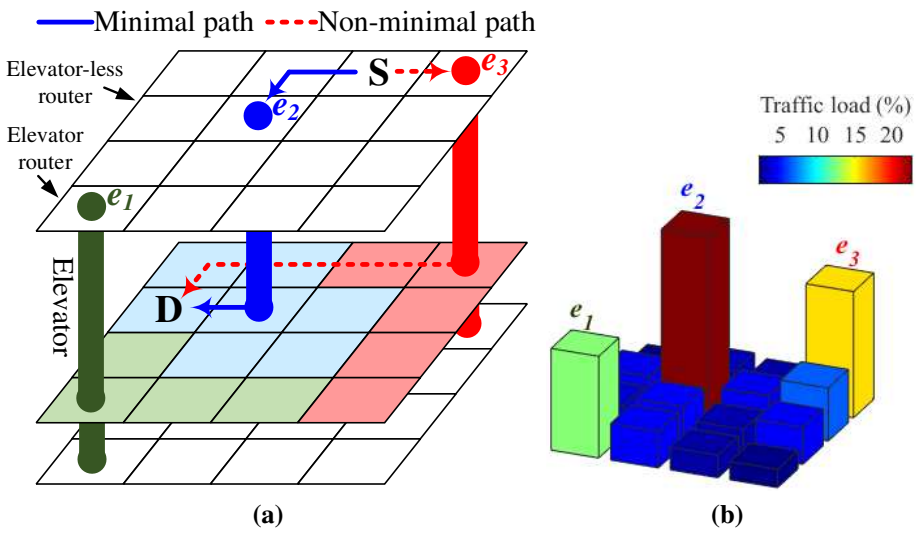



**Figure 2.1:** (a) An example PC-3DNoC with three elevators ($e_1$, $e_2$, and $e_3$). The routing path from S to D based on Elevator-First algorithm [1] (dotted-red line) and the minimal path (blue-solid line) are shown. The middle-layer routers are colored based on their Elevator-First selected elevator. (b) Traffic load on each router in the middle layer: the $e_2$ elevator is highly congested because of the inefficient elevator selection in Elevator-First algorithm (7 out of 16 routers use this elevator router).

The rest of the chapter is organized as follows. We present the background and review some prior related work on PC-3DNoCs in Section 2.2. Section 2.3 discusses the elevator-selection problem and its complexity in PC-3DNoCs. Moreover, it details our proposed technique (AdEle+) and its implementation. In Section 2.4, we explore the parameters of AdEle+ to optimize the elevator selection. In addition, Section 2.4 presents our simulation results including latency, energy, power, and hardware. Finally, Section 2.5 concludes the chapter.

## 2.2 Background and Related Work

This section discusses PC-3DNoCs and related challenges in routing, elevator placement, and elevator selection.

### 2.2.1 Partially Connected 3D Networks-on-Chip

The total number of TSVs highly affects the overall cost of 3D chips [27, 28]. Since each elevator link includes tens to hundreds of TSVs (e.g., $2 \times 128$ bits in [29]), limiting elevator links to only some routers—a Partially Connected 3D NoC (PC-3DNoC)—can help significantly

improve the fabrication cost [1, 6, 18, 30]. For example, a PC-3DNoC with elevators at 25% of the routers significantly improves the chip fabrication cost with only 18.7% overhead in the network latency, compared to a fully connected 3D NoC [31]. Moreover, PC-3DNoC schemes can be better extended to deal with TSV faults as they are designed to work in a partially connected network with missing TSV links [19]. Note that our work assumes a given PC-3DNoC topology and thus can handle manufacturing TSV faults. Considering runtime TSV faults are beyond the scope of this chapter.

As shown in Fig. 2.1a, PC-3DNoCs consist of two types of routers: elevator-less routers with five ports (east, south, west, north, and local), and elevator routers that include two additional ports (up and down) for a total of seven ports. As elevator-less routers are not directly connected to the other layers, their inter-layer packets must first route to an elevator router.

### 2.2.2 Elevator Placement and Routing Algorithms

Similar to elevator selection during routing in PC-3DNoCs (see Section 2.3), elevator placement also plays an important role in the network performance. In [31], the authors proposed an elevator placement that minimizes the hop count from all routers to nearby elevators. It is assumed that elevator-less routers utilize their closest elevator in the layer. Considering this assumption, an optimized placement is found to minimize the average router to elevator distance. However, in runtime network operation, different elevators might be selected for elevator-less routers, which is not consistent with the assumption in the placement optimization. To this end, [20] uses a Genetic Algorithm (GA) to place elevators while considering elevator selection to minimize the average distance and load variance. However, these approaches will likely end up with a non-uniform elevator placement, after manufacturing faults, which can lead to non-uniform elevator usage. Even with uniform elevator placement, traffic is typically non-uniform leading to imbalanced elevator utilization. Fortunately, an adaptive elevator selection algorithm can compensate for these effects and balance the traffic over the elevators.

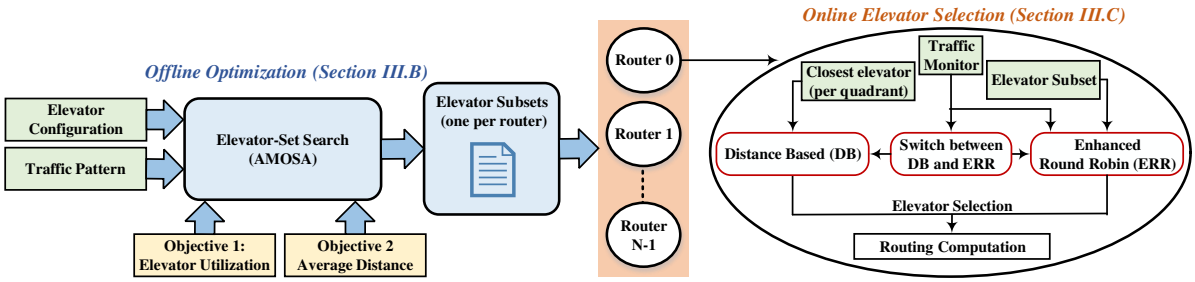



**Figure 2.2:** An overview of our proposed elevator-selection algorithm: AdEle+.

Many routing algorithms have been proposed for PC-3DNoCs [1,6,18,19,24,30,32–34]. They are mainly different in their approach to guarantee deadlock freedom. For instance, in Elevator-First [1], the most popular deadlock-free routing for PC-3DNoCs, upward-bound packets are routed in one virtual channel while downward-bound packets are routed in another virtual channel to break the cyclic dependency and maintain deadlock freedom. However, Elevator-first is a deterministic routing algorithm and it suffers from a low adaptation in path selection. To this end, LEAD [24] offers an adaptive routing algorithm while guaranteeing deadlock freedom by defining some subnetworks. However, all of these routing algorithms employ a simple selection, where the closest elevator to the source router is used for inter-layer routing. Although simple, this selection results in an unbalanced traffic load over elevators and, therefore, unnecessary network congestion and performance loss.

### 2.2.3 Elevator Selection Algorithms

In PC-3DNoCs, the elevator selection process can highly impact the traffic distribution across the elevators and consequently the performance of the entire network. Unfortunately, elevator selection in PC-3DNoCs has been barely studied in related work. In conventional routing algorithms, usually the closest elevator is selected to route inter-layer packets [1, 32, 33]. However, this selection ignores the elevators' load distribution and can lead to network congestion. This can be especially harmful for PC-3DNoCs with non-uniform elevator placements, small number of elevators, or non-uniform traffic distribution. Adaptive elevator-selection techniques have been proposed [6, 18, 30], but they mainly focus on designing fault-tolerant approaches to handle eleva-

tor failures. These strategies select the closest non-faulty elevator to the source without considering the elevator's load and congestion.

To improve the traffic distribution in PC-3DNoCs, [21] proposed an offline optimized elevator-selection algorithm using the Tabu search algorithm to distribute the load over elevator links and reduce network latency. However, this approach does not consider network energy and cannot capture the dynamics of the runtime network traffic due to its offline nature. In [24], a simple online selection strategy was presented where routers select one elevator randomly. This random selection improves the elevator traffic distribution compared to the closest elevator selection. However, it increases the average hop count and energy consumption as some packets will have to travel much farther to distant elevators. In [20] and [22], an online Congestion-aware Dynamic elevator Assignment (CDA) was proposed. CDA selects the elevator that minimizes the sum of the delay and the buffer utilization of the routers between the packet's source and the elevator. However, CDA requires online global information of the routers' delay and buffer utilization, sharing of which imposes high overhead.

Considering the aforementioned efforts, an efficient elevator-selection solution is yet to be realized for PC-3DNoCs. In the rest of this chapter, we propose an Adaptive congestion- and energy-aware Elevator-selection algorithm (AdEle) to optimize a subset of elevators for each router, then leverages such subsets in runtime to dynamically avoid congested elevators and improve traffic while relying only on local information. Employing a set of elevators instead of one elevator per router enables the online selection to adapt to new traffic patterns, hence improving the network performance. Moreover, we realize a low-cost Distance-Based (DB) elevator selection for the low congestion scenarios to improve the energy efficiency. In addition, an algorithm is proposed for switching between the traffic-aware and distance-based selections to enable AdEle+ to account for a dynamic traffic load. We explore design parameters of AdEle+ and optimize them to significantly improve the performance. We also show the effectiveness of the distance-based elevator selection, under low traffic load scenarios, and the dynamic switching between the distance-based and congestion aware selections.

## 2.3 Proposed Elevator-Selection Algorithm: AdEle+

This section discusses the main challenges for elevator selection in PC-3DNoCs and details our proposed adaptive congestion- and energy-aware elevator-selection algorithm, AdEle+. As an overview, Fig. 2.2 shows the building blocks of the proposed algorithm: AdEle+ uses an offline multi-objective simulated-annealing-based algorithm (AMOSA) to find an optimal subset of elevators for each router and an online elevator-selection algorithm to then select the best elevator from the subset in the presence of runtime traffic.

### 2.3.1 Motivation: Routing in PC-3DNoCs

In PC-3DNoCs, the routing process requires three main steps because of the irregular topology: 1) selecting a vertical link (elevator) for each packet in the source router and then routing the packet to that elevator on the source layer; 2) vertically routing the packet to the destination layer; and 3) routing the packet from the elevator to the destination node on the destination layer. In PC-3DNoCs, the elevator selection (the first step) is critical as elevators quickly become the network bottleneck due to the smaller number of elevators. As we will show, AdEle+ optimizes the elevator selection in the first step to balance the traffic over the elevators, thereby reducing network traffic hot-spots.

Fig. 2.1a shows an example of a PC-3DNoC with three elevators ($e_1$–$e_3$). Router tiles are colored with the elevator's color they would use under the Elevator-First policy (i.e., the closest elevator to the source router is selected) [1, 6, 18], i.e., four routers will use the green ($e_1$) elevator, seven will use the blue ($e_2$) elevator, and five will use the red ($e_3$) elevator. Unfortunately, such an unbalanced elevator selection can put severe traffic pressure on certain elevators ($e_2$ in this example). Ideally, some of the load on $e_2$ should be assigned to $e_1$ or $e_3$, making $e_2$ less congested. Fig. 2.1b demonstrates the utilization of the middle-layer routers with Elevator-First selection policy under uniform traffic. This confirms that $e_2$ is highly congested due to the unbalanced elevator selection. In terms of energy efficiency in low traffic loads, the best elevator selection is on the minimal path between the source and destination. However, for the path between S and D in Fig. 2.1a,

policies that ignore the location of the destination during the elevator selection, e.g., Elevator-First, may end up with longer paths (red-dotted line) than the minimal path (blue-solid line).

AdEle+ considers both the elevator utilization and energy efficiency to select elevators with distributed traffic load and minimize source-destination distance. To the best of our knowledge, AdEle+ is the first congestion- and energy-aware elevator-selection algorithm in PC-3DNoCs that includes dynamic elevator selection to accommodate dynamic traffic behavior while relying only on local router information.

### 2.3.2 Offline Optimization in AdEle+

To find the optimal subset of elevators for each router, AdEle+ performs an offline optimization to distribute the expected traffic load across all elevators and minimize the average inter-node (source to destination) distance. To do this, we first define two optimization objectives: 1) elevator-utilization variance to improve the traffic load distribution, and 2) average inter-node distance to minimize the energy consumption. Leveraging these objective functions, we will use a multi-objective simulated-annealing-based algorithm (AMOSA [26]) to find the optimal elevator subsets.

#### Objective 1—Elevator Utilization

To balance the traffic on the elevators, AdEle+ attempts to minimize the elevator-utilization variance. As discussed above, it is important to evenly distribute the traffic over elevators to avoid highly congested elevators. To calculate the utilization variance, let us consider an $N$-node or $N$-router network with a set of elevators $\mathcal{E} = \{e_1, e_2, \ldots, e_E\}$, where $E$ is the total number of elevators. Moreover, assume that in runtime, each router $i$ can select its elevator from a subset $A_i \subseteq \mathcal{E}$. For the time being, let us assume that each router selects each elevator from its elevator subset ($A_i$) uniformly (e.g., using a round-robin policy). Therefore, the utilization of elevator $e$ ($U_e$) is:

$$U_e = \sum_{i=1}^{N} \frac{1}{|A_i|} \sum_{j=1}^{N} f_{ij} \cdot P_{ije}, \tag{2.1}$$

where $f_{ij}$ is the communication frequency (i.e., traffic) between routers $i$ and $j$, and $P_{ije} = 1$ when the routing between routers $i$ and $j$ uses the elevator $e$, otherwise $P_{ije} = 0$. Leveraging (2.1), the average traffic over all the elevators ($\mu$) can be defined as:

$$\mu = \frac{1}{E} \sum_{i=1}^{E} U_i. \tag{2.2}$$

Using (2.1) and (2.2), elevator-utilization variance ($\sigma^2$) is:

$$\sigma^2 = \frac{1}{E} \sum_{i=1}^{E} (U_i - \mu)^2. \tag{2.3}$$

Minimizing the elevator-utilization variance will result in a better distribution of traffic load on the elevators and eventually lower the network latency by reducing network congestion and traffic hot-spots.

**Objective 2—Average Inter-Layer-Node Distance**

To improve network energy efficiency, AdEle+ attempts to minimize the average inter-layer-node distance when selecting the elevator subsets. The distance ($D_{ij}^e$) between inter-layer nodes $i$ and $j$ using elevator $e$ can be defined as:

$$D_{ij}^e = \begin{cases} 0, & i \text{ and } j \text{ are on the same layer} \\ d_{se} + d_e + d_{ed}, & \text{otherwise.} \end{cases} \tag{2.4}$$

Here, $d_{se}$, $d_e$, and $d_{ed}$ are the Manhattan distances between the source and elevator, the source and destination layers (through the elevator), and the elevator and destination, respectively. Based on (2.4), the average inter-layer-node distance (AD) in an $L$-layer network can be calculated as:

$$AD = \frac{1}{N \cdot (\frac{L-1}{L} \cdot N)} \sum_{i=1}^{N} \frac{1}{|A_i|} \sum_{e \in A_i} \sum_{j=1}^{N} D_{ij}^e. \tag{2.5}$$

**Multi-Objective Optimization**

We use a multi-objective simulated annealing-based optimization algorithm (AMOSA [26]) to find a set of optimal elevator subsets for all routers in the network ($\mathcal{A} = \{A_1, \ldots, A_N\}$) while minimizing the objective functions (2.3) and (2.5). Similar to Simulated Annealing (SA) [35], AMOSA performs a broad exploration at the start of the search process and gradually chooses more greedy moves to select the best solutions to help approximate the global optima in a large solution space. Differently than SA, AMOSA outputs a *set* of solutions that lie on the Pareto front of the optimization objectives. In AdEle+, AMOSA provides solutions with different tradeoffs in terms of elevator-utilization variance and average inter-layer-node distance. From these solutions, a designer can choose the appropriate tradeoff (see Fig. 2.6). Selection of solutions are discussed in detail in Section 4.

### 2.3.3 Adaptive Online Elevator Selection

Here, we discuss how a router $i$ can efficiently select an elevator during runtime from its elevator subset ($A_i$) identified in the previous subsection. A common method is to use a simple Round Robin (RR) approach where each elevator is selected equally in a sequential order. However, solutions such as RR do not consider traffic patterns and congestion that occur during runtime. As we are interested in an even distribution of traffic load over all elevators to improve traffic congestion during runtime, we propose an Enhanced RR (ERR) algorithm for selecting elevators. Our proposed ERR approach includes a probability of skipping $P_{Sik}$ a congested elevator $k$ for each router $i$. $P_{Sik}$ is adjusted based on the average latency imposed by the elevator $k$, i.e., higher latencies seen using elevator $k$ increases the probability of skipping it in the future. Accordingly, AdEle+ can adaptively manage dynamic traffic loads and congestion.

To find $P_{Sik}$, AdEle+ first estimates the cost of selecting a particular elevator by considering the time between when the first flit (the header flit) and when the last flit (the tail flit) leave the

source router. The latency ($T_k$) imposed by selecting elevator $k$ is:

$$T_k = \frac{t_{tail} - t_{head} - l_p}{l_p}, \tag{2.6}$$

where $t_{tail}$ and $t_{head}$ denote the time when the tail flit and the header flit leave the source router, respectively. Also, $l_p$ is the length of the packet. After each packet leaves a router $i$, the elevator-selection cost ($C_{ik}$) is updated:

$$C_{ik} \leftarrow (a \times T_k) + ((1 - a) \times C_{ik}), \quad 0 \leq a \leq 1 \tag{2.7}$$

where $a$ is used to adjust the impact of the new cost versus the old one. This allows us to keep some past information about the congestion at elevator $k$ while updating it with the most recent transmission. We have experimentally found that $a = 0.2$ produces good results in AdEle+.

Leveraging (2.7), AdEle+ can estimate the latency cost at the source router with *only local information*, while the state-of-the-art [20, 22] requires global information with high overhead. With wormhole switching, any blocking in an elevator can be propagated along the path from the elevator to the source router. Since we expect the elevators to be the predominant source of congestion in these PC-3DNoCs, blocking at a source router can be interpreted as blocking in the elevator. Note that incorporating global-network information into AdEle+ would improve the selection policy but will impose high hardware area, energy consumption, and latency costs.

Using (2.7), we define router $i$'s relative cost when selecting elevator $k$ among other elevators in its elevator set $A_i$:

$$C_{ik}^{rel} = \frac{C_{ik}}{\sum_{p=1}^{|A_i|} C_{ip}}. \tag{2.8}$$

Where $C_{ip}$, is the cost of elevator $p$ in router $i$ (See (2.7)). Using the relative cost of a particular elevator selection, the possibility of skipping that elevator in our ERR approach is:

$$P_{Sik} = \begin{cases} 1-\xi, & \text{if } C_{ik}^{rel} \geq \frac{2}{|A_i|} \\ |A_i| \cdot (C_{ik}^{rel} - \frac{1}{|A_i|}) \cdot (1-\xi), & \text{if } \frac{2}{|A_i|} > C_{ik}^{rel} \geq \frac{1}{|A_i|} \\ 0, & \text{otherwise.} \end{cases} \tag{2.9}$$

Here, $\xi$ is included to allow a small fraction of packets to be sent to highly congested routers so that the cost can be updated to reflect the current state ($\xi = 0.05$ in our experiments). To clarify the use of $\xi$, let us assume that $\xi = 0$. This would allow a value of 1 for $P_{Sik}$ when there is high congestion. In this case, elevator $k$ will not be selected in the ERR sequence at all and have no chance to update its elevator-selection cost ($C_k$). Even if the congestion at elevator $k$ is resolved, we would constantly skip elevator $k$ unless the cost for the other elevators rise and reduces $C_{ik}^{rel}$ (see (2.8)). To address such an update failure, $\xi$ allows every elevator to be selected with a low probability regardless of their $P_{Sik}$, so the cost function has a chance for updating. In (2.9), the expected load—i.e. when the load is evenly distributed over elevators—is $\frac{1}{|A_i|}$. Therefore, when the elevator load is below the expected load, the skip possibility is zero (the third line in (2.9)). On the other hand, if the load of an elevator is twice of the expected load ($\frac{2}{|A_i|}$) or more, the elevator is skipped with a high possibility (the first line in (2.9)). For the loads between $\frac{1}{|A_i|}$ and $\frac{2}{|A_i|}$, the elevator is skipped with respect to the extra load (the second line in (2.9)). The proposed ERR is described in Algorithm 1.

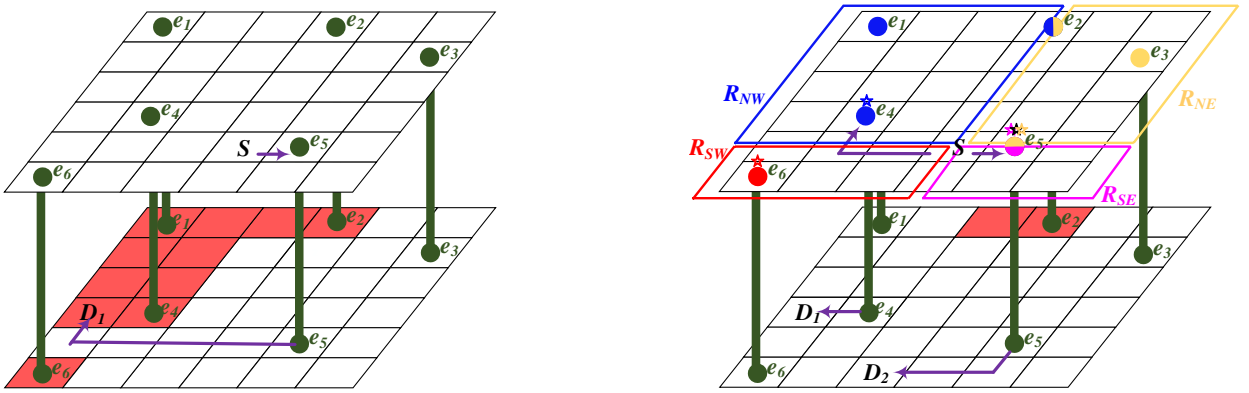
### 2.3.4 Elevator Selection in AdEle+ under Low Traffic

AdEle+ employs ERR elevator selection to balance elevator utilization and improve network congestion. However, under low traffic loads, congestion is not a concern and employing ERR can increase both the latency and the energy by taking non-minimal paths. Therefore, we propose AdEle+ to use a distance-based routing when the congestion at the elevators is expected to be low.

**Algorithm 1** Enhanced Round Robin in AdEle+

$R$: Round robin counter

**for** all elevators $k$ in elevator set of router $i$ **do**
  $C_{ik}^{rel} \leftarrow$ Calculate the relative cost (Equation 2.8)
  $P_{Sik} \leftarrow$ Calculate skip probability (Equation 2.9)
**while** elevator is not selected **do**
  Skip elevator $R$ with probability $P_{SiR}$
  Go to next selection ($R$++)
Update $C_{iR}$ after sending tail flit




**(a)** Closest elevator selection example

**(b)** AdEle+'s Distance-Based (DB) elevator selection

**Figure 2.3:** (a) Closest elevator selection example shows that 11 out of 36 destination routers in the bottom layer use non-minimal paths from source $S$ (red destinations receive packets non-minimally). (b) Distance-Based (DB) elevator selection in AdEle+ shows example quadrants ($R_{NW}$, $R_{NE}$, $R_{SW}$, $R_{SE}$) based on $S$. Each elevator is color-coded based on their quadrant (with some elevators with two colors), and the regional closest elevator ($RCE$) for each region is marked with a color-coded star. Similarly, the closest elevator ($CE$) is marked with a black star ($e_5$). Our DB elevator selection will consider the paths through the $RCE$ in the destination quadrant and the overall closest elevator. Our DB elevator selection only routes 2 out of 36 destinations non-minimally from source $S$.

We also proposed a mechanism, which will be elaborated in the next subsection, to dynamically switch between the ERR and distance-based selections.

Unlike a regular mesh topology, finding the minimal path in PC-3DNoCs requires knowledge of the source, destination, and nearby elevators' locations. The minimal path is not necessarily through the source's nearest elevator. Let us consider an example shown in Fig. 2.3a. Here, the conventional closest elevator selection (i.e., Elevator-First selection) would choose $e_5$ from $S$ to $D_1$ instead of using the shortest path through $e_4$. To find the minimal path in PC-3DNoCs, we can apply an exhaustive search across all the elevators and save the results in each router. However, this

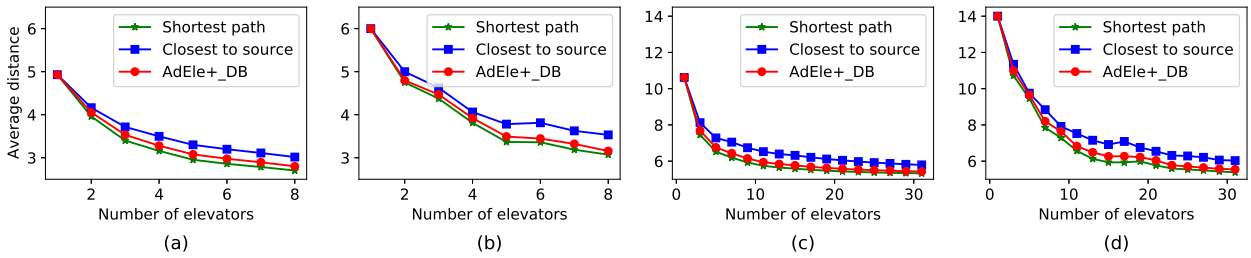



**Figure 2.4:** Comparison of different elevator selection policies in terms of average distance under different network sizes and number of elevators. For each network size and number of elevators, 100 random elevator-placement patterns are evaluated. Here, we report both the average (average case) and the maximum (worst case) of all the 100 random elevator patterns' average distance. (a) $4 \times 4$ layer size - average case; (b) $4 \times 4$ layer size - worst case; (c) $8 \times 8$ layer size - average case; and (d) $8 \times 8$ layer size - worst case.

is unscalable and imposes a rather large overhead. Instead, our efficient distance-based elevator selection takes advantage of the observation that the shortest path between many source-destination pairs goes through either 1) the closest elevator to the source, or 2) the closest elevator to the source in the same quadrant as the destination, assuming the source as the origin (0,0). We divide the source layer into 4 quadrant regions considering the source as the origin: northwest ($R_{NW}$), northeast ($R_{NE}$), southwest ($R_{SW}$), and southeast ($R_{SE}$) regions. In each of the 4 regions, we find one elevator as the closest elevator to the source in the quadrant. We define $CE_{NW}$, $CE_{NE}$, $CE_{SW}$, and $CE_{SE}$ as the closest elevator in $R_{NW}$, $R_{NE}$, $R_{SW}$, and $R_{SE}$ quadrants, respectively.

For example in Fig. 2.3b , we show the $R_{NW}$, $R_{NE}$, $R_{SW}$, and $R_{SE}$ quadrants using $S$ as the source. In this example, $R_{NW} = \{e_1, e_2, e_4\}$, $R_{NE} = \{e_2, e_3, e_5\}$, $R_{SW} = \{e_6\}$, and $R_{SE} = \{e_5\}$. Therefore, $CE_{NW} = e_4$, $CE_{NE} = e_5$, $CE_{SW} = e_6$, and $CE_{SE} = e_5$. Note that the elevators on the border of each quadrant belong to both quadrants for the purpose of our distance-based elevator selection. For example, $e_2$ belongs to both $R_{NW}$ and $R_{NE}$ quadrants and $e_5$ is in both $R_{NE}$ and $R_{SE}$ quadrants. However, in the case of $S$, $e_2$ is not the quadrant's closest elevator in neither $R_{NW}$ nor $R_{NE}$. Also, if there is no elevator in a quadrant, the closest elevator to the source ($CE$) is considered as that quadrant's regional closest elevator ($RCE$). For destination $D_1$, based on our observation earlier, we consider $e_5$ (closest to the source) and $e_4$ (closest in the destination quadrant), and select $e_4$ which is the shortest-path elevator. Similarly, for the path from $S$ to $D_2$, we consider $e_5$ and $e_6$ and choose $e_5$, again the shortest-path elevator. Although this finds the shortest-

path elevator in many situations, there are some cases that may not be minimal. For example, from $S$ to the red routers—two routers out of 36 routers in the bottom layer—distance-based AdEle+ finds elevators with two hops more than the shortest path. However, using the closest elevator is even worse as it will fail to find the shortest-path elevator for all the routers in red shown in Fig. 2.3a (11 out of 36 routers).

To analyze the benefit of the proposed approach, we evaluated the average inter-layer distance of our approach for many network configurations (100 random elevator patterns with layer sizes of $4 \times 4$ and $8 \times 8$). Fig. 2.4 shows that the proposed distance-based selection in AdEle+ is able to achieve an average distance that is always better than the closest elevator and very close to the shortest-path selection. Fig. 2.4(a) and Fig. 2.4(c) show the average across 100 random elevator patterns' average distances, while Fig. 2.4(b) and Fig. 2.4(d) show the worst case among the 100 random elevator patterns' average distances. In all scenarios, DB is able to nearly achieve the same distance as the shortest path selection at every number of elevators. In these experiments, the average rate of non-minimal routing in DB AdEle+ was smaller than 4.1% and 7.5% for $4 \times 4$ and $8 \times 8$, which resulted in an increase of 3.2% and 2.7% in the average distance for $4 \times 4$ and $8 \times 8$, respectively. Due to its simplicity, the distance-based elevator selection only needs to store five different elevators regardless of the size of the network or the number of elevators: one for each quadrant and the closest elevator. Therefore, this approach is a very scalable and has a low overhead to help improve network latency and energy efficiency. Algorithm 2 details the distance-based elevator selection in AdEle+.

### 2.3.5 Adaptive Selection Between Distance-Based or Enhanced Round-Robin

To determine when to switch between the Distance-Based (DB) elevator selection and ERR, AdEle+ leverages the congestion information already gathered by the relative cost $C^{rel}$ of elevators defined in (2.8). When $C^{rel}$ is below a threshold ($tr_{DB}$) for all the elevators, AdEle+ will switch to the distance-based elevator selection and minimize hop counts. Otherwise, ERR is used to balance network congestion.

**Algorithm 2** AdEle+'s distance-based elevator selection

$RCE$: Regional closest elevator
$CE$: Closest elevator to source
$CE_{NE}$, $CE_{SE}$, $CE_{NW}$ and $CE_{SW}$: Closest elevator in north-east, south-east, north-west and south-west quadrant of source router

**if** $S.x \geq D.x$ and $S.y \geq D.y$ **then**
  $RCE \leftarrow CE_{NE}$
**else if** $S.x \geq D.x$ and $S.y \leq D.y$ **then**
  $RCE \leftarrow CE_{SE}$
**else if** $S.x \leq D.x$ and $S.y \geq D.y$ **then**
  $RCE \leftarrow CE_{NW}$
**else**
  $RCE \leftarrow CE_{SW}$
Select $CE$ or $RCE$ based on source-destination distance

The proposed framework to find $tr_{DB}$ is shown in Fig. 2.5. In order to find the point at which ERR starts to outperform the distance-based selection ($tr_{DB}$), we perform network simulations with uniform traffic under increasing injection loads. At a high enough injection load, congestion starts to build up at the elevators and ERR begins to outperform the distance-based elevator selection. Then, we find the injection load ($P_{ins}$) when distance-based and ERR have the same latency and use the average cost of elevators ($C$) at the injection load ($P_{ins}$) to find $tr_{DB}$:

$$tr_{DB} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|A_i|} \sum_{k \in A_i} C_{ik}. \tag{2.10}$$

Although we use different traffic patterns in runtime, we will show in the next section that the average cost extracted using uniform traffic is close to the optimal one under different traffic patterns. Also, note that the proposed framework models the NoC at design time (here we used our simulator to model it) and calculate $tr_{DB}$ to be used at design time. Therefore, there would not be any performance and area cost to estimate $tr_{DB}$ at runtime.

## 2.4 Experimental Results and Evaluations

In this section, we describe our simulation setup and present some parameter explorations in AdEle+ and its simulation results compared to the state-of-the-art elevator-selection algorithms.

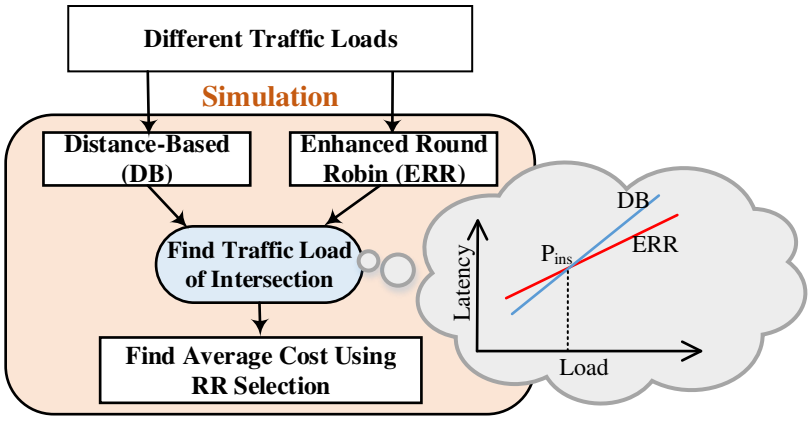


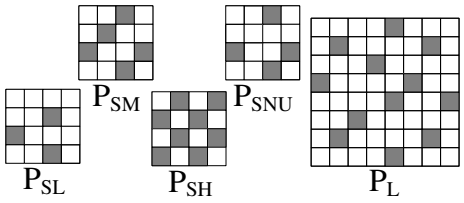

**Figure 2.5:** Proposed framework to find $tr_{DB}$. ERR and distance-based elevator selections are simulated under different injection loads. The average cost (2.8) of the load where latency of both ERR and distance-based selections are equal is used as the minimal threshold ($tr_{DB}$)

### 2.4.1 Simulation Setup

We compare the proposed AdEle+ with the state-of-the-art elevator-selection algorithms using Access Noxim [36], which is an extended version of Noxim [37], GEM5 [38] (for real-application traffic extraction), and Cadence Genus [39] (for hardware area analysis). We compare AdEle+ with two well-known elevator-selection algorithms: Elevator-First [1] and CDA [20]. Table 6.1 summarizes the simulation setup. We used Elevator-First [1] routing algorithm for deadlock freedom in our simulation (albeit, with a different elevator selection for CDA, AdEle, and AdEle+), although AdEle+ can be added to any other routing algorithms in PC-3DNoC.

In PC-3DNoCs, the number and location of elevators can be affected by different performance-cost trade-off considerations [7]. Therefore, AdEle+ is evaluated using different PC-3DNoC elevator-placement patterns to show that its efficacy is independent of any such patterns. Four elevator patterns are considered for a 4×4×4 network (see Table 6.1): $P_{SL}$, $P_{SM}$, and $P_{SH}$, which are patterns with low, medium, and high density of elevators, respectively; and a non-uniform pattern $P_{SNU}$, which is designed based on $P_{SM}$ where one of the elevators is faulty (i.e., one elevator is removed due to, for example, TSV manufacturing faults [17]). These patterns are optimized using Simulated Annealing (SA) to minimize the overall average distance. A large network elevator

**Table 2.1:** Simulation setup of AdEle+

| | |
|---|---|
| Simulator | Access Noxim [36]<br>(for latency & energy analyses) |
| Network size | 4×4×4 and 8×8×4 |
| Routing and VC selection<br>(w/o elevator selection) | Elevator-First [1]<br>(used to avoid deadlock) |
| Switching | Wormhole switching |
| Buffer depth | 4 flits |
| flit width | 32 bits |
| Packet size<br>(flits) | 10–30 (uniform) for synthetic<br>18 for real application |
| Traffic pattern | Uniform, Shuffle, and Real |
| PC-3DNoC Elevator Placements |  |

pattern $P_L$ (8×8×4) with optimized average distance is also considered in our evaluation to show the scalability of AdEle+.

AdEle+'s offline optimization (see Section 2.3.2) is implemented in Python to extract the elevator subsets for each router. These subsets are then added to the AdEle+ router implemented in Access Noxim simulator [36]. As it is hard to predict online traffic accurately at the design time, for the offline optimization we considered uniform traffic, the most pessimistic assumption (i.e., traffic is not known *a priori*), while the network evaluations are done using different synthetic and real-application traffic patterns. Our analyses will demonstrate that AdEle+ does not require runtime traffic in its offline optimization as its online selection policy will adjust to runtime traffic (see AdEle_RR versus AdEle+ in evaluations presented in Figs. 2.10 and 2.11) . However, if the traffic is known, AdEle+ can use the runtime traffic during elevator-subset selection to offer further latency and energy improvements.

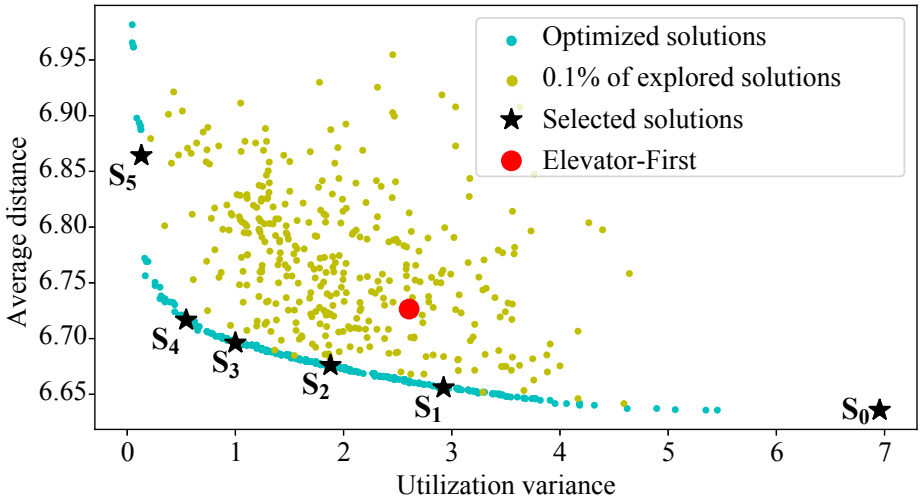



**Figure 2.6:** Elevator-subset solutions found by AMOSA in AdEle+.

### 2.4.2 Parameter Exploration and Optimization

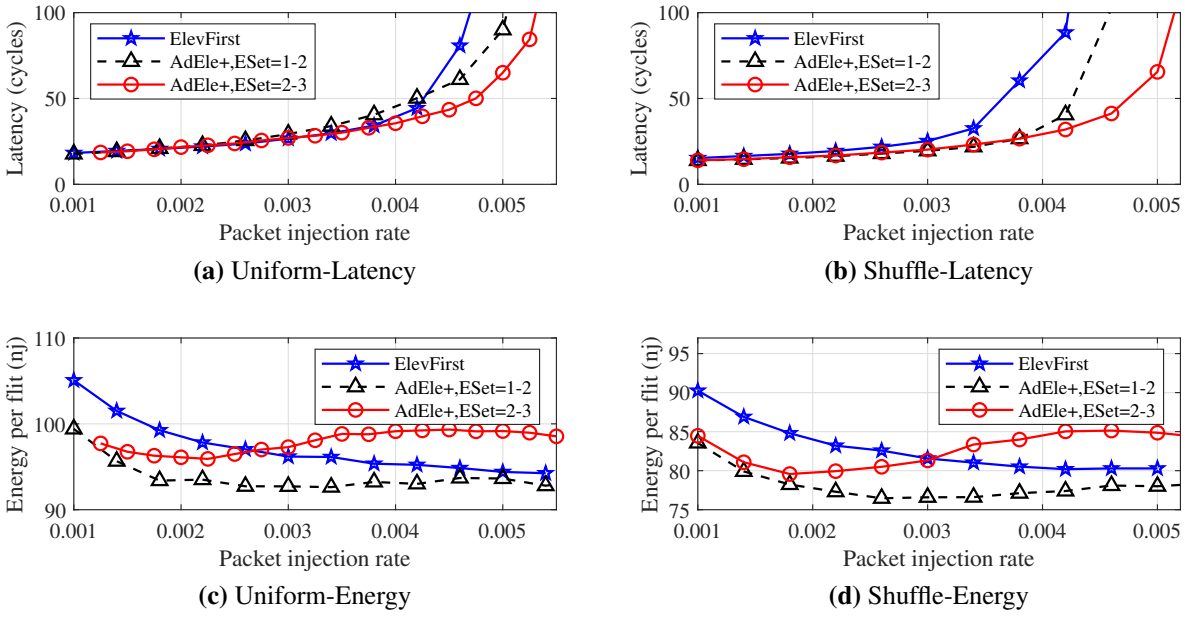
#### AMOSA Elevator-Subset Exploration

As discussed in Section 2.3.2, AMOSA finds various solutions with different elevator utilization variances and average distances. To show the solution-selection process in AdEle+ and as an example, the optimization for $P_L$ is detailed here. A small sample of AMOSA's explored solutions is shown in Fig. 2.6. As AMOSA explores the solution space, it makes its way towards the Pareto front (blue curve) to find the optimal trade-offs between utilization variance (see (2.3)) and average distance (see (2.5)). Given the final set of solutions, a desired solution can be selected depending on the importance of energy efficiency (average distance) and latency (utilization variance). For brevity, we simulated several solutions spread along the Pareto front ($S_0$ to $S_5$) and summarized the results in Table 2.2. As expected, lower utilization variance and lower average distance improves the latency and energy consumption, respectively. As we are able to significantly reduce the latency with fairly minimal increases in energy, we select $S_5$ for further analysis. Moreover, as we discussed in Section 2.3.4, AdEle+ will dynamically switch to DB elevator selection to save energy when the traffic load is low and congestion is not a concern. We follow the same procedure for $P_{SL}$, $P_{SM}$, $P_{SH}$, and $P_{SNU}$ to find an optimized set of elevator subsets.

**Table 2.2:** Performance of selected solutions from Fig. 2.6

| | Elev. First | Optimized solutions | | | | | |
|---|---|---|---|---|---|---|---|
| | | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$✓ |
| Latency* | 161.4 | 396 | 209 | 156.6 | 76.9 | 67.4 | 56.6 |
| Energy# | 94.4 | 93.1 | 94.2 | 94.6 | 94.4 | 94.8 | 98.3 |

*Avg. Latency (cycles) #Energy/flit ($nj$) ✓Selected




**(a)** Uniform-Latency **(b)** Shuffle-Latency

**(c)** Uniform-Energy **(d)** Shuffle-Energy

**Figure 2.7:** Impact of elevator set size on (a and c) uniform traffic and (b and d) shuffle traffic.

In the elevator subset optimization, we establish a minimum ($ESet_{min}$) and maximum ($ESet_{max}$) number of elevators that each elevator subset may have. This allows AMOSA to choose different elevator subset sizes for different routers. Although having a large number of elevators in the subset helps the router adapt to different traffic scenarios, considering a very large number of elevators also increases the chance of selecting distant elevators, which impacts the energy efficiency.

To analyze the impact of elevator set size, average latency and energy-per-flit for $P_L$ is shown in Fig. 2.7 using uniform and shuffle traffic patterns. Two ranges of elevator subset sizes are considered: $ESet_{min} = 1$ and $ESet_{max} = 2$ ($ESet$ =1–2); and $ESet_{min} = 2$ and $ESet_{max} = 3$ ($ESet$ =2–3). For elevator routers, we force them to only use their local elevator because the cost of rerouting outweighs the load balancing benefits. Since the elevator subsets are optimized based on uniform traffic, adding more elevators in the subset to help adapt to runtime traffic does

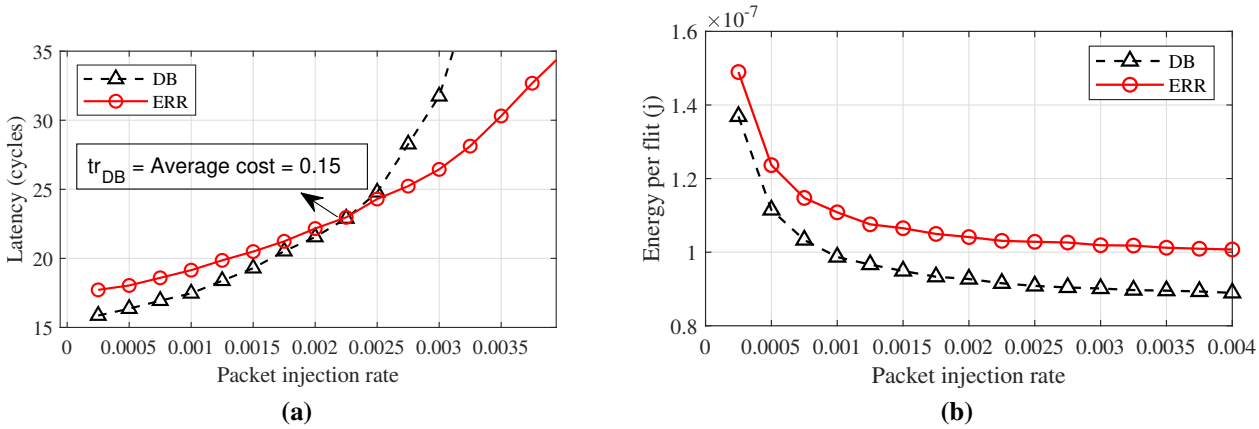



**Figure 2.8:** Comparison of minimal versus ERR elevator selection under Uniform traffic: (a) average latency and (b) energy per flit. This is used to find $tr_{DB}$.

**Table 2.3:** $tr_{DB}$ for different elevator placements

| | $P_{SL}$ | $P_{SM}$ | $P_{SH}$ | $P_{NU}$ | $P_L$ |
|---|---|---|---|---|---|
| Uniform | 0.08 | 0.07 | 0.06 | 0.07 | 0.15 |
| Shuffle | 0.13 | 0.06 | 0.1 | 0.07 | 0.3 |
| Transpose | 0.07 | 0.05 | 0.04 | 0.05 | 0.06 |

only little to improve the latency under the uniform traffic. On the other hand, in an unseen traffic pattern like shuffle, $ESet$ =2–3 shows better performance because it can benefit from a larger elevator subset size to adapt to the new traffic. We also evaluated $ESet_{min} = ESet_{max} = 3$, and $ESet_{min} = 3$ and $ESet_{max} = 4$. However, we did not observe any significant improvement in the latency, but we observed energy overhead. Similar to these results, we found that, for different network sizes, traffic patterns, and elevator patterns (See Table 6.1), $ESet_{min} = 2$ and $ESet_{max} = 3$ will result in a better performance, and therefore we use this configuration.

### Distance-based selection threshold ($tr_{DB}$)

As discussed in Section 2.3.5, we find the DB selection threshold ($tr_{DB}$) by varying the traffic injection rates and examining the latency. Results for $P_L$ are shown in Fig. 2.8. As expected, DB elevator selection has a lower latency when injection rates are low, while ERR improves network congestion and outperforms DB selection at higher injection rates. However, as ERR can use non-minimal paths, it consistently has worse energy consumption compared to DB (see Fig. 2.8b). To

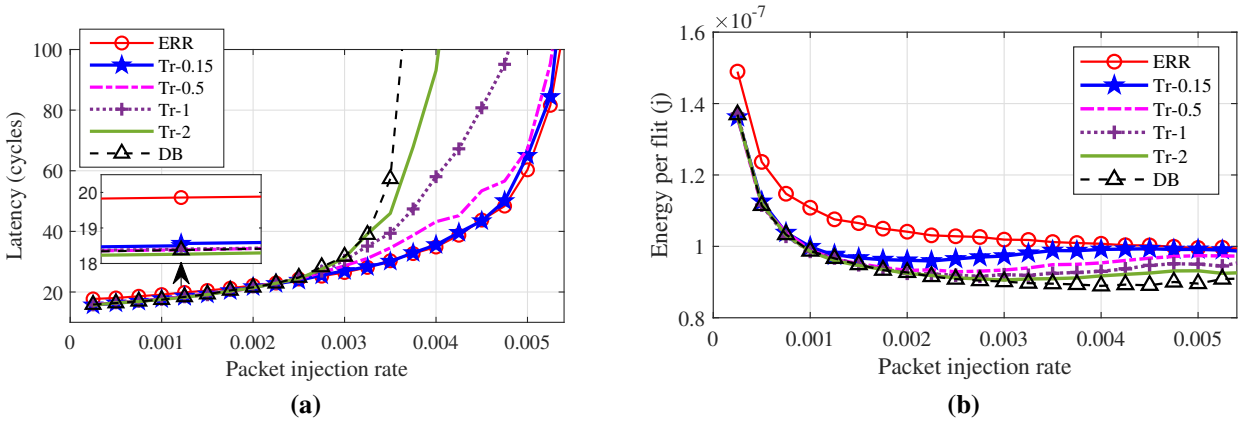



**Figure 2.9:** Comparison of ERR selection under Uniform traffic with different values of $tr_{DB}$ for $P_L$ in (a) average latency and (b) energy per flit.

improve latency, ideally we would like to use DB when the packet injection rate is below 0.00225 (the intersection occurred when packet injection rate is 0.00225), and ERR otherwise. Therefore, at an injection rate of 0.00225, we calculate the average cost of elevator selections (for all selections in all the routers) and using (2.10) $tr_{DB}$ is calculated, which is 0.15 for $P_L$.

To show the impact of different values of $tr_{DB}$, we show the latency and energy results across a range of $tr_{DB}$ for $P_L$ in Fig. 2.9. As it can be seen, the threshold value we earlier determined (i.e., $tr_{DB} = 0.15$) is able to achieve a relatively low latency at both high and low injection rates, demonstrating the efficacy of our approach. Moreover, as shown in Fig. 2.9b, energy consumption is improved in low injection rates, compared to ERR, due to switching to DB selection.

Following the same procedure, we find $tr_{DB}$ for $P_{SL}$, $P_{SM}$, $P_{SH}$, and $P_{SNU}$ across different traffic patterns (see Table 2.3). Although there is a small difference in optimal $tr_{DB}$ values, we do not expect to see a large impact on the overall results if we use the $tr_{DB}$ computed based on the uniform traffic for other traffic patterns. From Fig. 2.9, we can see that even moving from $tr_{DB} = 0.15$ to $tr_{DB} = 0.5$—a range much larger than what can be seen in Table 2.3—has a relatively small impact on the latency and energy. Therefore, for each PC-3DNoC configuration, we assume the optimal $tr_{DB}$ found under the uniform traffic for all the traffic patterns.

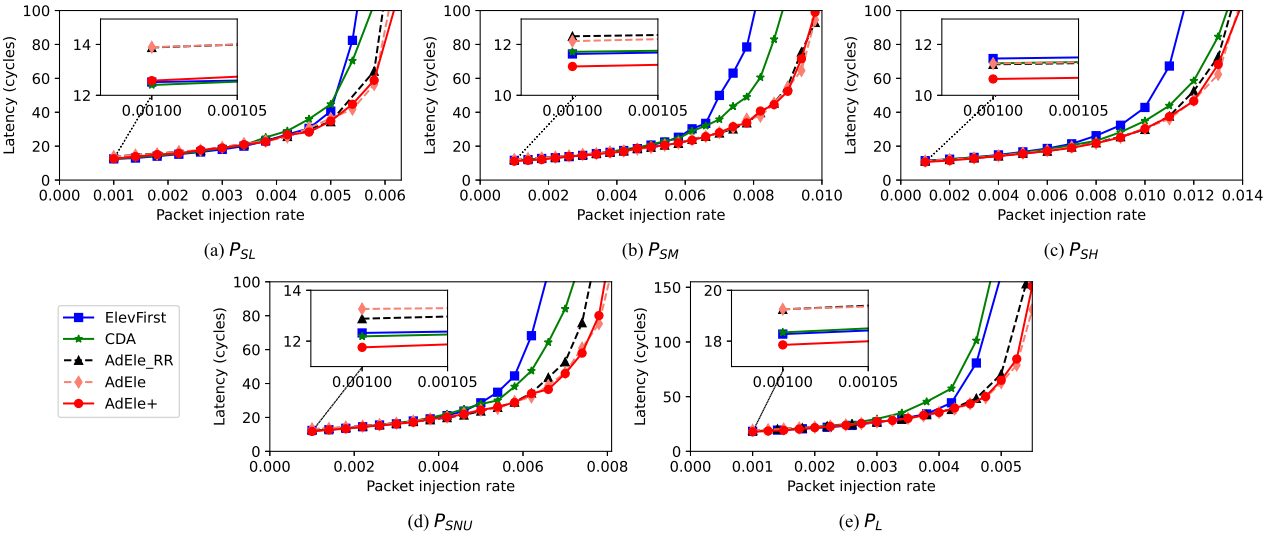



**Figure 2.10:** Average latency for Elevator-First, CDA, AdEle, AdEle_RR, and AdEle+ under uniform traffic and using different elevator placements.

### 2.4.3 Evaluation Results

In this subsection, we evaluate AdEle+ compared to the state-of-the-art selection algorithms (Elevator-First [1] and CDA [20]) in terms of latency, energy, and area efficiency. Our evaluation includes both synthetic and real traffic patterns.

#### AdEle+ Performance Under Synthetic Traffic

We first compare the average latency in AdEle+ under uniform and shuffle synthetic traffic patterns, with Elevator-First, CDA and AdEle in Figs. 2.10 and 2.11. Under all the elevator placements (see Table 6.1) and both traffic patterns, AdEle+ achieves the lowest latency and highest saturation throughput. Even though CDA employs global traffic information, AdEle+ still shows better performance while only relying on local information. In this work, we do not consider the high cost of CDA's global information sharing and optimistically assume that the information is instantaneously received at every router. In reality, CDA will likely perform even worse with stale information or include significant implementation overhead.

With a higher elevator density (e.g., $P_{SH}$) or larger horizontal dimensions (e.g., $P_L$), the intra-layer traffic will become more critical. AdEle+ still shows better performance in these cases. To demonstrate the efficacy of our DB approach, we compare AdEle+ with the policy that uses the

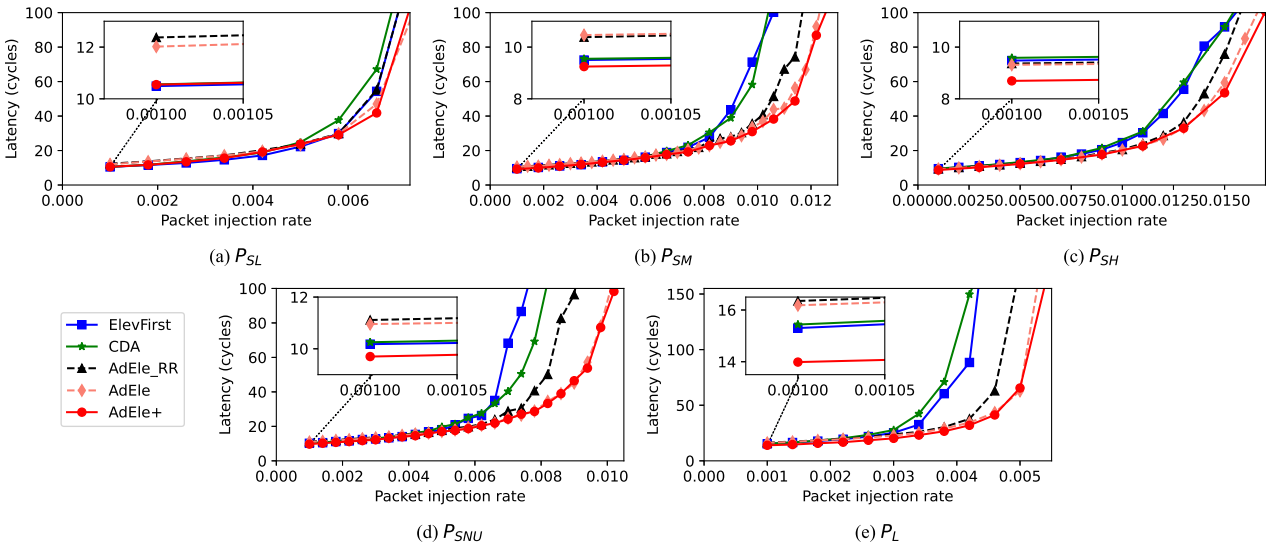



**Figure 2.11:** Average latency for Elevator-First, CDA, AdEle, AdEle_RR, and AdEle+ under shuffle traffic and using different elevator placements.

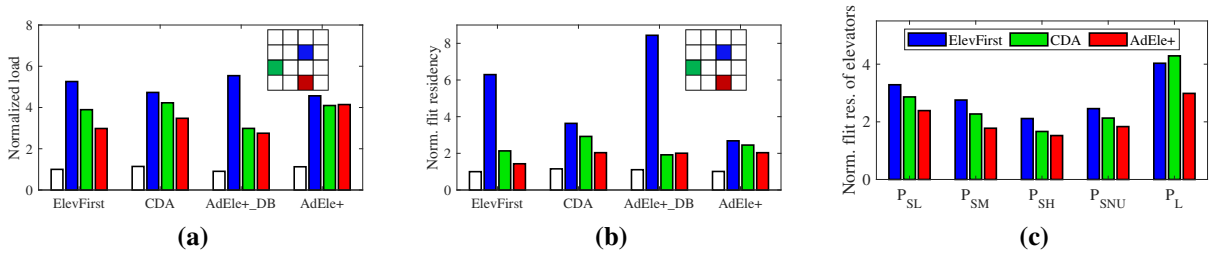



**Figure 2.12:** Comparison of elevator traffic distribution for Elevator-First (ElevFirst), CDA, and AdEle+ in terms of (a) traffic load over elevators (blue, green, and red) normalized to the average load over horizontal links (white bars) of Elevator-First; (b) average flit residency over elevators; and (c) average flit residency of all elevators normalized to average flit residency of horizontal links. AdEle+_DB shows DB selection of AdEle+.

ERR approach at all injection rates (AdEle [40]). As it can be seen, the DB approach is able to significantly improves the average latency at low injection rates (see in-set plots). Since the DB approach is able to utilize the shortest path when traffic congestion is not an issue, AdEle+ is able to lower the latency compared to the ERR approach. At high injection rates, AdEle+ switches over to ERR and has a negligible latency overhead.

In addition, recall that AdEle+'s offline optimization is performed using uniform traffic only. Yet, as Fig. 2.11 shows, while the shuffle traffic is new for AdEle+, it still achieves the lowest latency because its online selection policy can monitor runtime congestion and select better eleva-

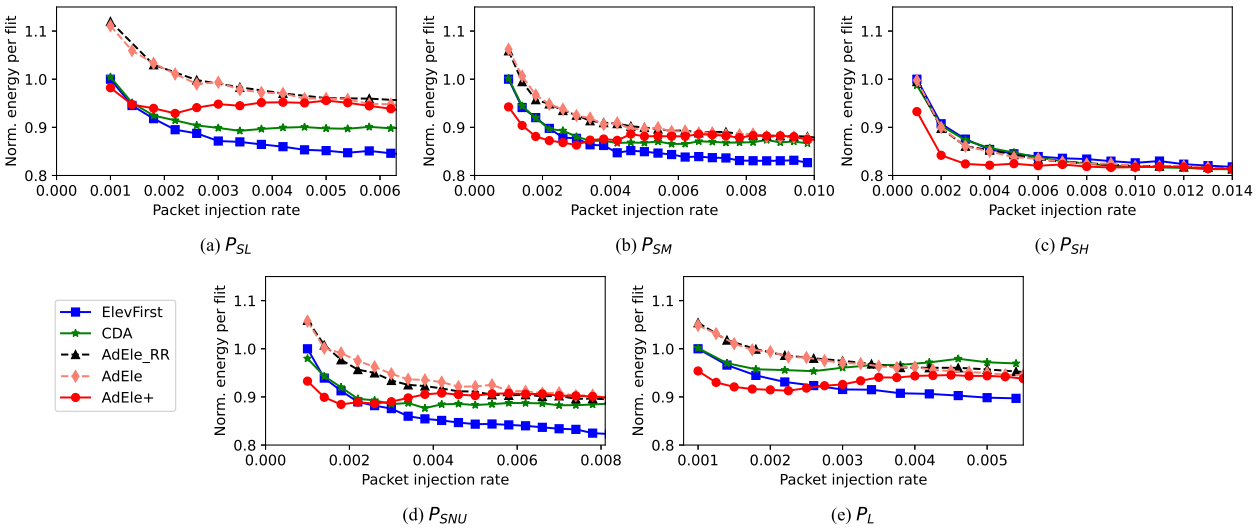



**Figure 2.13:** Normalized energy per flit for Elevator-First, CDA, AdEle, AdEle_RR, and AdEle+ under uniform traffic with different injection rates.

tors. We also include the average latency of AdEle with conventional round-robin selection (called AdEle_RR). As can be seen, AdEle+'s proposed online skipping policy achieves higher improvements in latency compared to RR under both uniform and shuffle traffic patterns. Notably, AdEle+ with ERR (shown as AdEle+ in the results) has more improvement with the unseen traffic (i.e., shuffle) than the one used in its offline optimization (i.e., uniform). This demonstrates that ERR can successfully adapt to new traffic patterns.

To further explain the latency improvement in AdEle+, we show the elevator traffic load distribution for $P_{SL}$ normalized to the average router load in Fig. 2.12a. The white bar shows the average load over elevator-less routers. The other colored bars show the load over different elevators as indicated in the inset. As it can be seen, AdEle+ better balances the load across the three elevators and reduces the load on the highest utilized elevator. To help quantify the congestion at an elevator, we examine the average residency of flits on elevators (i.e., the time that a flit is waiting at an elevator) in Fig. 2.12b. Due to the high traffic load on the blue elevator in Elevator-First and CDA, we see a high flit residency on the blue elevator. By balancing the traffic load, AdEle+ shows almost the same flit residency on the three elevators. Therefore, balancing the traffic load on the elevators results in less waiting time at the elevators, and hence the average latency is improved. We also analyzed flit residency on elevators for other patterns in Fig. 2.12c. The same trend is seen:

AdEle+ significantly improves the average flit residency because of better load balancing on the elevators. In Fig. 2.12, packet injection rate is the injection rate at which latency is $3\times$ zero-load latency: $P_{SL}$: 0.0054, $P_{SM}$: 0.0078, $P_{SM}$: 0.011, $P_{SNU}$: 0.0062, and $P_L$: 0.005.

We compare the energy consumption in different elevator-selection algorithms and under various elevator placements in Fig. 2.13. Under low injection rates, AdEle+ always has the lowest energy consumption with an average of 5.6% energy improvement over AdEle, because it switches to DB selection and uses the minimal paths. As it can be seen, DB selection of AdEle+ offers significantly lower energy compared to AdEle [40]. However, because AdEle+ takes non-minimal paths at higher injection rates to improve traffic loads across elevators, it typically imposes a small energy overhead. When using $P_{SL}$, the low density of elevators causes more pressure on each elevator and increases the chance of taking non-minimal paths, incurring at most 6.4% energy overhead compared to CDA. However, distributing the traffic properly is especially important with a low number of elevators as the traffic pressure is already high and unbalanced loads can greatly affect the performance. If maximum energy efficiency is desired, then AdEle+ can use configurations with lower energy but higher latency (see Table 2.2). For PC-3DNoC configurations with a higher density of elevators ($P_{SM}$, $P_{SH}$, and $P_{SNU}$), there is almost no energy overhead compared to CDA and negligible overhead compared to Elevator-First at higher injection rates. Although in a non-uniform elevator placement like $P_{SNU}$ congestion aware selection of elevators can potentially result in a larger average distance, AdEle+ has a negligible energy overhead due to the efficient offline optimization. Finally, using $P_L$, AdEle+ shows energy consumption improvement compared to CDA. The reason is that AdEle+ selects an elevator among a set of nearby elevators, while CDA is free to select any elevator including those farther away.

**AdEle+ Performance under Real-Application Traffic**

We extracted the traffic of several SPLASH-2 [41] and PARSEC [42] benchmarks using Gem5 [38] for real-application simulations. We obtained the real-application traffic traces using Gem5 simulations in full-system mode with 64 x86 cores, four coherence directories, four shared L2 cache banks (each core also has a private L1 cache), 64 threads, and simmedium input size. Because

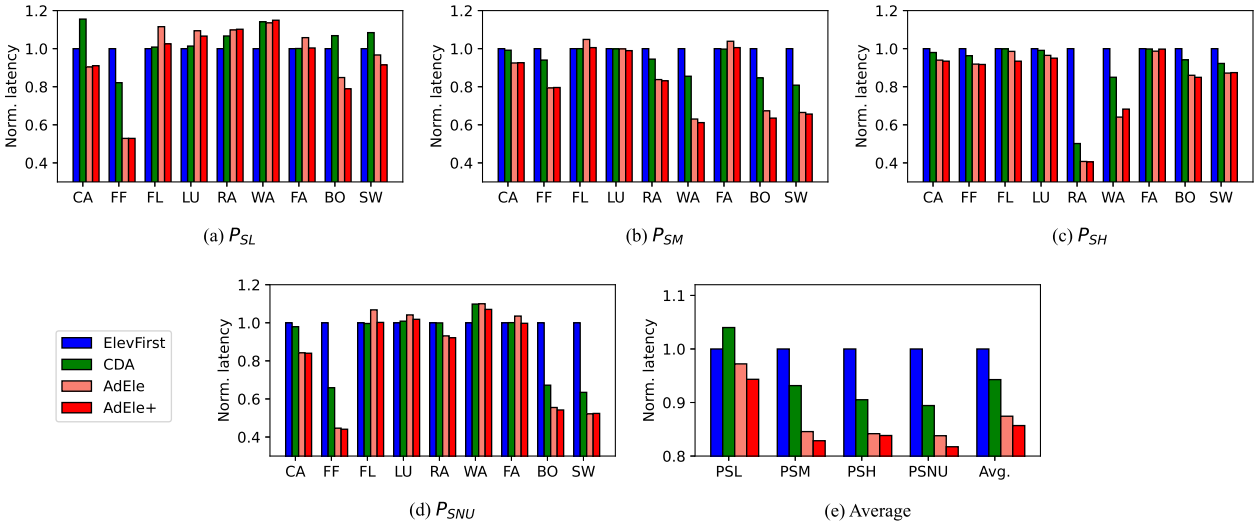



**Figure 2.14:** Latency for Elevator-First (ElevFirst), CDA, AdEle and AdEle+ normalized to ElevFirst under real-application traffic with different elevator placements.

Gem5 is limited to 64 cores, we demonstrate our results for $P_{SL}$, $P_{SM}$, $P_{SH}$ and $P_{SNU}$. As shown in Figs. 2.14(a)–(e), AdEle+ improves the network latency by 9.3%, 11.2%, 8.4%, and 8.9% (9.5% on average) compared to CDA; by 5.7%, 17.2%, 16.2% and 18.3% (14.3% on average) compared to Elevator-First; and by 3%, 2.1%, 0.5% and 2.5% (2% on average) compared to AdEle using $P_{SL}$, $P_{SM}$, $P_{SH}$ and $P_{SNU}$, respectively. Note that the first two letters of application are used on the x-axis in the figures—CA: canneal, FF: fft, FL: fluidanimate, Lu: lu, RA: radix, WA: water, facesim: FA, bodytrack: BO and swaption: SW. In particular, AdEle+ has more improvements in applications with higher traffic loads (canneal, fft, radix, water, bodytrack and swaption), as there is more opportunity to reduce the resulting elevator congestion. In applications with lower traffic loads (fluidanimate, lu and facesim), AdEle+ maintains almost similar performance to the other approaches as there is little contention on the elevators and the latency is close to zero-load latency. Although $P_{SL}$ still shows some improvements for AdEle+, the lower number of elevators (three in $P_{SL}$) results in minimal opportunity for AdEle+ to redirect traffic and improve latency. Compared to AdEle, AdEle+ shows lower latency (2% on average) especially in low load traffic (e.g., 5.9% averaged across network configurations for fluidanimate), due to its efficient DB elevator selection.

Fig. 2.15 shows the average energy-per-flit for each elevator-placement pattern ($P_{SL}$, $P_{SM}$, $P_{SH}$, and $P_{SNU}$), normalized to Elevator-First. AdEle+ imposes small overhead because it routes

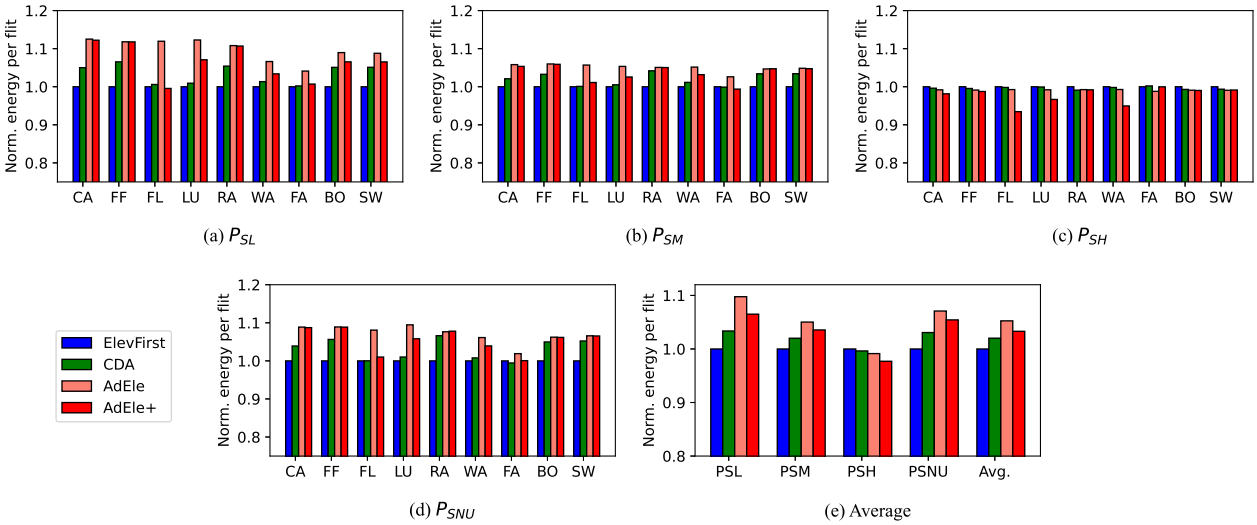



**Figure 2.15:** Energy for Elevator-First (ElevFirst), CDA, AdEle, and AdEle+ normalized to ElevFirst under real-application traffic with different elevator placements.

packets over non-minimal paths in case of congestion to improve the latency. Compared to CDA, AdEle+ has a negligible overhead using $P_{SL}$, $P_{SM}$, and $P_{SNU}$, while it slightly improves the energy consumption under $P_{SH}$. On average, AdEle+ has improved energy consumption by 2%, in comparison with AdEle, due its efficient DB selection.

To see how sensitive AdEle+ is to packet size, we perform the same analysis for a packet size of five flits. In Fig. 2.16, we present the average latency and energy-per-flit over all network configurations ($P_{SL}$, $P_{SM}$, $P_{SH}$, and $P_{SNU}$) and all nine real applications. To evaluate the effect of packet size, we maintain the same flit injection rate in both 5-flit and 18-flit cases. Although flit injection rate is the same, as the chance of congestion is reduced under the small packet size, both CDA and AdEle, which are congestion-aware approaches, have slightly less improvement in comparison with Elevator-First under the small packet size. On the other hand, AdEle+ offers slightly lower latency (0.2% improvement) and energy-per-flit (2.5% improvement) in the small packet size. AdEle+ can update the cost function faster with a higher packet frequency (i.e., smaller packet size) because, unlike CDA, AdEle+ updates its cost function every packet instead of using a time interval for the update (i.e., an event driven approach instead of a time driven one). This allows AdEle+ to more quickly adapt to traffic dynamics and improve performance.

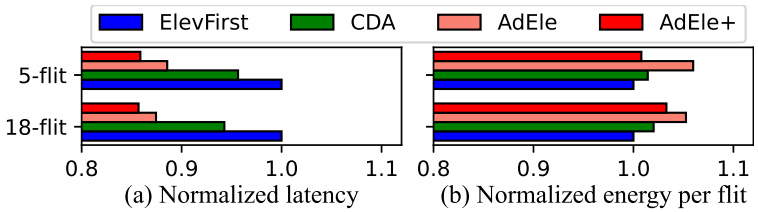



**Figure 2.16:** Packet size effect on latency and energy-per-flit.

### Hardware-Area Analysis and Comparison

The hardware of Elevator-First, AdEle+, and CDA are implemented and analyzed using Cadence Genus [39] in 45 nm technology. Here, we consider a 1 GHz clock. For AdEle and AdEle+, we consider 8-bit precision for (2.6) and (2.7) and 5-bit precision for (2.8) and (2.9). We find that 8-bit precision is sufficient to cover the range of latency values before the network reaches saturation and maximum latency. However, we chose a lower precision for relative cost to save area overhead due to the division operation. We found that at 5 bits, the approximation resulting from lower precision resulted in a negligible effect (less than 0.1%) on AdEle+'s latency and energy. Therefore, we use these level of precision assumptions in our latency and energy evaluation presented in Section 2.4. The results are shown in Table 2.4. Compared to CDA, AdEle+ has smaller area overhead. Since AdEle+ only requires local traffic information, AdEle+ does not affect router frequency and AdEle+ calculations can be done in 1 cycle. However, CDA's area overhead is an optimistic assumption here as it does not include any overhead related to the actual sharing of information. Therefore, real CDA will likely impose higher area and latency overhead. Also, AdEle+ does not affect the router stages and will scale well with the network size, while CDA requires an additional cycle (or more for larger networks) to update its tables. As mentioned in Section 2.3.4, AdEle+ will, at most, only need to store five elevator locations regardless of the system size and elevator density to support the DB selection. Compared to AdEle, AdEle+ imposes a negligible area overhead (i.e, 0.2%) due to DB selection, but DB selection improves both energy and latency significantly. In summary, the results presented in this section using different traffic patterns and network configurations show the promise of AdEle+ to manage congestion on elevators with low hardware overhead.

**Table 2.4:** Area overheads for the different elevator selection algorithms

| | Cycles | Router area ($\mu m^2$) | |
|---|---|---|---|
| Base (ElevFirst) | 1 | 35550 | Overhead |
| CDA* | 2 | 41088 | 14.4% |
| AdEle | 1 | 36875 | 3.7% |
| AdEle+ | 1 | 36954 | 3.9% |

*global information sharing is not included.

## 2.5 Conclusion

Elevator selection plays a crucial role in the network latency and energy efficiency of partially connected 3D NoCs. This chapter has combined an offline elevator subset optimization process with an online elevator selection, to create a lightweight adaptive congestion- and energy-aware elevator-selection algorithm, called AdEle+, that addresses the traffic congestion on elevators while delivering packets with less hop counts in low traffic circumstances. By employing a set of elevators instead of one elevator for each source router, AdEle+ is able to adapt to runtime traffic loads and select the best elevator during runtime. Moreover, AdEle+ only requires local router information and is able to improve average latency in various scenarios under both synthetic and real traffic. AdEle+ also improves the energy consumption in some applications, especially under low traffic loads where there is a low chance of congestion. Results indicate the promise of AdEle+ to improve the network latency in PC-NoCs and prevent network hot-spots. Therefore, the work presented in this chapter can help in designing low-latency and energy-efficient networks for high performance 3D manycore systems.

# Chapter 3

# A Reliable and Deadlock-Free Routing Technique for 2.5D Chiplet-based Interposer Networks

2.5D integration offers a cost-effective and reliable solution for implementing large-scale modular systems. A 2.5D chiplet system can be designed by connecting smaller chiplets through an interposer, where the chiplets may have heterogeneous architectures. In addition to the intra-chiplet network (e.g., a network-on-chip on the chiplet), a network is used on the interposer to enable efficient and scalable communication among different chiplets. However, this global network, which consists of intra-chiplet and inter-chiplet networks, is susceptible to deadlock, despite using deadlock-free networks on the chiplets and interposer. Moreover, 2.5D networks are not only vulnerable to horizontal link (HL) faults but also to those in the vertical links (VLs) connecting the chiplets to the interposer. In addition, such faults cannot be effectively addressed by existing fault-tolerant routing techniques designed for 2D and 3D networks-on-chip. To overcome these challenges, this chapter introduces a novel Reliable and Deadlock-free routing algorithm, called *ReD*, for fault-tolerant communication in 2.5D chiplet systems. *ReD* leverages a virtual-network-based approach to guarantee deadlock freedom while tolerating VL and HL faults. Besides VL faults, due to the difference in VL technology (i.e., microbump technology), the number of VLs connecting a chiplet to the interposer is limited, making VLs a source of congestion. *ReD* enhances VL selection in such scenarios to tolerate VL faults and improve network congestion by balancing VL utilization. Compared to the state-of-the-art routing algorithms, simulation results obtained by simulating chiplet systems under HL and VL faults demonstrate that *ReD* significantly improves the network reachability by up to 75% and reduces the network latency by up to 40%, while incurring less than 2% area overhead.

## 3.1 Introduction

Large-scale systems-on-chip (SoCs) with increasing heterogeneity and integration density have recently received much attention due to the growing demand for high computation capabilities [43, 44], especially for emerging machine learning applications. However, conventional 2D SoCs lack scalability because manufacturing yield significantly decreases as the chip size increases [45]. Therefore, high manufacturing costs are imposed when attempting to manufacture a system with high computation capabilities on a single large-scale chip [45]. To this end, 3D and 2.5D integration partitions an SoC into multiple smaller dies that are connected using different technologies. In the 3D integration, the dies are vertically stacked and interconnected using through-silicon vias (TSVs) to enable higher integration density and improved performance. However, 3D integration can result in high fabrication costs, power density, and low cooling conductivity, creating thermal hotspots and degrading the system reliability [46]. On the other hand, the 2.5D integrated approach presents a modular solution by placing several chiplets on an interposer on which inter-chiplet communication is supported [47, 48]. In addition, and similar to 3D SoCs, in such a modular integration, each chiplet can be designed heterogeneously and independently [49–51]. Moreover, the design time of chiplet systems is significantly shorter than the traditional monolithic designs since off-the-shelf chiplets can be integrated on an interposer to make chiplet systems with different computation capabilities for various purposes [25, 47]. Therefore, chiplet-based systems can leverage the strengths of different manufacturing processes and technologies for each chiplet. However, 2.5D integration introduces different challenges, including deadlock avoidance [52] and fault tolerance [23] due to both vertical link (VL; connecting chiplets to the interposer) and horizontal link (HL; on the chiplets and the interposer) faults.

The intra- and inter-chiplet networks in a 2.5D chiplet system can suffer from deadlock when a cyclic dependency of requests for buffers occurs in the routing process. Conventionally, to avoid deadlock, some turns (i.e, some requests for buffers in specific directions) can be restricted to break the cyclic dependency in a network [53]. However, even when deadlock-freedom is guaranteed in intra-chiplet networks using conventional approaches [53], the inter-chiplet packets can still

create some dependencies, and hence deadlock. In [52], a modular-turn restriction (MTR) routing algorithm was proposed where the routing on each chiplet is designed separately by avoiding some turns from the chiplet to the interposer and vice versa. However, the interposer network needs to be aware of the restricted turns within chiplets. This violates a key attribute in 2.5D systems to enable the design of each chiplet and the interposer network independently without the knowledge of the whole design [54, 55]. Moreover, the turn restrictions limit routing adaptation and impose performance overhead by creating imbalances in load among routers, especially on the boundary routers that connect the chiplets to the interposer. To this end, in the Remote Control (RC) routing algorithm [54], deadlock-freedom of chiplet systems is guaranteed by reserving an extra buffer on the boundary routers to store and forward the inter-chiplet packets. However, RC not only imposes high area and power overhead due to the extra resources needed but also limits VL selection and path diversity in the routing process.

In addition to deadlock-freedom, enabling link and router fault-tolerance is essential in 2.5D chiplet systems [56], but it has not yet been addressed. Existing fault-tolerant solutions in 2D and 3D networks cannot be applied to 2.5D networks, where deadlock-freedom is more challenging due to higher irregularity in the network. In particular, enabling fault-tolerance in Networks-on-Chip (NoCs) requires higher path redundancy to be supported in the routing algorithm [57], which when extended to 2.5D chiplet systems makes the deadlock-freedom and load-distribution even more complex. Current routing algorithms [52, 54] to address deadlock in 2.5D chiplet systems limit the VL selection and hence deteriorate network reliability and performance. Furthermore, in 2.5D systems, it is imperative to acknowledge the inherent limitations in bandwidth at the boundaries of chiplets, particularly on the VLs connecting chiplets and the interposer. To mitigate potential congestion at these boundaries, the load distribution strategy becomes paramount. This becomes especially critical when considering fault scenarios, where a fault occurrence can lead to a reduction in the chiplet-to-interposer bandwidth. In such cases, distributing traffic efficiently across the available bandwidth is essential to alleviate and prevent congestion.

To address the aforementioned challenges and constraints, this chapter develops a Reliable and Deadlock-free routing algorithm, called *ReD*, for 2.5D chiplet-based interposer networks. In particular, the main contributions of this chapter are:

- We propose a novel virtual-network (VN) based approach for 2.5D networks to guarantee deadlock-freedom in both the intra- and inter-chiplet networks while offering high-performance communication. The VN-based approach uses a novel VN assignment strategy to maximize network virtual-channel (VC) utilization balance.

- We present a novel pseudo-dynamic VL-selection strategy that not only enables *ReD* to tolerate VL faults but also improves the load distribution on the VLs to reduce the network latency under fault scenarios.

- We propose a Hamiltonian-based routing technique to share the faulty/healthy status of VLs among all routers in the local chiplet/interposer. The presented mechanism is designed to tolerate HL faults without the need for any extra control network or VCs.

*ReD* only requires two VNs without imposing any limitations on VL selection. VN assignment in *ReD* is performed in such a way that intra-chiplet packets and inter-chiplet packets on the interposer are free to be routed using each of the two VNs. The freedom in VN assignment enables *ReD* to tolerate HL faults without adding any extra VCs. Our experimental results show the promise of *ReD*; it can achieve 100% reachability under different VL-fault scenarios (e.g., 25% faults) in chiplet systems of different sizes, ranging from 4 to 12 chiplets. In addition, with less than 2% area overhead, *ReD* improves the latency under real-application traffic on average by 3% and 13.5% for low and high traffic scenarios, respectively.

The rest of the chapter is organized as follows. We discuss some background and related work in Section 3.2. Section 3.3 presents the proposed *ReD* routing algorithm and its novel solution for deadlock-freedom, fault-tolerance, and congestion-aware VL selection. Section 3.3 also discusses our fault-tolerant approach to detour packets around faulty HLs in intra-chiplet/interposer routing.

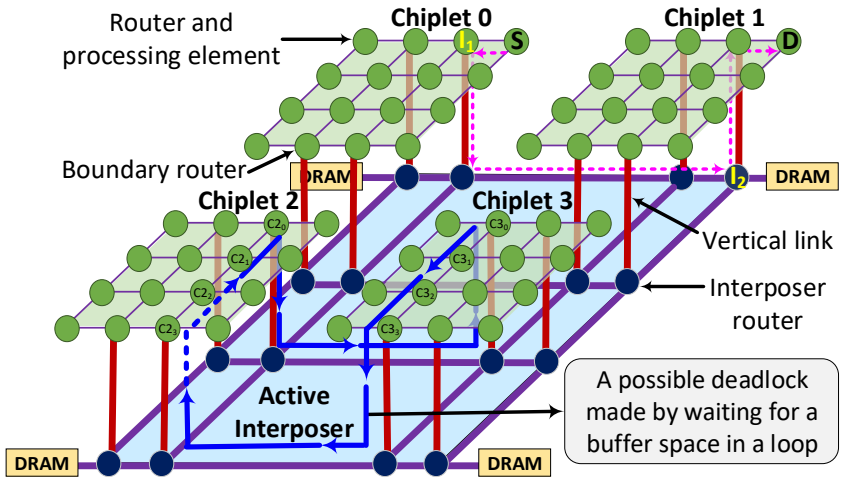



**Figure 3.1:** An abstract overview of the baseline 2.5D network with four chiplets on an active interposer.

We present our evaluation and simulation results in Section 3.4. Finally, Section 3.5 concludes the chapter.

## 3.2 Background and Related Work

### 3.2.1 Active interposers and their challenges

There are two main types of interposers: active and passive. Passive interposers act as simple substrates with wiring layers that enable connectivity between chiplets. They are generally less complex and have lower power consumption compared to active interposers. However, passive interposers may face limitations in realizing flexible and efficient long-distance communication, especially when integrating a substantial number of chiplets. In contrast, active interposers incorporate additional circuitry, such as routers and power management units, enabling them to perform more complex functions. They offer advantages such as facilitating communication among chiplets with different protocols, supporting scalable cache coherency, and providing new opportunities for performance enhancement [58]. Despite their benefits, active interposers pose challenges, particularly in yield, power, and thermal issues. However, the lower complexity of the circuits on the interposer, including routers and circuits for power management and testing, contributes to a higher yield compared to the chiplets. Specifically, in [58], the interposer's complexity is 99.5%

less than the chiplet complexity in terms of the number of transistors. This results in low power consumption in the interposer and mitigates thermal concerns due to the lower heat generation and heat confinement of the thinner silicon layer. Furthermore, employing older mature technology for the interposer enhances yield, as highlighted in [59].

### 3.2.2 Deadlock-Free Routing in Active Interposers

An example of a 2.5D chiplet system is shown in Fig. 3.1. Note that we also use this architecture as our case study in our evaluation presented in Section 3.4. Various integration approaches are available for chiplet systems that combine CPU, GPU, and DRAM chiplets [45]. Although we focus on the configuration shown in Fig. 3.1, our methodology can be applied to any chiplet system. The system in this example consists of an active interposer and four CPU chiplets. Each chiplet is connected to the interposer through four bidirectional VLs. The routers located on the chiplets that are connected to the VLs are referred to as *boundary routers*.

Routing in chiplet-based systems involves two intermediate destinations: one on the source chiplet and another on the interposer. For instance, in Fig. 3.1, the magenta routing path from node $S$ on Chiplet 0 to node $D$ on Chiplet 1 uses $I_1$ and $I_2$ as intermediate destinations. Initially, a VL is selected on the source chiplet as the first intermediate destination. The packet is then routed to the selected VL and vertically transmitted to the interposer. Subsequently, the second intermediate destination is chosen on the interposer to direct the packet toward its destination chiplet. Finally, the packet reaches its intended destination on the destination chiplet.

In the process of routing, when an incoming packet reaches a router's buffer, it can move forward by selecting an output port, based on the routing algorithm and the availability of the next router's buffer. However, waiting for buffer availability can create a cyclic dependency or deadlock, where packets are stuck in a loop, waiting for each other to release reserved buffers. To prevent deadlock, certain routing algorithms like dimension-order routing (e.g., XY routing in a 2D mesh network topology) have been proven to be deadlock-free according to Glass et al. [60]. These algorithms restrict some turns in the network to prevent cycles. Although intra-chiplet and/or inter-

chiplet network routing can be deadlock-free by using XY routing or conventional turn models as described in [53], integrating multiple chiplets on an interposer can introduce cyclic dependencies in the global network, which includes both intra-chiplet and inter-chiplet networks, resulting in deadlock. An illustration of a deadlock scenario caused by specific turns (indicated by blue arrows) between the deadlock-free Chiplet 2, interposer, and Chiplet 3 is depicted in Fig. 3.1. This cycle may occur with the following packet routing: $C3_2$ to $C2_2$, $C2_1$ to $C3_1$, $C3_0$ to $C3_3$, and $C2_3$ to $C2_0$.

Several routing algorithms have been proposed recently to address the issue of deadlock in 2.5D chiplet systems based on the mesh architecture. Two notable deadlock-free routing algorithms are the Modular-Turn Restriction (MTR) algorithm [52] and the Remote Control (RC) algorithm [54]. The MTR routing algorithm prevents certain inter-chiplet turns at the boundary routers, effectively breaking cyclic dependencies. For instance, in Fig. 3.1, avoiding the left-to-down turn in Chiplet 2 (indicated by the dashed blue arrow) can resolve the cyclic dependency. However, the MTR algorithm introduces a dependency between the interposer router and chiplet designs, which goes against the modular design requirement in chiplet-based systems. This is because each interposer router needs to know whether a packet can reach its destination through a VL while considering the turn restrictions. The RC routing algorithm [54] addresses the inter-chiplet cyclic dependency by incorporating an additional buffer called the RC-buffer in the boundary routers. The RC-buffer stores and forwards the entire packet and is shared among the chiplet routers that utilize the boundary router for inter-chiplet communication. The store and forward technique can break the dependency at the boundary router. However, implementing RC requires additional hardware for the RC-buffer and a permission network since the RC-buffer is shared among multiple routers. Moreover, both MTR and RC algorithms restrict VL selection, which limits their ability to dynamically re-select VLs and renders them unable to tolerate VL faults. In [61], Wu et al. proposed a deadlock recovery approach for 2.5D chiplet networks. However, while deadlock recovery approaches can potentially result in high performance for low-load traffic, they suffer from latency and energy overhead in high-load traffic due to the frequent need to recover from deadlocks in the

network . Other research on chiplet systems, such as Kannan et al. [45] and Bharadwaj et al. [62], briefly touch upon routing algorithms and propose VC-based deadlock avoidance with unbalanced utilization of VCs. However, they also do not address VL and HL faults in their considerations.

### 3.2.3 Fault-Tolerant Routing in Conventional 2D Networks

Several fault detection techniques have been proposed for NoCs, such as those in [63] and [64]. To tolerate faults in 2D NoCs, increasing path redundancy in routing proves to be efficient. Ebrahimi et al. [65] introduced a deadlock-free routing approach that enhances path redundancy. Meanwhile, VCs are utilized in [66] and [57] to tolerate router and link faults. NARCO [67] generates redundant packets for fault tolerance, employing odd–even (OE) and inverted odd–even (IOE) turn models. OE+IOE [68] and NS-FTR [69] enhance reliability by detouring around faults and proactive packet replication. However, these approaches neglect load balancing over VCs, impacting traffic congestion.

Generating redundant packets has also been used to tolerate faults in 2D NoCs. In NARCO [67], if the fault rate is above a threshold, for each original packet, a redundant packet is generated and routed to improve communication reliability. Two VCs are utilized to ensure deadlock-freedom where an odd–even (OE) turn model is employed in one VC and an inverted odd–even (IOE) turn model is used in the other VC. The original packet is sent using the OE turn model, while the redundant packet is sent using the IOE turn model. Moreover, to reduce the possibility of encountering a fault, the routing uses an idea of neighbor awareness, in which a router selects a path according to downstream routers' fault status. However, there are two drawbacks to this approach: 1) under a low fault rate, where replication is not triggered, faults are not tolerated, and 2) under high fault rates, where some of the network resources are already faulty and congestion is more likely, redundant packets significantly increase the possibility of network congestion and performance loss. OE+IOE [68] improves upon NARCO by detouring around a fault instead of replicating packets. Similar to NARCO, another fault-tolerant routing approach called NS-FTR [69] is presented in which two North-last and South-last turn models are used in separate VCs. NS-FTR

slightly improves reliability compared to OE+IOE because it replicates packets from the source router and proactively guarantees packet delivery for most fault scenarios. However, NARCO, OE+IOE, and NS-FTR employ extra VCs to only tolerate faults, while the load balancing over the VCs is neglected but required to improve traffic congestion.

Janfaza et al. [70] proposed a routing algorithm that can tolerate HL faults in 2D mesh-based NoCs. Moreover, the algorithm can handle router faults by treating a faulty router as a router with all faulty links. The proposed routing algorithm uses a backtracking phase for livelock avoidance and a store-and-forward mechanism to remove deadlock in the routing process. However, the routing algorithm is reactive to both livelock and deadlock circumstances, which can result in significant performance and energy losses, especially in high-load traffic scenarios.

### 3.2.4 Fault-Tolerant Routing in 2.5D Chiplet Systems

To the best of our knowledge, no previous work has specifically addressed the issue of VL and HL faults in 2.5D chiplet systems. Furthermore, the existing routing algorithms proposed for 3D NoCs, including both fully connected networks [65] and partially connected networks [30,46], cannot be directly applied to 2.5D chiplet systems due to two main reasons: 1) In 3D NoCs, packet routing can occur vertically from one layer to another (either up→down or down→up). In contrast, in a 2.5D network, packets follow a different routing pattern where they go down and then up (up→down→up). Consequently, an inter-chiplet packet in a 2.5D chiplet system requires two vertical routings and three intra-layer routings (on the source chiplet, interposer, and destination chiplet); 2) The connectivity between routers in 2.5D chiplet systems may be indirect, and the size of chiplets and interposer can differ, resulting in a more irregular topology compared to 3D NoCs.

In our prior work [23], we introduced DeFT, a fault-tolerant and deadlock-free routing algorithm designed for 2.5D integrated chiplet systems. DeFT improves redundancy in VL selection to tolerate VL faults while maintaining balanced vertical-link utilization in fault scenarios. However, DeFT relied on the assumption that each router had access to fault information from all routers, guiding its selection optimization. We propose *ReD* which is a significant extension of

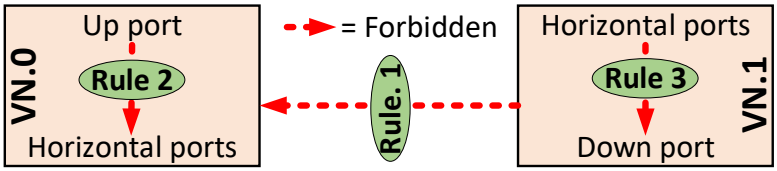



**Figure 3.2:** *ReD*'s rules for VN utilization and deadlock-freedom.

our proposed DeFT approach in [23]. *ReD* utilizes two VNs without imposing restrictions on VL selection where *ReD* can route both intra-chiplet packets and inter-chiplet packets on the interposer using either of the two VNs. *ReD*'s flexibility empowers it to tolerate HL faults without requiring additional virtual channels, effectively handling the majority of HL faults with minimal overhead. Moreover, we introduce a Hamiltonian-based routing mechanism that efficiently shares information about faulty or healthy VLs among all routers within the local chiplet or the interposer. This mechanism is designed to handle HL faults without extra control networks or VCs, further enhancing *ReD*'s fault tolerance capabilities.

In contrast to conventional 2.5D routing algorithms like MTR [52] and RC [54], the proposed *ReD* algorithm does not impose any restrictions on VL selection to achieve deadlock avoidance. Therefore, *ReD* has the capability to tolerate any VL fault patterns unless the network suffers from disconnected chiplets. Additionally, *ReD* incorporates a novel congestion-aware VL-selection strategy to enhance traffic distribution, particularly in the presence of VL faults. *ReD* shares the faulty/healthy status of VLs to all local routers employing a Hamiltonian-based routing, so the routers can proactively avoid selecting faulty VLs and distribute traffic over the remaining healthy VLs. The proposed Hamiltonian-based routing tolerates HL faults when sharing the updates. Moreover, *ReD* is able to tolerate HL faults by misrouting packets around the faulty HLs without using any extra VC.

## 3.3 *ReD*: Proposed Routing Algorithm

In this section, we discuss our VN-separation approach to enable deadlock-freedom. This VN-separation enables adaptation in VL selection to realize a fault-tolerant and congestion-aware

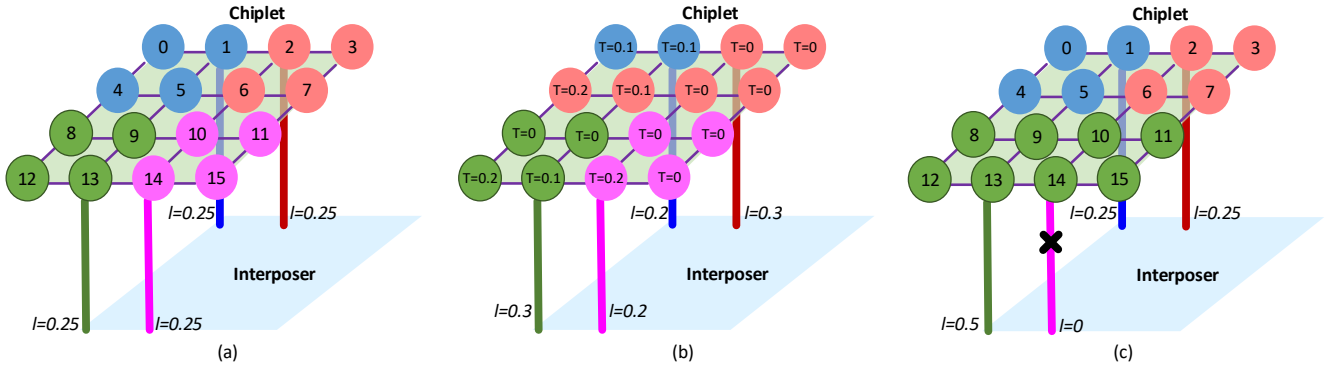



**Figure 3.3:** Examples for a VL selection in a chiplet with 16 routers: (a) A fault-free distance-based selection (closest VL is selected) under uniform traffic, (b) A good selection under non-uniform traffic, and (c) A distance-based selection with a VL fault and under uniform traffic, suffering from imbalanced traffic load on VLs. Here, $T$ and $l$ denote the inter-chiplet traffic rate of routers and VLs, respectively. Also, a router's color represents its selected VL.

VL selection, as discussed in Section 3.3.2. The effective VN separation in *ReD* also enables flexibility to effectively utilize VCs for the majority of packets in intra-chiplet/interposer routing. Therefore, in Section 3.3.3 we discuss *ReD*'s approach in tolerating HL faults without using any extra VCs. Subsequently, Section 3.3.4 outlines our methodology for sharing fault updates across routers. Lastly, in Section 3.3.5 we explore how *ReD* effectively guarantees deadlock- and livelock-freedom.

### 3.3.1 Proposed VN Separation and Deadlock-Freedom

The idea of employing VN separation for deadlock-freedom, in the context of 2D networks, is relatively old. However, employing VNs while considering hardware overheads and network latency, especially in 2.5D networks, is quite challenging. Consequently, existing routing techniques [52, 54] in chiplet systems have underestimated the efficiency of VN separation for deadlock-freedom. As we will show, *ReD* employs two VNs with negligible hardware overhead and fair utilization of VNs, ensuring its effectiveness even with a limited number of VCs, simplifying hardware requirements, and enhancing traffic distribution, all while tolerating both HL and VL faults.

*ReD* utilizes two VNs for deadlock-freedom, where at least one VC is required for each VN. Here, we assume one VC per VN, but the number of VCs can be increased without loss of gen-

erality. In cases where multiple VCs are utilized per VN, packets are distributed among the VCs in a round-robin manner to ensure a fair distribution of the load. We define “Down" port as the one going from a chiplet to the interposer, “Up" port goes from the interposer to a chiplet, and “Horizontal" ports (East, West, South, and North) are intra-chiplet and intra-interposer ports. The following rules, also shown in Fig. 3.2, are defined for deadlock-free VN utilization:

**Rule 1.** *Routing from VN.1 to VN.0 is forbidden, while packets can go from VN.0 to VN.1.*

**Rule 2.** *For packets in VN.0, routing from an Up port to Horizontal ports is forbidden.*

**Rule 3.** *For packets in VN.1, routing from Horizontal ports to a Down port is forbidden.*

To satisfy these three rules, for *inter-chiplet packets injected from the non-boundary routers*, VN.0 is assigned to the packets in the source router. This is because to route from the chiplet to the interposer, routing from Horizontal ports to down ports is required, while the routing is avoided in the VN.1 (Rule 3) and switching to the other VN is also avoided (Rule 1). Such VN.0 assignment is changed to VN.1 (VN.0$\rightarrow$VN.1) between the first (leaving the source chiplet) and the second (entering the destination chiplet) vertical routing. While these rules guarantee deadlock-freedom, VN utilization is also efficient due to the following theorems:

**Theorem 3.3.1.** *Both VNs can be assigned to intra-chiplet packets.*

*Proof.* Intra-chiplet packets do not use vertical ports (Up or Down ports), and hence they do not face any forbidden routings in VNs stated in Rules 2 and 3. □

**Theorem 3.3.2.** *Both VNs can be assigned to inter-chiplet packets for routing on the interposer.*

*Proof.* Packets entering the interposer are in VN.0. They can remain in VN.0 based on Rule 2 (Horizontal to Down is not forbidden in VN.0) or, based on Rule 1, switch to VN.1. □

**Theorem 3.3.3.** *Inter-chiplet packets on the source chiplet are free to be routed via any fault-free VLs toward the interposer.*

*Proof.* Packets from a source chiplet to the interposer need to go from the Horizontal ports to the Down port of the selected VL, and then from the Down port to the Horizontal ports of the interposer. Inter-chiplet packets on the source chiplet are in VN.0. When using any VL to go to the interposer, if the packet stays in VN.0, based on Rule 2, routing from Horizontal to Down ports, and Down to Horizontal ports is not forbidden. If the packet is switched to VN.1, based on Rule 1, it can go to a Down port and, based on Rule 3, it can go from Down to Horizontal ports. □

**Theorem 3.3.4.** *Inter-chiplet packets on the interposer are free to be routed via any fault-free VLs toward the destination chiplet.*

*Proof.* Routing from the interposer to the destination chiplet is done by going from Horizontal ports to the Up port of the selected VL, and then from the Up port to the Horizontal ports of the destination chiplet. Based on Rules 2 and 3, regardless of their VN, packets can go from Horizontal to the Up port. When using any VL to go to the destination chiplet, if a packet is in VN.0, based on Rule 1, it can be switched to VN.1 to go from Up to Horizontal ports. If the packet is in VN.1, based on Rule 3, it can go from Up to Horizontal ports. □

Based on Theorems 3.3.1 and 3.3.2, the proposed VN separation offers a careful balance of VC utilization to improve traffic distribution on VCs or tolerate HL faults (see Section 3.3.3), while also utilizing a small number of VCs (i.e., two VCs in total). For the cases where both VNs can be assigned, round-robin assignment can be used to balance the VN load. Moreover, according to Theorems 3.3.3 and 3.3.4, the choices for VL selection are maximized to tolerate faults on VLs. In addition, as we will discuss in Section 3.3.2, such flexibility in VL selection helps balance load traffic on VLs and improves the network latency. Algorithm 3 summarizes our VN-assignment strategy. Note that, as shown in Algorithm 3 and Fig. 3.1, an interposer router can also be a source router (e.g., for the packets injected by DRAMs). Algorithm 3 operates locally on each router with $O(1)$ time complexity, involving simple if-else conditions. Space complexity is negligible and included in our hardware analysis (see section 3.4.4). Moreover, since the algorithm is local, scaling up the system does not affect its input size, time, or space complexity.

**Algorithm 3** VN Assignment in *ReD*

```
1: if Current router is Source then
2:     if Source ∈ {interposer ∪ dest. chip ∪ boundaries} then
3:         Do round-robin assignment between VN.0 and VN.1
4:     else if Destination is on a different chiplet then
5:         Assign VN.0
6: else if Current router ∈ boundary routers then
7:     if going to the interposer then
8:         Do round-robin reassignment between VN.0 and VN.1
9:     else if coming from the interposer then
10:        Go to (remain in) VN.1
11: else
12:    Stay in the previously assigned VN
```

### 3.3.2 Proposed Fault-Tolerant Congestion-Aware VL Selection

#### Motivation for Fault-Tolerant and Traffic-Aware ReD

So far, we have presented our VN-based technique that provides high path redundancy while guaranteeing deadlock-freedom at a low cost. In this section, we focus on the fault tolerance and traffic challenges, which *ReD* can effectively address thanks to its flexibility, as discussed next.

First and foremost, it is essential to consider VL faults in 2.5D chiplet systems, which may arise due to inevitable microbump misalignment [71], electromigration in microbumps [72], and thermomigration in microbumps [71]. For the first time, *ReD* takes on this challenge that has been ignored in existing routing algorithms. To tolerate VL faults affecting inter-chiplet routing, *ReD* supports an adaptive VL selection where VLs are adaptively selected as intermediate destinations (see Section 3.2.2). In addition to VL faults, an efficient VL selection should take the traffic profile into account to properly distribute the load on VLs [73].

In Fig. 3.3, we show several examples in a chiplet with 16 routers to illustrate how VL selection can significantly affect traffic distribution and fault tolerance in 2.5D networks. In each example, VL selection of a chiplet with four VLs is shown, where the color of each router indicates its selected VL. Here, $T$ and $l$ denote the inter-chiplet traffic rate of routers and VLs, respectively. The traffic rate of a VL (i.e., $l$) is the sum of the traffic rates of the routers (i.e., $T$ of routers) selecting the VL. As is shown in Fig. 3.3(a), in the distance-based selection, where the closest VL

to the source router is selected, traffic is well distributed on VLs when the traffic generated by the routers is uniform and VLs are fault-free. In this example, four routers are sharing a VL, and as the generated traffic by the groups of routers is assumed to be uniform, the traffic on the VLs is also uniform. However, when applying distance-based selection under non-uniform traffic, it may lead to congestion on specific links. For example, assuming a distance-based selection like the selection in Fig. 3.3(a), the non-uniform traffic in Fig. 3.3(b) imposes half of the total load on the blue VL, leaving no load on the red VL. On the other hand, with an effective VL selection, as illustrated in Fig. 3.3(b), the loads on each VL will be $l_{\text{blue}}= 0.2$, $l_{\text{red}}=0.3$, $l_{\text{green}}= 0.3$, and $l_{\text{magenta}}= 0.2$, demonstrating a more favorable load distribution. Therefore, disregarding the traffic generated by routers during the VL selection process can result in a significantly imbalanced traffic distribution on the VLs.

In the example shown in Fig. 3.3(c), in the presence of a VL fault and under uniform traffic, a distance-based selection suffers from an imbalanced traffic load on VLs. In this selection scenario in which one VL is faulty (the magenta VL), a distance-based selection approach decides that, for example, eight routers should utilize the green VL while the two other VLs are utilized by four routers each. In this case, all the traffic of the faulty magenta VL is misrouted to the green VL, imposing unbalanced traffic on VLs as half of the whole load is now on the green VL. Such an unbalanced utilization can significantly increase the chance of congestion and degrade the overall performance. A more efficient selection in this example would be for Routers 8 and 11 to utilize the blue and the red VL, respectively. That way, for example, although the blue VL is farther to Router 8, packets injected by Router 8 can benefit from a lower traffic load on the blue VL and latency. Therefore, when dealing with a VL fault, it is important to assign the load to healthy VLs while considering the distance to the boundary router and ensuring well-distributed traffic across all the VLs. *ReD* considers both the VL faults and traffic profile to enable a fault-tolerant and congestion-aware VL selection, as discussed in the next section.

In addition to VL faults, HL faults in 2.5D networks can arise from various factors including physical defects [67], aging [74], and electromigration [67]. These HL faults can introduce reli-

ability concerns or even lead to a total loss of connectivity unless a fault tolerance mechanism is defined in the network. In the context of *ReD*, we specifically address HL faults and develop a fault-tolerant routing algorithm to handle both VL and HL faults. It is important to note that since router faults can be represented as certain link faults, our focus in *ReD* is solely on addressing link faults. To provide further clarification, a faulty router is regarded as one in which all of its links are deemed faulty.

**ReD's Vertical-Link Selection**

As discussed earlier, for inter-chiplet packets, two temporary destinations are selected by the VL-selection process. Based on a selection algorithm, a VL is selected and the packet is forwarded to the VL to route the packet towards its destination. To tolerate VL faults and improve congestion on boundaries of chiplets, during inter-chiplet routing, *ReD* supports an adaptive VL-selection approach where VLs are adaptively selected as intermediate destinations, as discussed below.

The VL-selection strategy in *ReD* includes a design-time (offline) step to analyze optimal VLs under different VL-fault scenarios and an online selection among such pre-analyzed VLs when a fault occurs. During design-time and considering VL-utilization balancing and distance (i.e., source to VL hop-count) minimization, we explore different VL selections for all the possible VL-fault scenarios. VL utilization balancing is important because it relieves over-utilized VLs from extra load. It not only improves the network latency but also increases the system reliability, as over-utilization of VLs can increase stress-migration-based faults [72]. Moreover, distance minimization can also improve energy consumption, as reducing the path length of a packet can reduce dynamic power dissipation in routers due to that packet, as well as its end-to-end latency. Considering these two objectives, the best resulting VL selections are analyzed and saved in look-up tables in routers to be utilized during run-time. This improves hardware complexity when calculating an optimized selection, providing *ReD* with an adaptive selection under different VL-fault scenarios. Here, the look-up table size and hardware cost are negligible compared to the overall size of a router (see Section 3.4.4 for cost analysis).

For any inter-chiplet packet, two VL selections are required: one on the source chiplet (towards the interposer) and one on the interposer (towards the destination chiplet). The first selection is influenced by packets from the same source-chiplet routers, whereas in the second VL selection, packets destined for the destination-chiplet routers play a role in the selection process. As both VL selections are performed in a similar manner, here we only discuss VL selection on the source chiplet for brevity (i.e,. the first VL selection).

Given a fault scenario and based on the VL selection on the source chiplet, i.e., the first VL selection, we consider the source-chiplet routers ($r$) which are included in the VL-selection process: $R_C = \{r_1, r_2, .., r_R\}$. Selection is done per chiplet, where $C$ is the chiplet and $R_C$ is the chiplet's routers. The chiplet is connected to the interposer using a set of fault-free VLs ($v$): $VL_C = \{v_1, v_2, .., v_V\}$ and $V$ is the number of fault-free VLs. The objective is to find a set of optimized VL selections for all the source-chiplet routers while balancing the VL utilization: $s^* = \{V_{r_1}, V_{r_1}, ..., V_{r_R}\}$, where $V_{r_i}$ is the VL selected for router $i$. In the following, we discuss how ***ReD*** finds the optimized VL selections (i.e., $s^*$) during design-time.

To balance VL utilization, the load on each VL should be close to the average load on all the VLs. The load on $VL_v$ depends on the inter-chiplet traffic (packet injection) rate among the routers that select $VL_v$. The load on $VL_v$ is:

$$l_v = \sum_{r \in R_C} T_r^{inter} \times U_v^r, \tag{3.1}$$

where $T_r^{inter}$ is the inter-chiplet traffic rate of router $r$. Also, $U_v^r = 1$ ($U_v^r = 0$) whenever router $r$ utilizes (does not utilize) VL $v$ for vertical routing. Based on (3.1), the average load over all the VLs is:

$$l_{avg} = \frac{1}{V} \sum_{v \in VL_C} l_v. \tag{3.2}$$

Considering (3.1) and (3.2), we define the load cost of VL $v$ as:

$$L_v = \left| \frac{l_v - l_{avg}}{l_{avg}} \right|. \tag{3.3}$$

The second objective in VL selection is distance minimization. Considering a mesh network, the Manhattan distance (i.e., hop count) between router $r$ to VL $v$ (on the same chiplet) is:

$$D_v^r = |x_r - x_v| + |y_r - y_v|\,, \tag{3.4}$$

where $x_r$ and $y_r$ are the coordinates of the router $r$, and $x_v$ and $y_v$ are the coordinates of the VL $v$. The distance cost of a VL $v$ to the routers that select $v$ is:

$$D_v = \sum_{r \in R_C} D_v^r \times U_v^r. \tag{3.5}$$

Based on (3.3) and (3.5), the overall cost of a selection set $s$ is:

$$C_s = \sum_{v \in VL_C} (\rho \times D_v) + L_v, \tag{3.6}$$

where $\rho$ can decide the importance of the load-balancing versus distance objectives. In our analyses (see Section 3.4), we experimentally found $\rho = 0.01$ to be efficient in ***ReD***. Based on the cost function $C_s$ in (3.6), an optimization search $O$ can be used to find the optimal selection set $s^*$ with the minimum cost $C_s^*$ (Algorithm 4):

$$s^* \leftarrow O(s \in S, \text{Objective} : \min(C_s)). \tag{3.7}$$

Here, $S$ denotes all the possible selection sets. We used an exhaustive search to address the optimization in (3.7) because the search space is small. In large networks with a large design space, efficient search algorithms should be used to reduce the optimization complexity. Our proposed VL-selection approach is summarized in Algorithm 4.

The algorithm iterates through selection sets and vertical links, finding the overall costs. Therefore, time complexity is $O(N_s.V)$, where $N_s$ is the number of selection sets and $V$ V is the total number of VLs in the chiplet system. The space complexity is negligible since each time one set is

**Table 3.1:** Abbreviations used in ReD's chapter

| Abbreviation | Description |
|---|---|
| $T_r^{inter}$ | Inter-chiplet traffic rate of router $r$ |
| $U_v^r$ | Whether router $r$ uses (does not use) VL $v$ |
| $l_{avg}$ | Average load over all VLs |
| $l_v$ | Load of VL $v$ |
| $L_v$ | Load cost of VL $v$ |
| $D_v^r$ | Manhattan distance of router $r$ and VL $v$ |
| $D_v$ | Distance cost of VL $v$ to the routers that select $v$ |
| $C_s$ | Overall cost of a selection set $s$ |
| $\rho$ | Importance of load balancing versus distance objectives |
| $S$ | Set of all possible selection sets |
| $s^*$ | Optimal selection set with minimum cost $C_s^*$ |

**Algorithm 4** VL Selection in *ReD*

```
for each s in all the possible selection sets (s ∈ S) do
    for v ∈ VL_C do
        L_v ← find load cost of v (using (3.3))
        D_v ← find distance cost of v (using (3.5))
    C_s ← find the overall cost of selection set s (using (3.6))
    if C_s < C_s^* then
        C_s^* ← C_s
        Update the saved selection sets (s^* ← s)
```

evaluated. Executing the algorithm offline ensures time complexity does not affect online performance. The algorithm is executed for different VL-fault scenarios at design-time and the selection sets ($s^*$) are saved in routers for run-time use. For the baseline system (shown in Fig. 3.1), where each chiplet has four VLs, there are 14 combinations of faults—$\binom{4}{1} + \binom{4}{2} + \binom{4}{3}$—and hence 14 VL addresses are saved in each router.In chiplet system scaling, while chiplet and VL numbers may increase, VLs per chiplet typically remain low to minimize fabrication cost. If a large number of VLs per chiplet are still desired, two options are suggested to improve the look-up table size: 1) Use the same solutions for each set of fault scenarios (approximation) at the cost of performance, and 2) employ online search at the cost of increased complexity.

When there is a change related to the VL fault status (i.e., a new VL fault occurs or a faulty VL becomes healthy), the *ReD* selection mechanism dynamically adapts the VL-selection process to ensure continued fault-tolerance and optimized traffic routing. During run-time, upon detecting a


**Figure 3.4:** Deadlock-free turn models used for intra-chiplet and -interposer routing in (a) the first VN and (b) the second VN.

fault, *ReD* leverages the pre-analyzed VL selections stored in the look-up tables within the routers, which were obtained from the design-time (offline) analysis considering various VL-fault scenarios, VL-utilization balancing, and distance minimization. This adaptive approach allows *ReD* to efficiently reroute affected packets by selecting the appropriate VLs from the pre-analyzed options, effectively avoiding VL faults and preventing potential congestion. The adaptive VL-selection process enhances the overall system reliability, reduces latency, and contributes to reduced stress-migration-based faults, thereby improving the 2.5D network's robustness and ensuring efficient packet delivery across chiplets.

### 3.3.3 Fault-Tolerant Routing to Handle Horizontal Link Faults

As discussed in Section 3.3.2, it is crucial to consider HL faults in interposer networks. These faults not only affect the sharing of VL updates, which will be discussed in the next section, but also impact the intra-chiplet and -interposer routing of network packets. One popular method for tolerating HL faults is to employ adaptive routing and divert traffic away from the faulty links. This adaptation in routing often requires the use of extra VCs (i.e., defining new VNs). Fortunately, *ReD* utilizes VNs already to ensure deadlock freedom (see Section 3.3.1), eliminating the need to add any extra VCs for fault tolerance. Our proposed VN-separation technique, as supported by Theorems 3.3.1 and 3.3.2, provides the flexibility to effectively utilize VNs for the majority of packets in intra-chiplet and -interposer routing. We also show this flexibility in Algorithm 3 (Lines 3 and 8), where packets can be routed in both VNs. We use this opportunity here to equip *ReD* with intra-chiplet and -interposer adaptive routing that enables the ability to tolerate HL faults.

We employ two different turn model routing algorithms, as depicted in Fig. 3.4, for routing in the first and second VNs of the intra-chiplet and -interposer networks. Solid black lines represent allowed turns, while dashed red lines represent avoided turns. The deadlock-freedom of these turn models will be discussed in Section 3.3.5. The turn model of the first VN enables adaptive routing for packets from West to East, while the turn model of the second VN allows adaptive routing for packets from East to West. We select these turn models to provide adaptive routing in the two East and West directions so that we can combine them and make a fully minimal adaptive routing. A fully minimal adaptive routing is defined as the routing in which all possible paths, each having a length equal to the Manhattan distance, can be taken. Moreover, we choose these turn models to support YX routing in both VNs, regardless of routing to the East or West. Therefore, for the small portion of packets that cannot be routed in both VNs, YX routing can be used as the default routing. To elaborate further, when such a packet is assigned to the first VN, it employs YX routing for routing to the West and minimal adaptive routing for routing to the East, whereas for the second VN, YX routing is utilized for routing to the East, and minimal adaptive routing is used for routing to the West.

When facing HL faults, there are two primary scenarios to consider: 1) a minimal routing approach to bypass the fault, and 2) a non-minimal routing strategy to get around the fault. We illustrate how to address these two scenarios in Figs. 3.5 and 3.6, respectively. These primary scenarios are discussed in more detail below.

**Minimal routing to bypass HL faults**

When a packet faces an HL fault and, according to the implemented routing, there is an alternative route available to bypass the fault without the need for taking a non-minimal path, a minimal fault-tolerant routing can be achieved. All instances of this scenario are presented in Fig. 3.5. In the first VN, which is adaptive-to-East (A2E), not only minimal adaptive routing is achieved toward the East direction but also YX routing is allowed toward the West direction. Therefore, when a packet is routed in the first VN, the well-designed turn model enables *ReD* to tolerate faults on both the East and West links of the router, as demonstrated in columns 2 and 4 of Fig. 3.5. However,

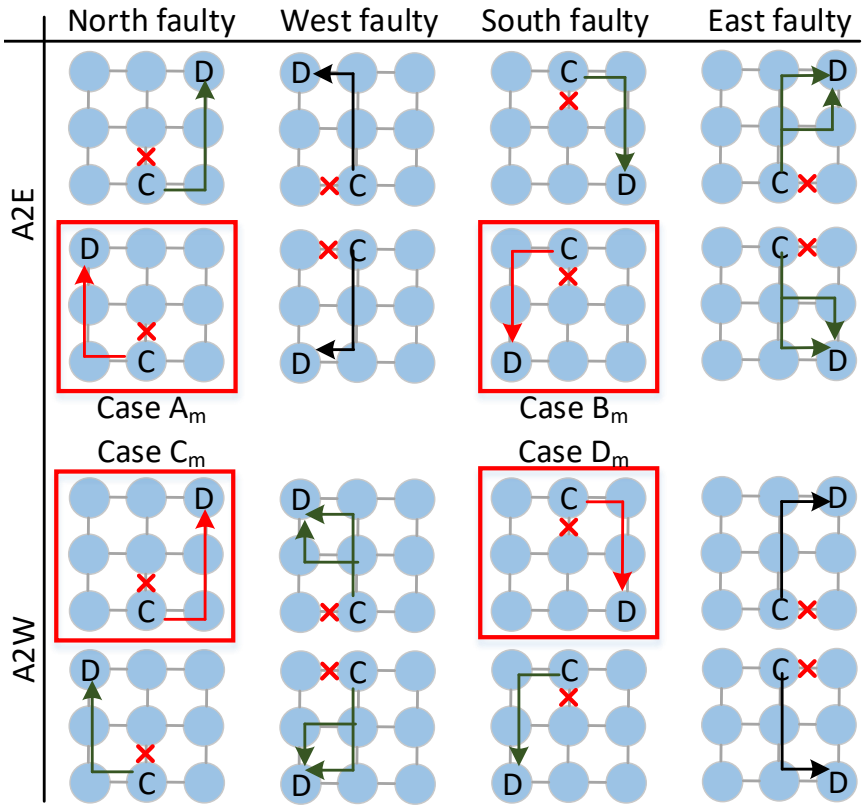



**Figure 3.5:** Minimal routing to bypass HL faults (A2E: adaptive-to-east VN, A2W: adaptive-to-west VN).

packets in the first VN are unable to bypass a fault on a North or South link if the destination is positioned on the West side of the source, denoted as cases $A_m$ and $B_m$. Similarly, packets in the second VN cannot bypass a faulty North or South link if the destination is situated on the East side of the source, labeled as cases $C_m$ and $D_m$. Therefore, to tolerate HL faults, instead of Round-Robin VN allocation shown in Algorithm 3 (Lines 3 and 8), here VNs are allocated to packets in a manner that minimizes cases $A_m$ to $D_m$ and, consequently, maximizes fault tolerance. To achieve this, the first VN is assigned to inter-chiplet packets if the destination is situated on the East side of the source, while the second VN is assigned when the destination is on the West side of the source. Additionally, when packets enter the interposer, if the destination boundary router is located on the West side of the source boundary router, they will be switched to the second VN. Conversely, if the destination boundary router is on the East side of the source boundary router, the packets will remain in the first VN.

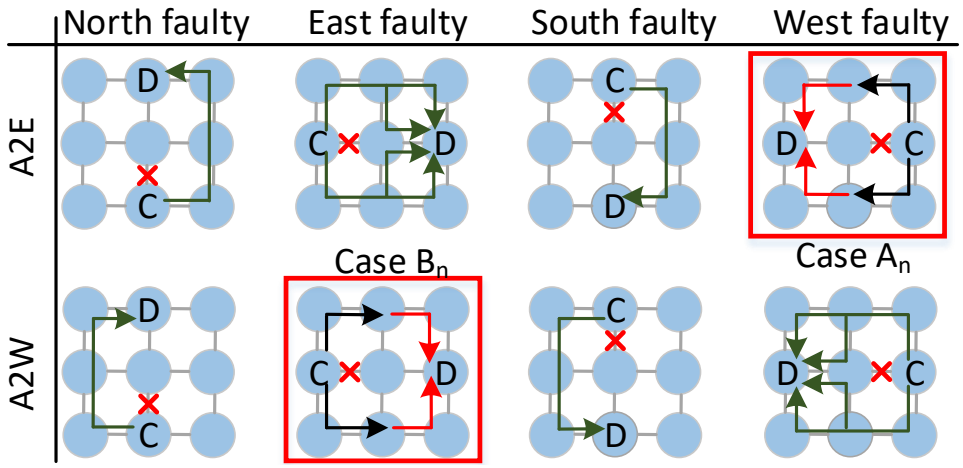



**Figure 3.6:** Non-minimal routing to get around HL faults (A2E: adaptive-to-east VN, A2W: adaptive-to-west VN).

#### Non-minimal routing to get around HL faults

To get around an HL fault, a non-minimal routing is required, when the current and destination routers are located on the same row or the same column. All possible cases are shown in Fig. 3.6. If a fault occurs on the North or South ports, it can be tolerated unless it is located on the East-most column in the first VN or the West-most column in the second VN. To maximize fault tolerance, packets in the first VN that are located on the East-most column are switched to the second VN when they encounter such a fault. Conversely, we prevent the situation where a packet in the second VN faces a fault on the West-most column. To avoid this, packets with destinations located on the West-most column are routed using YX routing if the packet is routed in the second VN. There are two cases in Fig. 3.6, cases $A_n$ and $B_n$, in which the fault cannot be tolerated. Similar to cases $A_m$ and $D_m$, cases $A_n$ and $B_n$ are minimized as VNs are allocated based on packet destinations, with inter-chiplet packets assigned to the first or second VN, and channel switching in the interposer is determined by the relative positions of source and destination boundary routers.

### 3.3.4 Sharing Fault Updates

To achieve fault-tolerant and traffic-aware VL selection, *ReD* relies on identifying faulty links. In our approach, updates related to HL faults are not shared; only a router is aware of the faults on its local HLs. This strategy minimizes the overhead of sharing HL fault updates, enabling packets

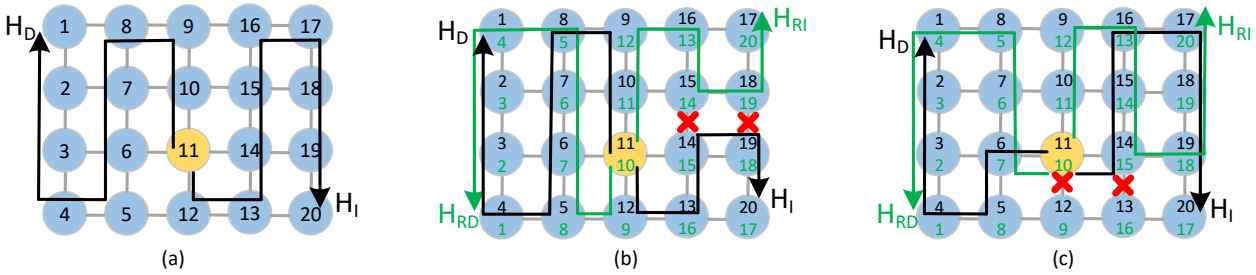



**Figure 3.7:** Proposed Hamiltonian-based approach for sharing faulty/healthy updates of VLs: (a) Hamiltonian-based routing is used to distribute information through increasing and decreasing IDs, routed in the first and second VNs respectively, (b) and (c) two Hamiltonian-based routings (green and black) are used for fault tolerance when sharing information, specifically designed to handle HL faults.

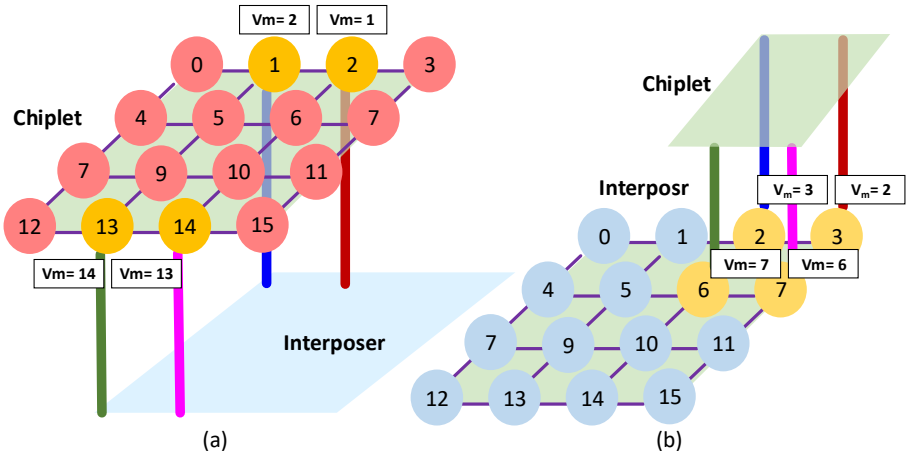



**Figure 3.8:** Spare VLs used for packets which failed to receive VL updates and are routed to a faulty VL: (a) Spare VL on chiplets and (b) Spare VL on the interposer.

to be detoured efficiently to avoid faulty HLs based on the rules in Figs. 3.5 and 3.6. On the other hand, *ReD* shares VL fault updates to proactively route packets to healthy VLs and avoid faulty VLs. We prioritize sharing VL fault updates because the number of VLs is much smaller than HLs, and not all routers have a VL locally. Additionally, as demonstrated in Section 3.3.2, VL faults significantly impact network traffic, making it crucial to distribute the load proactively among healthy VLs.

To facilitate sharing VL fault updates, *ReD* introduces a Hamiltonian-based routing technique, enabling the sharing of faulty/healthy status of VLs with all local routers within a chiplet or interposer. This approach enhances fault awareness and enables efficient traffic management to maintain fault tolerance and improve network performance. Fig. 3.7 shows the proposed Hamiltonian-

based approach for sharing the updates. In Fig. 3.7(a), we demonstrate the Hamiltonian-based routing scheme. Each router is assigned an ID from West to East, with increasing IDs from North to South in even columns and from South to North in odd columns. We employed the Hamiltonian-based routing scheme to distribute the updates to higher and lower ID numbers in the second and first VNs, respectively. A Hamiltonian path is defined as a path that visits all routers exactly once. Therefore, this routing strategy ensures an efficient flow of the updates within the chiplet/interposer while leveraging the advantages of the Hamiltonian path. As is shown in Fig. 3.7(a), the proposed Hamiltonian-based routing guarantees that all the routers in the chiplet/interposer receive the updates of VLs, unless there is a faulty HL on the Hamiltonian path of the chiplet/interposer. To send updates, the local router generates two 1-flit packets and sends them through the increasing path ($H_I$) and the decreasing path ($H_D$). This approach allows us to update all routers in the Hamiltonian path while utilizing the existing VNs without the need for additional ones.

We designed our Hamiltonian-based routing to be compatible with our main routing without using any extra VCs. It is evident that the allowed turns shown in Fig. 3.4 support our Hamiltonian-based routing presented in Fig. 3.7 when routing $H_I$ in the first VN and $H_D$ in the second VN. Thus, our Hamiltonian-based routing effectively sends the update packets to all routers and updates the healthy/faulty status of VLs without creating any deadlock, unless there is a faulty HL on the Hamiltonian path.

As discussed in Section 3.3.2, it is important to consider the potential faults in HLs, in addition to VLs. To ensure robustness against faults in sharing the updates, we employ two distinct Hamiltonian paths with different directions for propagating the updates. While alternative Hamiltonian paths could be considered, we opt for these specific paths to align with our turn models (see Fig. 3.4), ensuring seamless compatibility and obviating the need for additional VCs to accommodate this Hamiltonian routing. To illustrate this approach, we present two examples in Figs. 3.7(b) and 3.7(c). In these figures, we introduce another Hamiltonian path and assign different IDs to the routers (shown in green). In the even columns, the router IDs increase from North to South, while in the odd columns, the router IDs increase from South to North. Moreover, in each Hamil-

tonian path, the packet takes a detour by routing in the direction of a non-faulty link within the Hamiltonian-compatible route. For instance, from the yellow router (router 11 on the black path), when routing through the $H_I$ path, if the South port is faulty, the packet is routed to the East port to get router 14 with a larger ID number instead of 10. This means that when routing through the increasing path ($H_I$), the route going to a larger ID is taken, while when routing through the decreasing path ($H_D$), the route to a smaller ID is taken. Therefore, with two Hamiltonian-based paths and the simple fault-tolerant detouring, most of the HL faults can be tolerated, when sharing VL updates. Note that in the last subsection (Section 3.3.3), we elaborated on our methodology for handling HL faults when routing network packets, extending beyond updates alone.

While the proposed Hamiltonian-based routing effectively tolerates the majority of HL faults, as demonstrated in section 3.4, rare instances may arise where certain routers fail to receive updates. This is more prevalent in scenarios of exceptionally high fault rates or when the faults are region-specific. For instance, in Fig. 3.7(c), two routers do not receive updates. To address this, we implement a mechanism wherein routers equipped with VL routers (i.e, the routers connected to VLs) store the address of an additional spare VL. Consequently, in cases where update reception fails and a faulty VL is encountered, packets can be rerouted to the alternate VL. Importantly, the choice of the spare VL must align with our turn models, as illustrated in Fig. 3.4, to ensure compatibility with the misrouting scenario. In our case study, we establish the spare VL for chiplets and the interposer, as depicted in Fig. 3.8. It is worth noting that for interposer-bound packets, we consider switching or maintaining the second VN after any misrouting to a spare VL. Therefore, the misrouting on chiplets must conform to the turn model of the first VN in Fig. 3.4, while those on the interposer adhere to the turn model of the second VN in Fig. 3.4.

We summarized our Hamiltonian-based technique in Algorithm 5. For brevity, we show only the primary Hamiltonian paths in the algorithm. Each router only needs to be aware of the ports required to reach the next routers on the Hamiltonian paths. Please note that in the algorithm, the term "Port," refers to the port's address. Our Hamiltonian-based routing does not require any additional ports for sharing updates. Moreover, the terms "spare increasing Hamiltonian"

**Algorithm 5** Hamiltonian-based routing in *ReD*

```
1: P_{H_I}: Port to next router on increasing Hamiltonian
2: P_{H_D}: Port to next router on decreasing Hamiltonian
3: P_{SH_I}: Port to next router on spare increasing Hamiltonian
4: P_{SH_D}: Port to next router on spare decreasing Hamiltonian
5: Input: Fault sharing flit (F-flit)
6: Output: Out port
7: if Incoming flit is F-flit then
8:     Update fault table of current router based on F-flit
9:     if F-flit is increasing Hamiltonian then
10:        if P_{H_I} is not faulty then
11:            Out port ← P_{H_I}
12:        else
13:            Out port ← P_{SH_I}
14:    else if F-flit is decreasing Hamiltonian then
15:        if P_{H_D} is not faulty then
16:            Out port ← P_{H_D}
17:        else
18:            Out port ← P_{SH_D}
```

and "spare decreasing Hamiltonian" denote the additional port addresses strategically employed along the increasing and decreasing Hamiltonian paths, respectively. These spare port addresses contribute to enhancing fault-tolerance capabilities.The algorithm, with $O(1)$ time complexity and minimal space usage, remains unaffected by system size, except for the VL selection table storage, which we previously discussed in section 3.3.2.

### 3.3.5 Deadlock- and Livelock-Freedom

#### Deadlock-freedom

*ReD* is deadlock-free in any 2.5D chiplet system where each chiplet is locally deadlock-free, because of two main reasons: 1) there is no inter-chiplet cyclic dependency in VN.0 and VN.1. In VN.0, based on Rule 2, routing from Up port to Horizontal ports is avoided. In VN.1, based on Rule 3, routing from Horizontal ports to Down port is avoided; and 2) there is no cyclic dependency between VNs because Rule 1 avoids routing from VN.1 to VN.0. Considering these reasons, *ReD* is deadlock-free, if no cycle occurs in chiplets and the interposer locally.

**Table 3.2:** Simulation setup of ReD.

| | | | |
|---|---|---|---|
| Number of chiplets | 4,6,8, and 12 | VL per chiplet | 4 |
| Packet size | 8 flits | Flit width | 32 bits |
| Buffer size | 4 flits | Number of VCs | 2 |
| Router frequency | 2 Ghz | VL/HL latency | 1 cycle |
| Simulation time | 1 million cycles | Warmup time | 10K cycles |

In *ReD* routing, both chiplets and the interposer are locally deadlock-free. This is achieved by utilizing two turn models for intra-chiplet and -interposer routing, both of which are known to be deadlock-free. The first turn model (see Fig. 3.4(a)) is similar to a North-last turn model, rotated counterclockwise by 90 degrees. Similarly, the second turn model (see Fig. 3.4(b)) is also like a North-last turn model but rotated clockwise by 90 degrees. The deadlock-freedom of the North-last turn model has been proven in the context of a 2D mesh [60]. In the North-last turn model, North-East and North-West turns are avoided. As the North-last turn model is deadlock-free, the rotated turn models employed in the first and second VCs of *ReD* routing are also deadlock-free.

**Livelock-freedom**

*ReD* is livelock-free in any 2.5D chiplet system, given that both chiplets and the interposer exhibit local livelock-freedom. The inter-chiplet packets in *ReD* follow a specific routing path: source chiplet $\rightarrow$ interposer $\rightarrow$ destination chiplet. This routing is facilitated by two intermediate destinations (see Section II-A). Since packets do not route using any intermediate chiplet, packets in *ReD* are routed within a finite number of hops, ensuring that livelocks do not occur, provided that chiplets and the interposer maintain local livelock-freedom. *ReD* is also livelock-free in intra-chiplet and -interposer routing. It is important to note that the minimal routing approach is followed for most cases in *ReD*, ensuring livelock freedom in such cases. However, in instances where packets encounter HL faults and the destination is in the same row or column, non-minimal routing may be employed, as illustrated in Fig. 3.6. As illustrated in this figure, packets do not use backward routing, and after misrouting, the rest of routing is done minimally. Therefore, the packets are routed within a finite number of hops and livelock cannot occur.

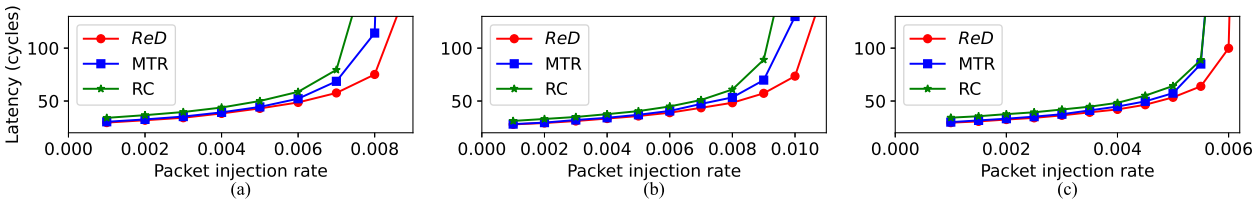



**Figure 3.9:** Average latency comparison among *ReD*, MTR, and RC routing algorithms when applied to 2.5D networks with four-chiplet system. The comparison is conducted under different synthetic traffic patterns: (a) Uniform, (b) Localized, and (c) Hotspot.

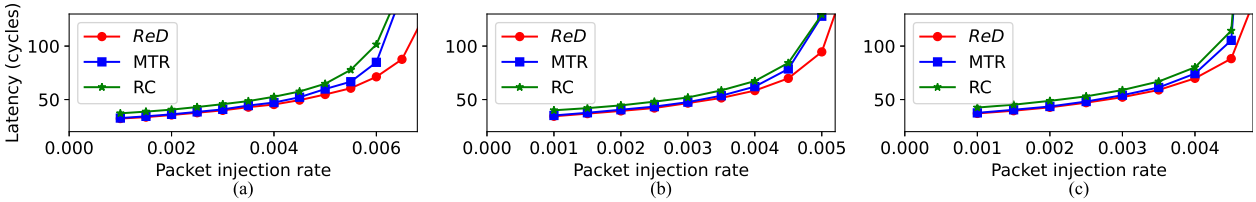



**Figure 3.10:** Average latency comparison under different system sizes and Uniform traffic. (a) 6 chiplets, (b) 8 chiplets, and (c) 12 chiplets.

## 3.4 Evaluation and Simulation Results

### 3.4.1 Simulation Environment and Configuration

We employed the Noxim simulator [75] for our network simulations, GEM5 [38] as our system simulator, and Cadence Genus for estimating the area and power consumption of the 2.5D networks considered. We enhanced the Noxim simulator, which is a cycle-accurate simulator for NoCs [75], to support 2.5D chiplet systems, and the system depicted in Fig. 3.1. In this configuration, each chiplet is connected to the interposer utilizing four bidirectional VLs. The decision to connect each chiplet with four VLs instead of a fully connected setup was motivated by the goal of reducing fabrication costs [52]. To further reduce VLs and enhance fabrication-cost optimization, serialization techniques can be employed [76]. According to [52], for a 4×4 chiplet system, placing the four VLs on the top and the bottom rows of the chiplet is an optimal choice, as shown in Fig. 3.1, when considering hardware complexity and network latency. It is important to note that the efficiency of *ReD* is not impacted by the the placement and density of VLs, as it does not rely on turn models to achieve inter-chiplet deadlock-freedom. In addition to the baseline four-

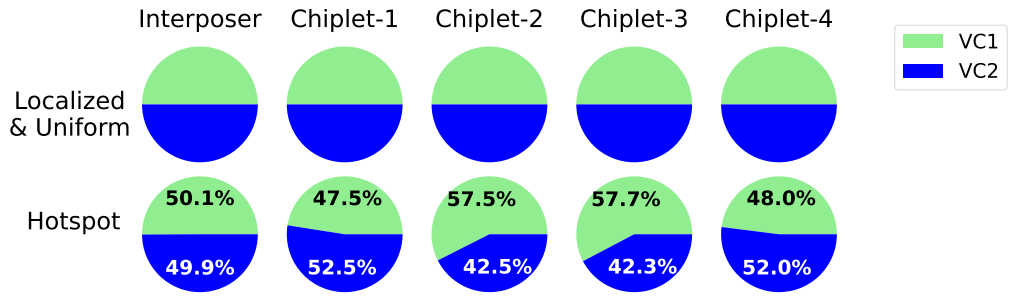



**Figure 3.11:** VC utilization of *ReD* under Uniform, Localized, and Hotspot traffic patterns.

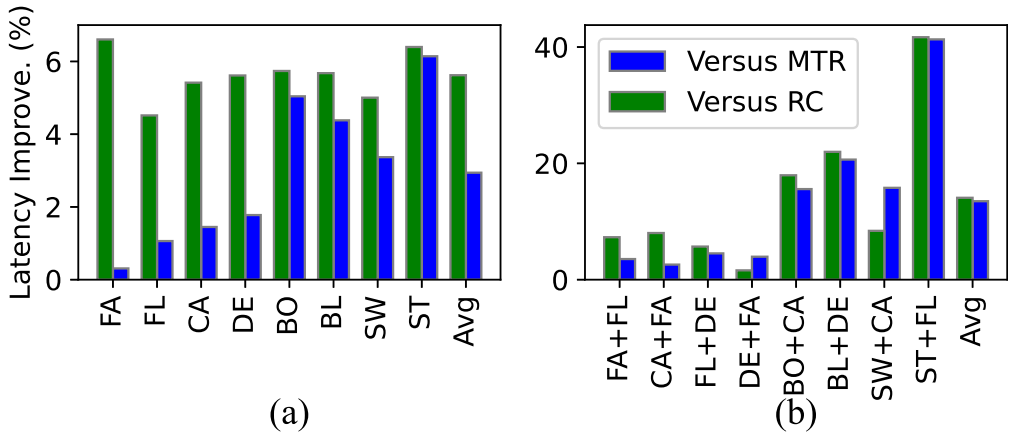



**Figure 3.12:** Latency improvement under real-application traffic using (a) a single application, and (b) two applications simultaneously.

chiplet system, we conducted simulations on six-chiplet, eight-chiplet, and twelve-chiplet systems to examine how the efficiency of *ReD* scales with system size.

We compared the performance of *ReD* against state-of-the-art routing algorithms for chiplet-based systems, MTR [52] and RC [54]. While MTR and RC do not impose any specific requirements on the number of VCs, we utilized two VCs in all algorithms to ensure a fair comparison with *ReD* (considering a single VC would result in inferior performance for MTR and RC). More details of our simulation setup is included in table 3.2. For offline VL-selection optimization, we focused on Uniform traffic as the most pessimistic assumption, although our simulations encompassed different traffic scenarios. Taking traffic information into account during the offline optimization process would yield further performance improvements.

### 3.4.2 Latency Analysis

The analysis of latency for synthetic traffic is depicted in Fig. 3.9. We employed three synthetic traffic patterns: Uniform, Localized, and Hotspot. In the Uniform traffic pattern, packets are randomly generated between all feasible source-destination pairs. This modeling simulates a scenario where data transmission is evenly distributed across the network, representing a balanced and unbiased traffic flow. The Hotspot traffic pattern involves a subset of nodes that receive a significantly higher volume of traffic than others. This setup facilitates the assessment of congestion handling. For the Localized traffic pattern, we assume uniform communication within chiplets, with nodes on the same chiplet exhibiting higher communication compared to nodes on different chiplets. This configuration emphasizes intra-chiplet communication over inter-chiplet communication. As shown in Fig. 3.9, *ReD* exhibits the lowest average latency in all traffic patterns compared to MTR and RC due to its balanced VL selection and VC utilization strategies. In the Localized traffic scenario, 40% of the packets are categorized as intra-chiplet packets, where both the source and destination are located on the same chiplet. As shown in Fig. 3.9(b), *ReD* demonstrates low latency in this scenario by effectively distributing the VC utilization for intra-chiplet packets, as supported by Theorem 3.3.1. For Hotspot traffic, presented in Fig. 3.9(c), *ReD* shows a slightly lower improvement due to the restriction of incoming packets to use only the second VC, leading to back-pressure. However, it is worth noting that our simulations employed a relatively high rate of hotspots (3 hotspot points with a 10% rate each).

In Fig. 3.10, we analyze the performance of six-chiplet, eight-chiplet, and twelve-chiplet systems, and it is noteworthy that *ReD* continues to exhibit high performance compared to the other approaches. The consistent performance gains observed across different system sizes highlight *ReD*'s scalability and suitability for large-scale chiplet-based architectures. *ReD* demonstrates performance improvement with larger systems, even with a 12-chiplet system (192 cores). However, it exhibits slightly less improvement compared to MTR and RC as system size increases. To understand the reason for this, we conducted an analysis of the load distribution; however, due to space constraints, we have chosen not to include it for brevity. Nevertheless, it is important to note that

in the comparison of load distributions across different system sizes, particularly in the 12-chiplet system, congestion becomes more pronounced on the interposer rather than the VLs. This shift can be attributed to the larger size of the interposer in the 12-chiplet system, leading to congestion at its center when managing communication among the twelve chiplets. This results in a slightly lesser improvement in average latency in larger systems for *ReD* compared to MTR and RC.

The distribution of VC utilization for the synthetic traffic patterns is presented in Fig. 3.11. In Uniform and Localized traffic, the VC utilization is balanced with a deviation of less than 0.4%, which is illustrated in the same chart. In Hotspot traffic, despite the relatively high rate of hotspots, the deviation in VC utilization remains below 8%. The balanced VC utilization achieved by *ReD* contributes to its low average latency while effectively avoiding deadlock situations.

To incorporate real-application traffic, we generated traffic using GEM5 [42] from PARSEC benchmarks [38] and simulated it using our chiplet-based Noxim simulator. The simulations were conducted in full-system mode with 64 x86 cores, four coherence directories, and four shared L2 cache banks (each core also has a private L1 cache). Fig. 3.12 illustrates the latency improvement achieved in eight PARSEC applications (blackscholes, bodytrack, canneal, dedup, facesim, fluidanimate, streamcluster, and swaptions), where the first two letters of each application are represented on the x-axis. In order to assess network performance under higher congestion, we also simulated the scenario of two applications running simultaneously, with each application utilizing 32 x86 cores and two shared L2 cache banks (as shown in Fig. 3.12(b)).

On average, *ReD* exhibits greater improvement when multiple applications are considered, mainly due to the increased likelihood of network congestion in such cases, which *ReD* is able to address more efficiently, compared to MTR and RC. In Fig. 3.12(b), the two-application combinations are arranged based on the traffic load, ranging from low (FA+FL) to high (ST+FL). Notably, for high traffic loads, *ReD* demonstrates a significant improvement of up to 40% compared to both MTR and RC.

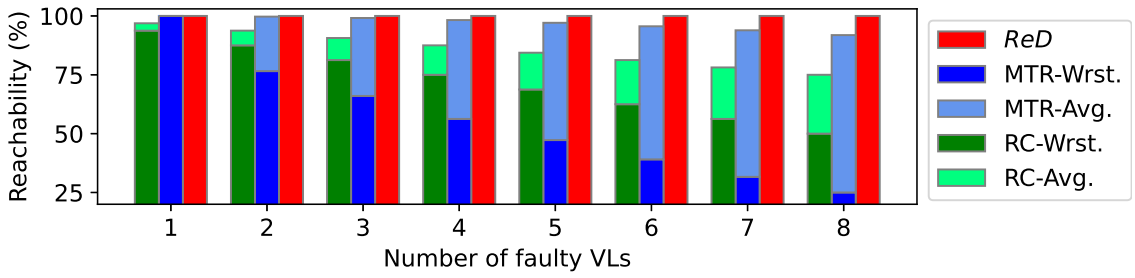



**Figure 3.13:** Reachability in the presence of VL faults in a system with four chiplets. Note that *ReD*-Wrst. and *ReD*-Avg. are the same (both shown by *ReD*).

### 3.4.3 Fault-Tolerance Analysis

#### VL faults

In order to evaluate the fault tolerance of *ReD*, we analyze network reachability in the presence of faults, following a similar approach as in the study by [30]. In fault scenarios, reachability is defined as the proportion of successfully routed packets to the total number of injected packets. Reachability in the presence of VL faults is shown in Fig. 3.13(a). We conducted fault injections for all possible combinations of fault patterns, excluding those that result in complete disconnection of chiplets (i.e., when all VLs of a chiplet are faulty). The figure presents both average and worst-case reachability for injected fault rates ranging from 3.125% to 25%, corresponding to 1 to 8 faulty VLs. In the average case, *ReD* enhances network reachability by 3.1–25% compared to RC and by 0–8.1% compared to MTR. In the worst case, *ReD* improves network reachability by 6.25–50% compared to RC and by 0–75% compared to MTR. Fig. 3.14 shows network reachability as the system scales up, comparing 4-chiplet and 12-chiplet systems. In both system sizes, *ReD* achieves 100% reachability for the given fault injection rates, while MTR and RC fail to achieve 100% reachability even at the lowest fault injection rate. The restricted turns in MTR and the permission network of RC limit their VL selection and, consequently, their fault tolerance against faulty VLs.

Fig. 3.15(a) presents the effect of VL faults on latency. The analysis excludes MTR and RC due to their inability to ensure complete reachability in fault scenarios. Two fault injection rates are considered where four faults correspond to 12.5% and eight faults correspond to 25% of VL links being faulty. Furthermore, in addition to utilizing *ReD* with our proposed VL selection (*ReD* in

Fig. 3.15(a)), we incorporate the average latency of *ReD* with distance-based VL selection strategies (*ReD*-Dis.) as commonly employed in 3D networks [30]. *ReD*-Dis. refers to routing using *ReD* while selecting VLs based on distance instead of our proposed method. Notably, *ReD* demonstrates significantly lower latency compared to distance-based VL selections at fault injection rates of 12.5% and 25%, respectively. As can be seen, in the higher fault injection rate, *ReD* shows more improvement. This confirms that *ReD* is able to handle the higher congestion due the smaller number of healthy VLs.

**HL faults**

In Figs. 3.15(b) and 3.15(c), we show the effect of HL faults on network latency, when the faults are on the chiplet and interposer, respectively. We also demonstrate the impact of combined faults in Figs. 3.15(d). In this scenario, we examined uniform combinations of VL faults, chiplet HL faults, and interposer HL faults. HL faults have a more pronounced impact on the interposer, as shown in Figs. 3.15(c), primarily due to the heightened traffic congestion it experiences. This congestion is further exacerbated by the interposer's role in managing global traffic flow among multiple chiplets. In the fault-free case, we show two versions of ***ReD***: $ReD_{XY}$ and $ReD_{AdaptiveXY}$. In $ReD_{XY}$, XY routing is used for the inter-chiplet/interposer routing, while $ReD_{AdaptiveXY}$ employs our adaptive routing, which is discussed in Section 3.3.4, to tolerate HL faults. Our $ReD_{AdaptiveXY}$ imposes small overhead on latency under Uniform traffic because it separates traffic into VCs based on source and destination location. This separation can make the VC utilization slightly imbalanced. However, the VC separation enables adaptive routing which not only provides us with the opportunity to handle the HL faults but also gives the routing the adaptation to avoid traffic congestion. Therefore, in the fault-free case, $ReD_{AdaptiveXY}$ offers high performance when considering non-uniform traffic (e.g., Hotspot). As we discussed in Section 3.3.4, HL faults not only can negatively affect the routing process of the NoC packets, but also they prevent the sharing of faulty/healthy status of VLs. Therefore, as we discussed, our proposed redundant Hamiltonian-based routing, which shares the updates in a different direction, significantly improves fault tolerance against the HL faults when sharing the updates.

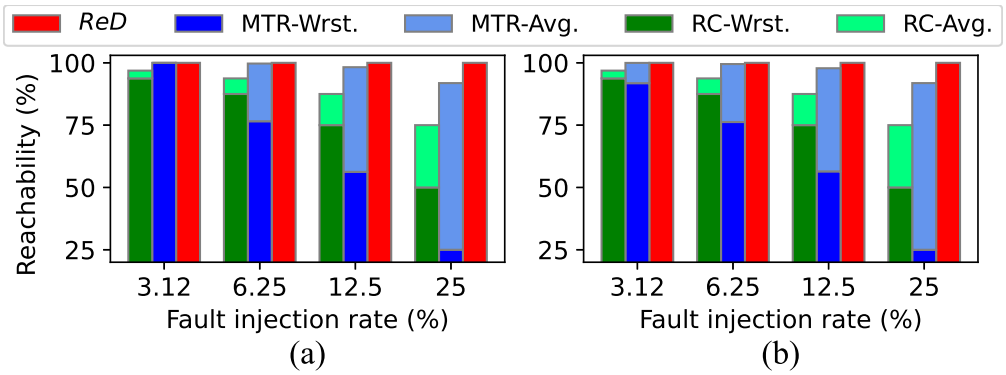



**Figure 3.14:** Reachability in the presence of VL faults in a system with (a) four chiplets (total VLs=32), and (b) twelve chiplets (total VLs=96). Note that *ReD*-Wrst. and *ReD*-Avg. are the same (both shown by *ReD*).

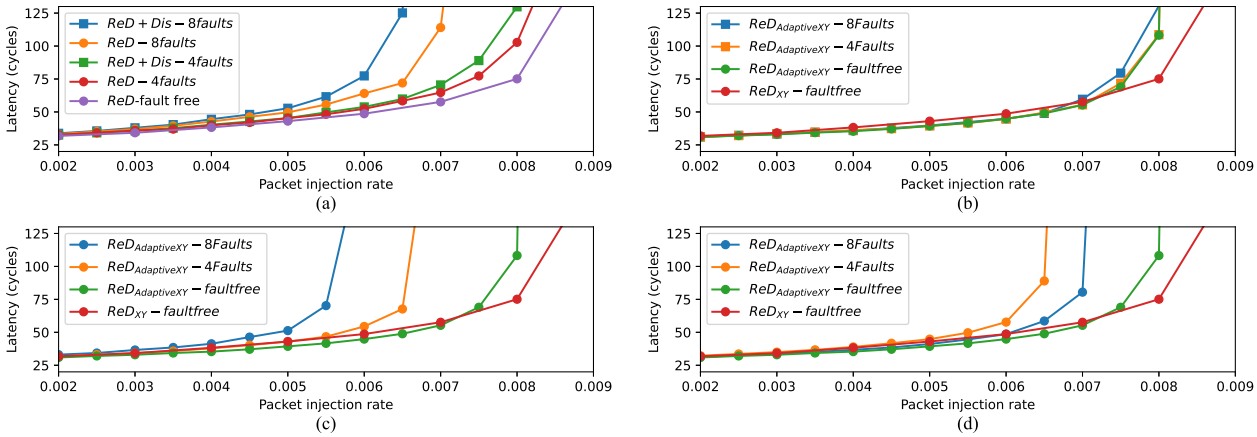



**Figure 3.15:** Average latency in *ReD* with different VL-selection strategies, fault-tolerant approaches, and fault-injection rates for a four-chiplet system. (a) VL faults, (b) HL faults on chiplets, (c) HL faults on the interposer, and (d) uniformly combined faults on VLs, HL faults on chiplets, and HL faults on the interposer.

Fig. 3.16 illustrates the average percentage of updated routers when sharing the status of faulty/healthy VLs for chiplets and the interposer. The analysis focuses on the effectiveness of updating the routers in the network when informing them about the status of VLs in terms of their faultiness or healthiness. Please note that an update is triggered, starting from the router connected to the VL, in an event-driven scheme: when a healthy VL becomes faulty or when a faulty VL becomes healthy. As detailed in Section 3.3.4, it is evident that HL faults can impede the sharing mechanism. Therefore, this analysis focuses on fault tolerance against HL faults when sharing VL fault updates. The percentages reflect the proportion of routers that receive updated information regarding the VLs. In the figure, “Hamiltonian" represents the simple Hamiltonian-based routing

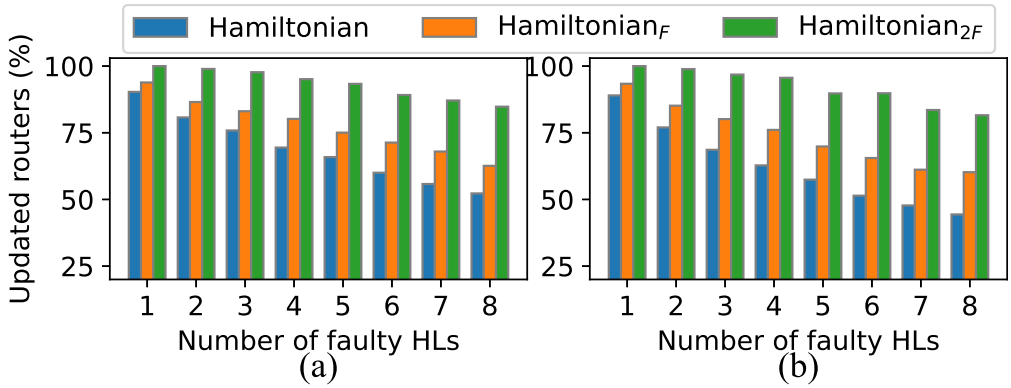



**Figure 3.16:** Average percentage of updated routers when sharing the status of faulty/healthy VLs for (a) chiplets and (b) the interposer.

**Table 3.3:** Area and power analysis of ReD, MTR, and RC

| | MTR | RC$_{\text{non-bndry}}$ | RC$_{\text{bndry}}$ | *ReD* |
|---|---|---|---|---|
| Router area ($\mu$m$^2$) | 45878 | 46663 | 51984 | 46730 |
| Norm. router area | 1 | 1.017 | 1.133 | 1.019 |
| Router power (mW) | 11.644 | 11.76 | 12.841 | 11.802 |
| Norm. router power | 1 | 1.009 | 1.102 | 1.014 |

with one path in the increasing Hamiltonian path and one path in the decreasing Hamiltonian path. "Hamiltonian$_F$" represents Hamiltonian routing that incorporates the selection of a spare path in the case of HL faults. Moreover, "Hamiltonian$_{2F}$" represents our proposed Hamiltonian routing with two Hamiltonian routes sharing the updates in two different direction, as shown in Fig. 3.7. As can be seen, "Hamiltonian$_{2F}$" is able to offer significantly high fault tolerance under HL-fault scenarios. Even with eight faulty HLs, which is a high fault rate, more than 80% of the routers are updated. Therefore, the proposed Hamiltonian-based routing ("Hamiltonian$_{2F}$") is efficient in sharing the faulty/healthy status of VLs, although it does not need extra VCs or any control network to share the updates. It is worth noting that we used "Hamiltonian$_{2F}$" across all the other experiments detailed in this chapter. Regular packets are delayed to the next cycle when F-flits are being shared, but since F-flits occur less frequently, the impact on regular packet routing is negligible.

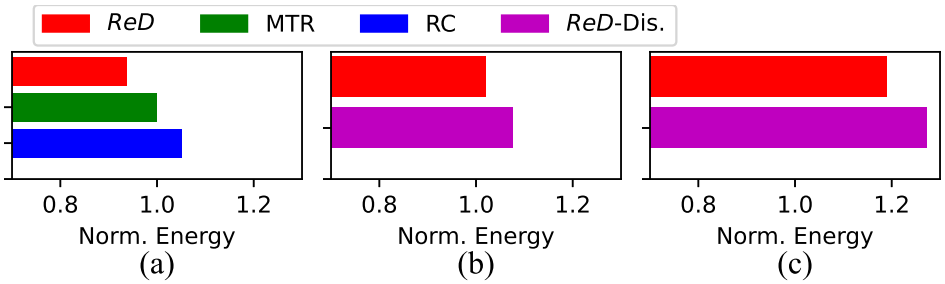



**Figure 3.17:** Normalized energy comparison for a four-chiplet network with (a) no faults, (b) 12.5% faulty VLs, and (c) 25% faulty VLs.

### 3.4.4 Hardware and Power Analysis

We utilized Cadence Genus and ORION 3.0 [77] for estimating the area and power of the router using the 45-nm technology. Table 7.1 presents a comparison of the area and power estimates for a six-port *ReD* router compared to six-port MTR and RC routers. It should be noted that the hardware implementations of non-boundary and boundary routers differ in RC, so we provide separate values in the table. The results in Table 7.1 demonstrate that *ReD* incurs a hardware overhead of less than 2% and a power overhead of less than 1%, relatively, when compared to related existing work. This minimal overhead includes all the techniques and functionalities used in *ReD*: the main fault-tolerant routing, the logic for the VN-assignment algorithm, the logic for sharing VL updates, and look-up tables used for fault-tolerant VL selection.

### 3.4.5 Energy Analysis

Fig. 3.17(a) shows energy analysis normalized to MTR for 100K packets injected under Uniform traffic and 0.01 packets/node/cycle injection rate. *ReD* achieves lower energy by 6.1% and 10.7% compared to, respectively, MTR and RC as it can route the injected packets faster. In addition, RC uses extra buffers for deadlock-freedom, which increases energy costs. MTR sends packets to non-minimal VLs to guarantee deadlock-freedom, while *ReD* is free to select any VLs. Also, under 12.5% and 25% VL-fault scenarios, shown in Figs. 3.17(b) and 3.17(c), compared to distance-based (*ReD*-Dis.) VL selection, *ReD* with its congestion-aware VL selection is able to achieve lower energy.

## 3.5 Conclusion

This chapter introduced a fault-tolerant routing algorithm, named *ReD*, specifically designed for 2.5D chiplet networks. The main objective of *ReD* is to provide a deadlock-free and fault-tolerant routing solution, while minimizing traffic congestion and area overhead. *ReD* achieves VC-based deadlock-freedom by allowing packets to freely choose any VL and efficiently balances VC utilization. This flexibility in VL-selection enables *ReD* to tolerate various patterns of VL faults. To address network congestion in the presence of VL faults, *ReD* incorporates a dynamic traffic-aware VL selection strategy to enhance runtime routing efficiency. Simulation results comparing *ReD* to state-of-the-art routing algorithms demonstrate significant improvements. *ReD* enhances network reachability by up to 75% when operating with a fault rate of up to 25%. Additionally, it reduces network latency by up to 40% in multi-application execution scenarios, all while incurring less than 2% area overhead. These findings underscore the potential of *ReD* in advancing the performance of emerging 2.5D chiplet systems.

# Chapter 4

# A Reconfigurable Silicon-Photonic 2.5D Chiplet Network with PCMs for Energy-Efficient Interposer Communication

2.5D chiplet systems have been proposed to improve the low manufacturing yield of large-scale chips. However, connecting the chiplets through an electronic interposer imposes a high traffic load on the interposer network. Silicon photonics technology has shown great promise towards handling a high volume of traffic with low latency in intra-chip network-on-chip (NoC) fabrics. Although recent advances in silicon photonic devices have extended photonic NoCs to enable high bandwidth communication in 2.5D chiplet systems, such interposer-based photonic networks still suffer from high power consumption. In this work, we design and analyze a novel Reconfigurable power-efficient and congestion-aware Silicon-Photonic 2.5D Interposer network, called ReSiPI. Considering runtime traffic, ReSiPI is able to dynamically deploy inter-chiplet photonic gateways to improve the overall network congestion. ReSiPI also employs switching elements based on phase change materials (PCMs) to dynamically reconfigure and power-gate the photonic interposer network, thereby improving the network power efficiency. Compared to the best prior state-of-the-art 2.5D photonic network, ReSiPI demonstrates, on average, 37% lower latency, 25% power reduction, and 53% energy minimization in the network.

## 4.1 Introduction

While a 2.5D chiplet system provides higher modularity and yield than a monolithic 2D chip with the same functionality, the interposer network becomes a potential bottleneck as it is supposed to provide low-latency and high bisection-bandwidth communication among the chiplets, which significantly impacts the system's performance and scalability. Although conventional electronic

NoCs can efficiently support a small chip with low to medium traffic load, such as at the intra-chiplet level, they impose a high latency when they are employed on an interposer to handle the global traffic among chiplets [78–80]. The high latency of an electronic interposer is due to its long metal interconnects and low inherent bandwidth to support the high volume inter-chiplet traffic.

To improve intra-chip communication performance in manycore systems, photonic NoCs (PNoCs) [81–84], which use silicon photonic devices and waveguides to modulate, switch, and transmit data among many processing cores and memory, can be used. Advances in silicon photonics technology [85] have allowed data transmission in PNoCs to benefit from the high throughput, reduced dynamic power, and lower transmission delays of light-speed communication [86]. The inherent high bandwidth and low latency of PNoCs also makes them a promising solution for inter-chiplet communication in 2.5D platforms [78, 79]. Accordingly, 2.5D chiplet systems with photonic interposer networks have recently received some attention [78–80, 87]. Such photonic interposer networks can employ wavelength-division multiplexing (WDM) to simultaneously support multiple data streams, each modulated on a different optical wavelength traversing a waveguide, to boost communication bandwidth. However, a high bandwidth photonic interposer network also requires a large number of wavelengths per waveguide, which imposes a high laser power consumption overhead [79]. Fortunately, as we show in this work, a reconfigurable photonic interposer network can handle such power-performance trade-off, where the network bandwidth can be increased for high traffic load scenarios, and similarly, the bandwidth can be reduced to save power under low traffic load conditions.

More recently, the integration of silicon photonics and phase-change materials (PCMs) has created a unique opportunity to realize adaptable, reconfigurable, and programmable photonic networks. PCM-based switches [88, 89] and couplers [90] have been proposed to realize energy-efficient optical signal switching in photonic networks. In particular, PCM-based silicon photonic devices are non-volatile devices in which the switching state is preserved even in the absence of an electrical voltage/current, hence improving the power efficiency in networks employing such

devices. Although PCM-based devices are relatively slow (e.g., 10 Mhz [88]) to be used for fast switching, they are still very efficient devices to support sporadic network reconfiguration [91].

Prior work has explored the use of silicon photonics to realize high performance interposer networks [78–80, 87]. However, these efforts either suffer from high power consumption or high network latency because the interposer is not reconfigurable to adapt to different traffic load conditions. To address these drawbacks, we develop a novel PCM-based Reconfigurable Silicon-Photonic Interposer (ReSiPI) network for 2.5D chiplet systems. The main contributions of ReSiPI are summarized below.

- We propose a reconfigurable photonic interposer network with an intelligent dynamic gateway-activation mechanism based on the network's traffic load at runtime.
- ReSiPI increases inter-chiplet communication bandwidth by increasing the number of active gateways, and not wavelengths, to efficiently distribute the bandwidth improvement across chiplets while saving laser power.
- We present a power-saving mechanism to tune input optical power of modulators by employing PCM-based devices and laser-power management in ReSiPI.
- As source routers on chiplets need to select a gateway to send packets to other chiplets, ReSiPI proposes an efficient dynamic gateway-selection approach to distribute traffic load while minimizing source-destination hop-counts.

The rest of the chapter is organized as follows. Background and prior related work in 2.5D chiplet systems are reviewed in Section 4.2. Section 4.3 discusses our proposed photonic interposer network, ReSiPI. In Section 4.3, evaluation results comparing ReSiPI to the state-of-the-art are presented. Finally, Section 4.5 concludes the chapter.

## 4.2 Background and Related Work

### 4.2.1 Chiplet systems and electronic interposers

To improve manufacturing costs, 2.5D integration was employed in [45] to disintegrate a large multicore chip into smaller chiplets. Doing so breaks the original, larger NoC into several smaller NoCs on each chiplet and an inter-chiplet interposer network. However, such a disintegration introduces some performance loss in the system because it is not trivial to create an interposer network that can support high bandwidth and fast communication required among chiplets. Moreover, the disintegration of the original deadlock-free NoC can introduce new system-wide deadlock conditions where a cyclic dependency of requests for buffer resources among different chiplets and the interposer negatively affects the system performance. To address deadlock, [52] and [54] proposed routing algorithms to avoid deadlock in 2.5D chiplet systems.

In addition to deadlock, the interposer network can suffer from traffic congestion especially when the system scales up [92]. As shown in Fig. 1.9, there are multiple chiplets and each with several integrated cores, all of which communicate through the interposer network. Therefore, the interposer network should be able to handle a high volume of traffic among chiplets. Moreover, the interposer is large and metal interconnects impose a high delay for long-distance communication [93]. To this end, silicon-photonic interposers have been proposed to improve the latency and bandwidth compared to conventional electronic interposer networks [78–80].

### 4.2.2 Silicon photonic interposers

In this dissertation, we explore the intricacies of data transmission within chiplets connected via a silicon-photonic interposer. The interposer houses various components like modulators, filters, and photodiodes (PDs) crucial for electro-optical and opto-electrical data conversions as it is discussed in section 1. Microring resonator (MR) devices are of particular interest for their efficiency in modulating and filtering, as evidenced by recent studies. Gateways, electronic circuits on chiplets, play a pivotal role in managing modulators and PDs on the interposer, while also facilitating data exchange with routers on the same chiplet. Optical signals of different wavelengths

originate from an off-chip laser source and are directed onto the photonic interposer through optical fiber and grating coupler arrangements. Within this system, MRs modulate electronic data onto optical signals at the writer gateway, while at the reader gateway, each MR filters its corresponding optical signal for detection by the PD. The design of each MR device is tailored to resonate with a specific wavelength, enabling multiple wavelengths to carry data concurrently through the same waveguide, a technique known as Wavelength Division Multiplexing (WDM).

Employing silicon photonics, [80] proposed an interposer based on arrayed-waveguide grating routers (AWGRs) to improve the high latency of electronic interposers. However, [80] considered static optical bandwidth under different traffic loads, which either wastes system power under low traffic loads or sacrifices performance under high traffic loads. PROWAVES in [79] proposed a dynamic bandwidth-management technique for optical gateways by adjusting the number of active wavelengths with respect to the runtime traffic load. The number of active wavelengths is updated in a time epoch based on the network delay experienced in the previous epochs. However, using a single high bandwidth gateway to support several routers on a chiplet creates contention among the intra-chiplet routers to access the high-bandwidth gateway. As we will discuss, ReSiPI increases the number of active gateways to improve optical bandwidth while, at the same time, the gateways are distributed over the chiplet to improve router-gateway access and network congestion. Moreover, ReSiPI intelligently power-gates the idle gateways and manages the input laser power based on the runtime traffic load, to improve the interposer energy-efficiency.

### 4.2.3 PCM-based silicon photonic devices

Photonic devices based on phase-change materials (PCMs) have recently received attention due to their non-volatile property which helps save static tuning power consumption in photonic-switched networks [88–90,94]. A PCM has two states with different optical properties: amorphous and crystalline states. A short optical or electrical pulse can switch the states [94] while a state can be preserved without consuming any power. As the amorphous and crystalline states have different optical properties, PCM-based devices are attractive to design non-volatile optical switches and

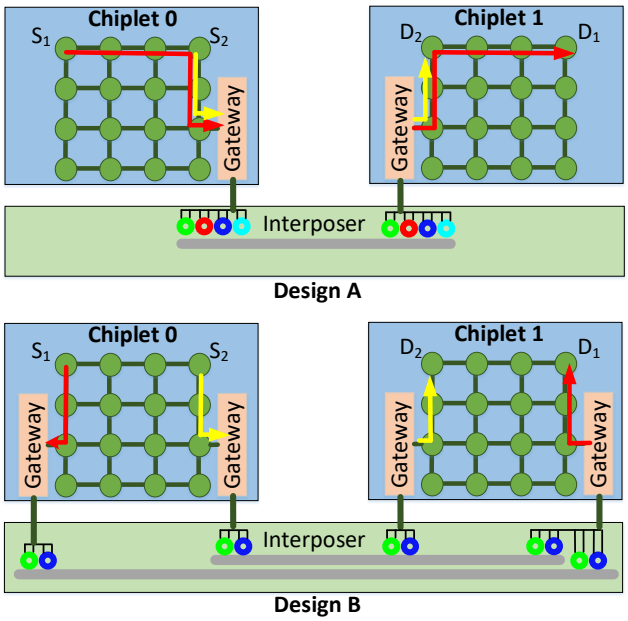



**Figure 4.1:** Dynamic inter-chiplet bandwidth management: Design A with a larger number of wavelengths (e.g., four) and Design B, which is considered in ReSiPI, with a larger number of gateways and fewer (e.g., two) wavelengths.

couplers for photonic networks. For example, a broadband PCM-based switch was proposed in [89] which requires ≈2 nJ energy for reconfiguration. In [91], the same switch was employed to power-gate MR filters in idle reader nodes to reduce a PNoC's tuning power consumption. However, the proposed architecture in [91] does not account for dynamic bandwidth management in the network, to handle the runtime traffic. Moreover, the main power consumption in PNoCs comes from the laser source [86], while [91] only accounts for MR tuning power consumption.

## 4.3 ReSiPI: Overview

In our ReSiPI architecture, an electronic intra-chiplet NoC is considered on each chiplet and a silicon photonic network is considered for the inter-chiplet interposer network. ReSiPI employs the gateway configuration in [78, 79], where gateways to the interposer are placed on the chiplets. The photonic devices are placed on the interposer, and microbump vertical links are used to pass control signals from the gateway (e.g., for driving modulators) to the silicon photonic devices on

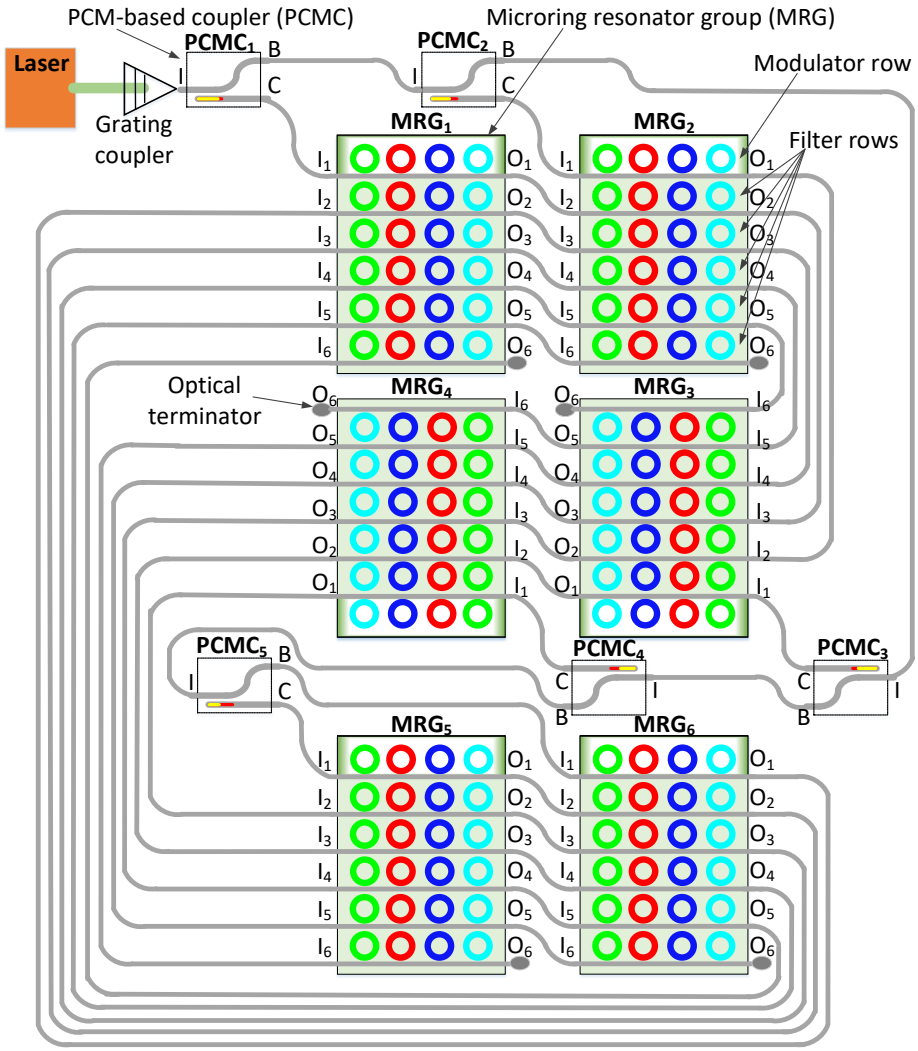



**Figure 4.2:** An example of the proposed photonic interposer architecture (ReSiPI) with a total of six gateways (one per chiplet) and four optical wavelengths. This architecture can be extended to have multiple gateways per chiplet.

the interposer. In this section, after motivating dynamic gateway management, we describe the ReSiPI architecture and its fundamental operational mechanisms.

### 4.3.1 Dynamic gateway management

Unlike state-of-the-art photonic interposer networks [79, 95], where inter-chiplet bandwidth is increased by utilizing a large number of wavelengths, ReSiPI manages the bandwidth by dynamically adjusting the number of active gateways on each chiplet (we consider four gateways per chiplet in our evaluation in Section 4.4). Fig. 4.1 motivates ReSiPI's dynamic gateway-

management approach. As can be seen, there are two ways to increase the inter-chiplet bandwidth: 1) using design A with a larger number of active wavelengths (similar to the approach in [79]), and 2) using design B with a larger number of active gateways (developed in ReSiPI). In this example, there are two packets going from Chiplet 0 to Chiplet 1: $S_1 \rightarrow D_1$ and $S_2 \rightarrow D_2$. Let us assume that each of the two packets requires bandwidth proportional to two optical wavelengths. In design A, four wavelengths are activated on the same gateway, while in design B, two gateways with two wavelengths are considered. In design B, as there are two gateways in Chiplet 0, packets can select between two gateways, resulting in a better traffic-load distribution. In other words, not only can the traffic load be better distributed between gateways but also the chiplet's intra-network traffic load can be better distributed across the chiplet's routers. Therefore, design B has the potential to offer higher performance as the intra- and inter-chiplet bandwidth is more efficiently distributed. Moreover, unlike design A, design B can allow for an intelligent gateway selection mechanism to further reduce the source-to-destination hop count. In this example, packet $S_1 \rightarrow D_1$ requires ten hops of intra-chiplet routing in design A, while it can be routed with only four hops of intra-chiplet routing in design B. As a result, improving the bandwidth with more number of gateways (as in design B) can result in a better performance-cost trade-off than in the case where number of wavelengths is increased (as in design A).

### 4.3.2 ReSiPI interposer network architecture

An example of the ReSiPI interposer network with six gateways and four wavelengths is shown in Fig. 4.2. There is a microring resonator group (MRG) associated with each gateway. Each MRG has four columns of MRs with different colors to show the four different optical wavelengths in this example. ReSiPI employs the Single-Writer Multiple-Reader (SWMR) protocol [86] in waveguides. The first row in each MRG consists of modulator MRs, to actively write electronic data on the associated wavelength on the waveguide (see Fig. 1.10). The last five rows are filter MRs that are wavelength-selective devices to passively read data from their associated wavelengths

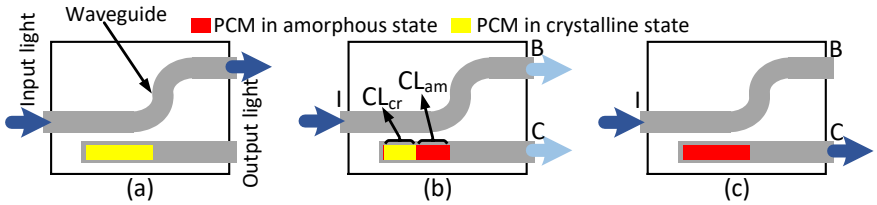



**Figure 4.3:** A PCM-based coupler (PCMC) in different states: (a) crystalline state to guide light to Bar (B) output, (b) partially crystalline state to guide a portion of light to the Cross (C) output and the rest to the Bar output, and (c) amorphous state to guide the input light to the Cross output.

on each waveguide. There are five rows of MR filters as each gateway receives data from the other five gateways.

To efficiently mange the power consumption in the network, ReSiPI not only power-gates the idle electronic gateways on the chiplets, but also the power on the optical signals entering the MRGs of idle gateways is appropriately readjusted. To do so, the laser is tuned to generate less optical power at its output and, therefore, consumes less input power. To power-gate the input of MRGs, a non-volatile, PCM-based reconfigurable directional coupler (PCMC) [90] is employed, as shown in Fig. 4.3. It utilizes PCM to divide the input optical signal between the Cross (C) and Bar (B) outputs. In Fig. 4.3.a, the PCM is completely in the crystalline state and all the input light goes to the B output. In Fig. 4.3.b, where the PCM is partially in the amorphous state, a portion of the input light goes to the C output and the rest traverses to the B output. In Fig. 4.3.c, all the input light propagates to the C output as the PCM is completely in the amorphous state. One practical way to adjust the PCM state is by using an embedded microheater on top of the PCM material on the waveguide [90], because the PCM state changes with a temperature change. For example, using a transparent conductive heater, the PCMC can work at the frequency of 10 Mhz [88].

The coupling ratio ($CR$) in the PCMC can be defined as:

$$\kappa = \frac{CL_{am}}{CL_{cr}}, \tag{4.1}$$

where $CL_{am}$ and $CL_{cr}$ are the coupling lengths of the amorphous and crystalline states, respectively (see Fig. 4.3.b). By adjusting this coupling ratio (e.g., using a microheater), we can tune the

portion of the input light transmitted to the Bar and Cross outputs in a PCMC. Accordingly, and assuming a lossless optical transmission, the optical power at the Cross ($P_C$) and Bar ($P_B$) output is:

$$P_C = \kappa \times P_I, \tag{4.2}$$

$$P_B = (1 - \kappa) \times P_I, \tag{4.3}$$

where $P_I$ is the input optical power in the PCMC. In our ReSiPI interposer network architecture, the coupling ratio ($\kappa$) is tuned to manage the input laser power on each waveguide. Considering Fig. 4.2, a PCMC controls the input optical power of each writer. The coupling ratio of PCMCs are tuned based on the total number of active gateways. If the associated writer gateway of PCMC$_i$ is deactivated, the coupling ratio of the PCMC should be zero (i.e., $\kappa_i = 0$, PCM is completely in the crystalline state). Otherwise, the coupling ratio of PCMC$_i$ is:

$$\kappa_i = \frac{1}{(\sum_{c=1}^{C} g_c) - i}, \tag{4.4}$$

where $C$ is the total number of chiplets in the system and $g_c$ is the number of active gateways of chiplet $c$.

The organization of MRGs and PCMCs, shown in Fig. 4.2, can be scaled with any number of gateways, chiplets, and PCMCs without loss of generality. Assuming $N$ gateways in the system, the number of MRGs is $N$ while the number of PCMCs is $N - 1$. Moreover, the number of the MRs in each MRG is equal to the number of wavelengths and the number of the waveguides in each MRG is $N$. Even rows of MRGs (e.g., the second row with MRG$_3$ and MRG$_4$ in Fig. 4.2) and their PCMCs (e.g., PCMC$_3$ and PCMC$_4$ in Fig. 4.2) are rotated at 180 degrees compared to the odd rows. For any MRG$_k$, if $k < N$, $O_j$ of MRG$_k$ is connected to $I_{j+1}$ of MRG$_{k+1}$ (see MRG connections in Fig. 4.2). When $k = N$, $O_j$ of MRG$_N$ is connected to $I_{j+1}$ of MRG$_1$. Additionally, $I_1$ of MRG$_k$, if $k < N$, is connected to output C of PCMC$_k$. Also, output B of PCMC$_j$, if $j < N - 1$, is connected to input I of PCMC$_{j+1}$ (see PCMC connections in Fig. 4.2).

### 4.3.3 Adaptive active gateway selection

ReSiPI aims to assign traffic load to gateways in a manner that minimizes congestion. Nevertheless, if the assigned load is too low—i.e, gateways are underutilized—the system power is wasted. The unnecessary power wastage is due to a larger than needed number of gateways that are activated, and their associated tuning and laser power overhead. Therefore, ReSiPI optimizes the number of active gateways per chiplet with the goal of a trade-off between the overall system power and the average packet latency, in a way that gateways are neither congested nor underutilized. To accomplish this, we define $L_m$ as the maximum allowable load on a gateway. This means that beyond $L_m$ load on a gateway, we can expect congestion and performance loss. We use the maximum packet transmission rate on a gateway to measure the gateway load. Then, we update the number of active gateways in each chiplet based on the average load on each chiplet's gateways, with respect to $L_m$. We discuss how to select an optimal value for $L_m$ in Section 4.4.2.

The average gateway load for chiplet $c$ in a reconfiguration interval $i$ ($L_c^i$) is defined as:

$$L_c^i = \frac{1}{g_c} \sum_{j=1}^{g_c} \frac{P_i}{T_i}, \tag{4.5}$$

where $g_c$ is the number of active gateways in the chiplet $c$, $P_i$ is the total number of transmitted packets during reconfiguration interval $i$, and $T_i$ is the duration of reconfiguration interval $i$ in cycles. Note that we assume a fixed packet size, otherwise $P_i$ should be the number of transmitted flits. Moreover, we define a threshold for increasing and a threshold for decreasing the number of active gateways per chiplet. Accordingly, $T_{P_g}$ ($T_{N_g}$) is the threshold for increasing (decreasing) the number of gateways when the current number of active gateways is $g$. For $T_{P_g}$, we have:

$$T_{P_g} = T_{P_1} = T_{P_2} = T_{P_2} = .... = T_{P_G} = L_m, \tag{4.6}$$

where $G$ is the maximum number of active gateways per chiplet. When a gateway's load is higher than $L_m$, the gateway will suffer from notable congestion. Therefore, in such a case, the total number of active gateways on the chiplet should be increased to reduce the load on the congested

gateway(s). As a result, $T_{P_g}$ is equal to $L_m$ in (4.6). On the other hand, $T_{N_g}$ can be defined as:

$$T_{N_g} = L_m(1 - \frac{1}{g}). \tag{4.7}$$

To understand the rationale behind (4.7), let us assume that we gradually reduce load $L$ from $L_m$ and try to find $T_{N_g}$. We need to reduce the number of active gateways from $g$ to $g-1$ when $L$ is small enough to avoid extra load on $g-1$ gateways (in the next reconfiguration interval). We can define this load reduction as:

$$L_d = L_m - L_c, \tag{4.8}$$

where $L_c$ is the current average gateway load of the chiplet. The sum of the reduced load of the $g$ active gateways is then:

$$Sum(L_d) = L_d \times g. \tag{4.9}$$

When $Sum(L_d)$ is equal to the maximum load of one gateway ($Sum(L_d) = L_m$), we can deactivate one gateway. Thus, we have:

$$T_{N_g} = L_c, \quad \text{if} \quad Sum(L_d) = L_m. \tag{4.10}$$

From (4.8), (4.9), and (4.10), we can calculate the threshold for decreasing the number of gateways: $T_{N_g} = L_m - L_d = L_m - \frac{L_m}{g} = L_m(1 - \frac{1}{g})$.

As an example, the procedure to increase and decrease the number of active gateways ($g$) for a network with four gateways per chiplet is illustrated in Fig. 4.4. Based on (4.6), in each $g$ in the figure, if $L_c^i$ exceeds $L_m$, a new gateway will be activated ($g \rightarrow g+1$). On the other hand, according to (4.7), if $L_c^i$ goes below $L_m(1 - \frac{1}{g})$, one gateway will be deactivated ($g \rightarrow g-1$). Based on (4.7), $T_{N_g}$ for different $g$ values is shown in a table in Fig. 4.4.

The dynamic gateway management algorithm in ReSiPI is illustrated in Fig. 4.5. The first step is to update the number of active gateways for each chiplet ($g_c$), which is initially set to the maximum allowed (four in our experiments in Section 4.4). After finding $g_c$, ReSiPI decides

| g | $T_{Pg}$ | $T_{Ng}$ |
|---|---|---|
| 1 | $L_m$ | 0 |
| 2 | $L_m$ | $0.5L_m$ |
| 3 | $L_m$ | $0.66L_m$ |
| 4 | - | $0.75L_m$ |

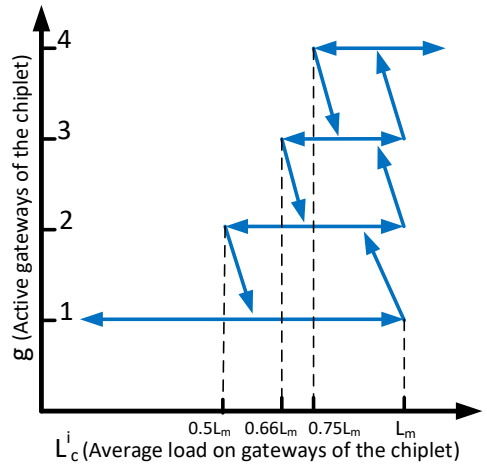



**Figure 4.4:** Number of active gateways based on load changes.

whether to change the number of active gateways, based on the procedure outlined above. If ReSiPI decides to increase the total number of active gateways ($GT$), first the laser power will be increased appropriately and then the additional gateways will be activated. On the other hand, to reduce the total number of active gateways, after waiting for packets of the candidate gateways to be routed (flushed), the gateways will be deactivated. After the gateway deactivation, the laser power can be reduced using a tunable SOA-based laser [96].

We define a reconfiguration interval (i.e., epoch) at which we trigger the procedure to update the number of active gateways. A short reconfiguration interval will result in more frequent and responsive adaptation to traffic dynamics, while a long reconfiguration interval will result in a low reconfiguration overhead cost but also low responsiveness. We consider a reconfiguration interval length such that the update cost is negligible and ReSiPI is also able to efficiently adapt to traffic dynamics. The reconfiguration interval length that we consider (one million cycles) is significantly larger than the time to perform the reconfiguration/update processes, as will be further discussed in Section 4.4.

### 4.3.4 Per-packet gateway selection

For inter-chiplet packets, where routing over the photonic interposer network is required, a gateway in the source chiplet and a gateway in the destination chiplet are selected to perform the

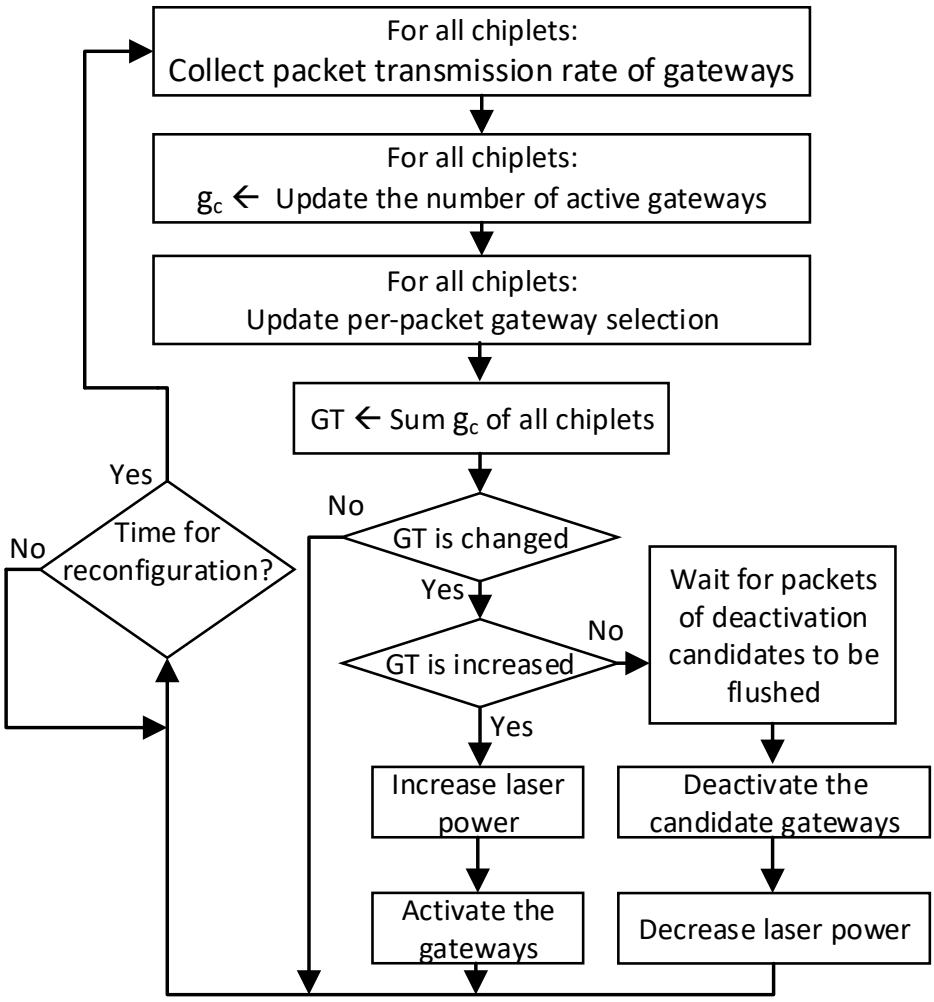



**Figure 4.5:** Dynamic gateway management in ReSiPI.

packet routing. Therefore, the routing process of an inter-chiplet packet is performed in three steps: 1) routing from the source router to the selected gateway on the source chiplet, 2) routing from the selected gateway on the source chiplet to the selected gateway on the destination chiplet, and 3) routing from the gateway on the destination chiplet to the destination router.

The gateway selection for each packet flow impacts the network performance, because it defines the assigned load on the gateways. An imbalanced gateway selection by packets can impose congestion on the gateways and degrade the overall performance [40, 92]. ReSiPI uses a dynamic per-packet gateway selection approach based on the number of active gateways in the source and destination chiplets. We take into account 1) gateways' traffic load and 2) router to gateway hop-

counts in the analysis for gateway selection. To distribute the traffic load on the gateways, we try to balance the load on the gateways. Therefore, the average number of routers which can utilize the same gateway is $R_g = \frac{R}{g_c}$, where $R$ is the total number of routers on the chiplet. Then, we assign $R_g$ routers to a gateway in its vicinity. An example of gateway selection is shown in Fig. 4.6. In Fig. 4.6.a, only one gateway is activated, so all routers utilize this gateway. In Fig. 4.6.b, as there are two activated gateways, half of the routers ($R_g = 8$) utilize the same gateway. For Figs. 4.6.c–d, similarly, the selection is done to balance the load on the gateways while each router is assigned to a gateway in its vicinity. For selecting the gateway at the destination chiplet, different gateway-selection scenarios are pre-analysed during design-time and the data related to the optimal destination gateway to minimize latency (for different scenarios of activated gateways at the destination chiplet) is stored in the gateway routers.

Thus, for any packet being transmitted, the first routing step is performed in the source router based on the number of local active gateways, while the second step is performed in the source gateway based on the number of active gateways in the destination chiplet. In this way, the global information about active gateways only needs to be stored at gateways. Therefore, the source router is only aware of the number of active gateways in the source chiplet. On the other hand, the source gateway is aware of the number of active gateways in the destination chiplet. Design-time analysis helps to achieve a low-cost destination gateway selection that minimizes the latency to the destination router in the third step. This analysis utilizes hop count (from the destination gateway to the destination router) and number of active gateways in the destination chiplet to store selection decisions at the source gateway router, and these decisions are updated at every reconfiguration interval.

### 4.3.5 Reconfiguration controller architecture

ReSiPI utilizes a controller in each chiplet to manage the gateway activation/deactivation, PCMCs, and laser power at the start of each reconfiguration interval. One of these controllers acts as the global manager that interacts with a local gateway controller (LGC) in each chiplet.

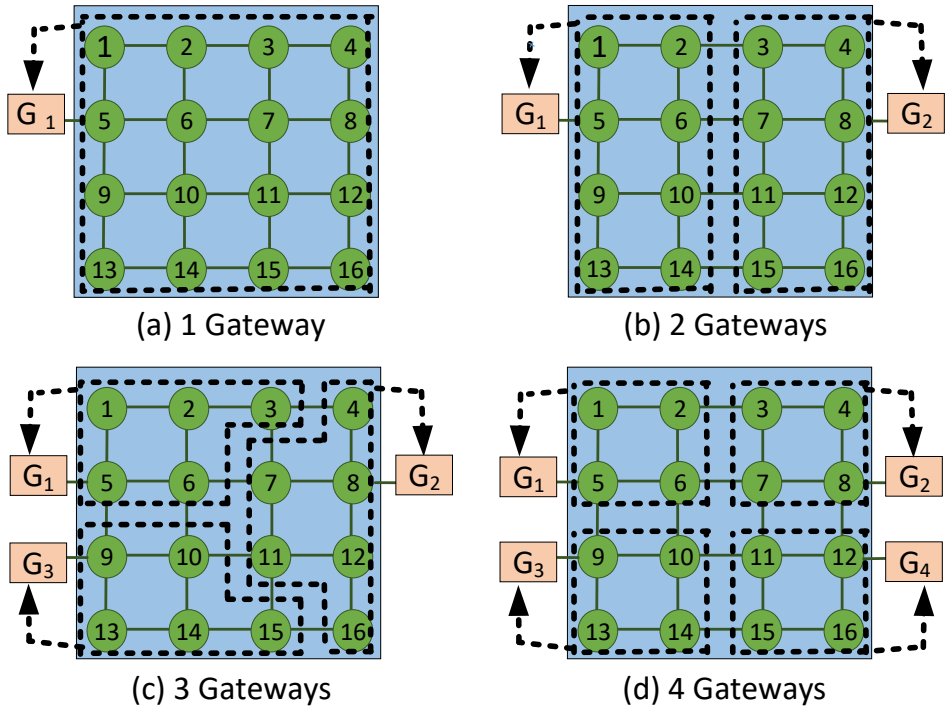



**Figure 4.6:** An example of the adaptive gateway selection in ReSiPI for different number of activated gateways. The dashed boxes show the routers that will use a specific gateway (G).

The structure of ReSiPI's controller is shown in Fig. 4.7. All controllers in the system have this architecture but note that the interposer controller (InC) is only present in the global manager controller. LGCs decide on the number of active gateways on a chiplet, based on the number of routed packets over the chiplets active gateways. The LGC of each chiplet sends its number of active gateways to the interposer controller (InC) of the global manager controller (in one of the chiplets) at the end of a reconfiguration interval. InC sums the number of active gateways of chiplets ($g_c$) to define the total number of active gateways ($GT$) and tunes the PCMCs (based on (4.4)) and the laser power, as discussed in the earlier subsections. The controller overhead is discussed in Section 4.4.

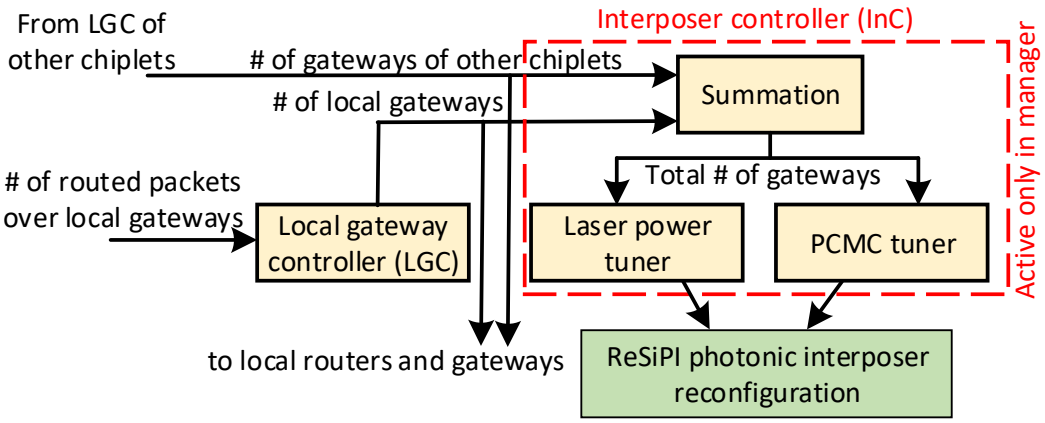



**Figure 4.7:** ReSiPI's reconfiguration controller architecture.

**Table 4.1:** Simulation setup of ReSiPI.

| Parameter | value |
|---|---|
| Number of chiplets | 4 (each a 4×4 mesh NoC) |
| Maximum gateways per chiplet | 1 for PROWAVES [79] |
| | 4 for AWGR [80] and ReSiPI |
| Gateways for memory controllers | 2 |
| Gateway buffer size | 32 flits for PROWAVES |
| | 8 flits for AWGR and ReSiPI |
| Intra-chiplet router buffer size | 4 flits |
| Routing in chiplets | DeFT (deadlock free) [92] |
| Intra-chiplet NoC frequency | 1 Ghz |
| Data rate of optical link | 12 Gb/s per wavelength |
| Simulation cycles | 100 M (10 K for warm-up) |
| Reconfiguration interval duration | 1 M cycles |
| Packet size | 8 flits (each flit 32 bits) |

## 4.4 Simulation Results and Analysis

### 4.4.1 Simulation setup

To model 2.5D chiplet network platforms, we enhanced Noxim [75], which is a cycle accurate NoC simulator. We used GEM5 [38] in full system mode to generate traffic traces of PARSEC benchmarks [42]. We considered 64 x86 cores, where each core has a private L1 cache, four coherence directories, and four shared L2 cache banks. We integrated the generated traffic traces into our enhanced Noxim simulator to analyze latency, power, and energy of the system. We compare ReSiPI with two photonic interposer networks, AWGR [80] and PROWAVES [79]. Our simula-

tion configuration and setup is summarized in Table 4.1. We simulated a 2.5D network with four chiplets, where each chiplet has 16 cores connected by a 4×4 mesh-based electronic NoC. We considered four gateways per chiplet where the gateways are connected to the chiplet similar to Fig. 4.6.d. The number of gateways per chiplet and the location of gateways are based on [52]. Unlike ReSiPI, PROWAVES advocates for changing the number of wavelengths to adapt a gateway's bandwidth to meet inter-chiplet bandwidth demands. We considered 16 wavelengths for PROWAVES, while ReSiPI uses 4 wavelengths. As a result, (number of wavelengths) × (number of gateways) in PROWAVES and ReSiPI are equal, to ensure that both have the same inter-chiplet bandwidth for a fair comparison. Moreover, we also considered the same buffer resource usage in both architectures. As ReSiPI has 4× gateways compared to PROWAVES, we considered 4× buffer size for PROWAVES (8 flit buffers in ReSiPI and 32 flit buffers in PROWAVES). AWGR [80] requires one wavelength per gateway, so 18 wavelengths are used in the AWGR approach as the 2.5D network has 18 gateways in total. We used the silicon photonic power model in PROWAVES [79]. In the power model, laser power is 30 mW (per wavelength per waveguide), TIA power is 2 mW, thermal tuning power (per MR) is 3 mW, and driver power is 3 mW [97].

### 4.4.2 Design-space exploration: Optimal $L_m$

As discussed in Section 4.3.3, $L_m$ is the maximum allowable load on a gateway. To find the optimal $L_m$, we evaluated our 2.5D network with various traffic and configuration scenarios. The results are shown in Fig. 4.8. We simulated eight PARSEC applications: blackscholes, swaptions, streamcluster, facesim, fluidanimate, bodytrack, canneal, and dedup. The four different colors in Fig. 4.8 indicate the four main network configurations that we explored, with different numbers of gateways (1 to 4) per chiplet. Each simulation configuration, which corresponds to one point in the figure, gives us the average gateways' load $L_c$ (see (4.5)) and the average packet latency. For the points with higher $L_c$, average latency is increased. Therefore, if we want to reduce the average latency, choosing a solution with lower $L_c$ is more efficient. However, a low $L_c$ means utilizing a larger number of active gateways, which will result in higher power consumption. This is because

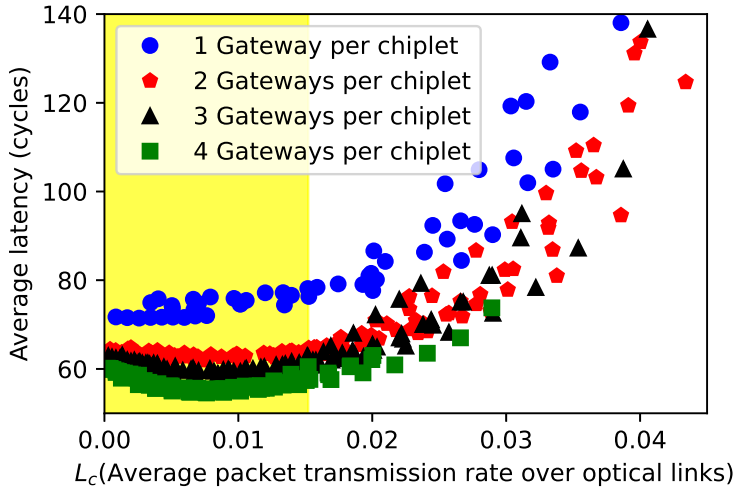



**Figure 4.8:** Average latency vs. optical link transmission rate.

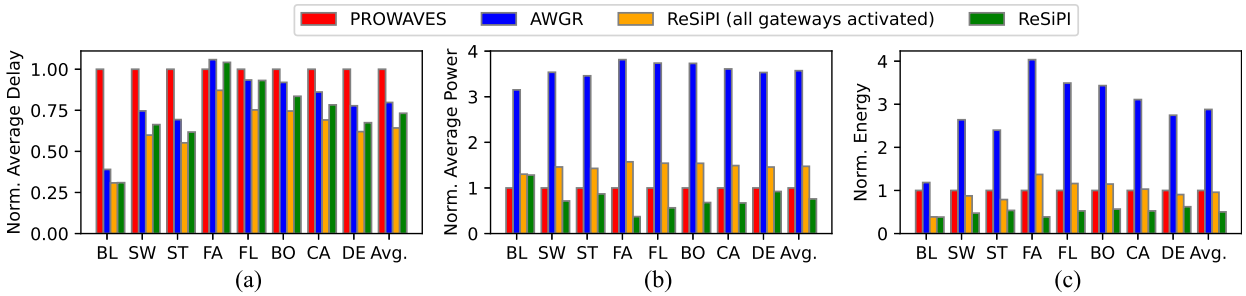



**Figure 4.9:** (a) Normalized average latency, (b) Normalized average power, and (c) Normalized energy.

under the same traffic load, if the number of gateways is larger, less traffic load is assigned to each gateway. As a result, there is a trade-off in selecting $L_c$ with lower average latency or lower power consumption. In selecting $L_c$, we accept up to 10% overhead in latency (empirically determined). The yellow-shaded region in Fig. 4.8 includes the points for which the average latency is smaller than 10% overhead compared to the lowest average latency. Note that each point is compared with the points with the same number of active gateways. By accepting 10% average latency overhead, $L_m$ is 0.0152 (maximum $L_c$ in the yellow-shaded region). With this value of $L_m$, the threshold for increasing the number of gateways ($T_{P_g}$) and the one for decreasing the number of gateways ($T_{N_g}$) can be calculated using (4.6) and (4.7).

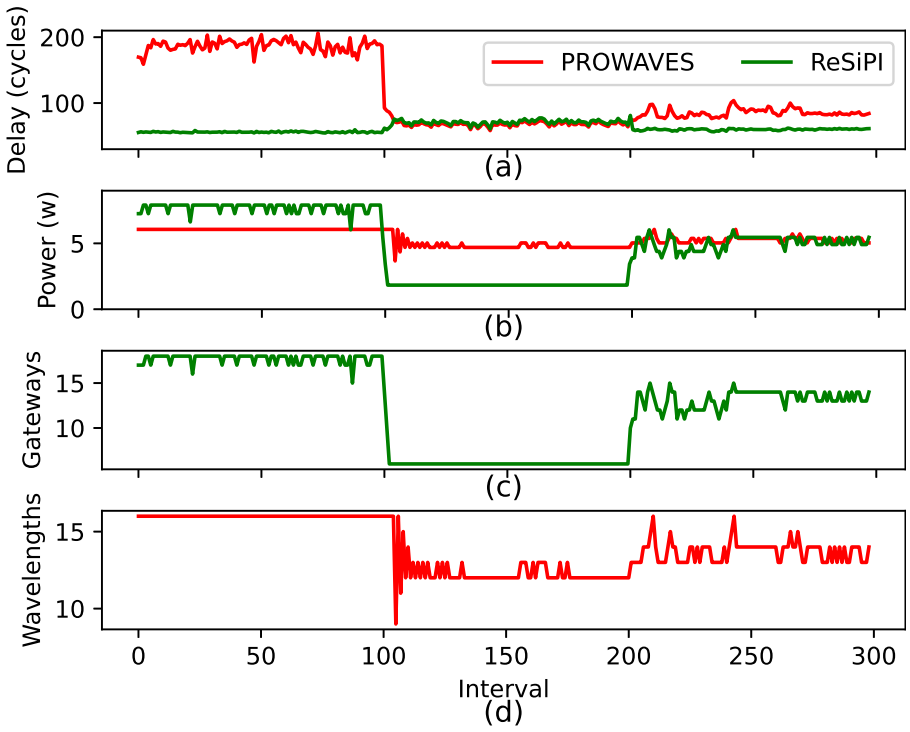



**Figure 4.10:** Adaptivity comparison between ReSiPI and PROWAVES: (a) average delay, (b) average power, (c) number of activated gateways in ReSiPI, and (d) number of activated wavelengths in PROWAVES.

### 4.4.3 ReSiPI controller overhead

We implemented ReSiPI's controller in HDL and synthesised it using Cadence Genus. We considered 1 Ghz clock frequency and 45 nm technology. The area and power overhead of the controller is summarized in Table. 7.1. Both area and power are negligible compared to the budget of a chiplet (e.g., in [79], the chiplet area is 53.83 $mm^2$). In addition to the controller circuit, there are two important actions in the update process: 1) the reconfiguration time of PCMCs, and 2) the delay for tuning the laser power. We assumed the heater used in [98], to change PCMCs' state. According to [98], a PCM's state can be reconfigured in 100 ns. As our NoC frequency is 1 Ghz, the reconfiguration time of PCMCs is 100 cycles. We also assume an SOA-based laser and the time to tune the laser power is 20–50 ps [96]. We consider a reconfiguration interval of one million cycles which is sufficient to capture major trends in traffic load changes, while it is quite large in comparison with the time to do the reconfigurations. The latency and power overhead for ReSiPI is considered in our simulation analysis in the rest of this section.

**Table 4.2:** Overhead analysis of ReSiPI's controller (see Fig. 4.7).

| Parameter | LGC | InC | Total |
|---|---|---|---|
| Area ($\mu$m$^2$) | 314 | 104 | 418 |
| Power ($\mu$W) | 172 | 787 | 959 |

### 4.4.4 Latency, power, and energy analysis

The average latency, power, and energy results for all compared 2.5D network architectures are shown in Fig. 4.9. The first two letters of each application is shown on the x-axis in the figure. In addition to ReSiPI, AWGR [80], and PROWAVES [79], we compared a variant of ReSiPI where all gateways are activated, to analyze the impact of dynamic inter-chiplet bandwidth management.

As shown in Fig. 4.9.a, ReSiPI significantly improves the average latency in all the eight applications. On average, ReSiPI offers 37% lower average latency due to its efficient architecture and bandwidth management. Moreover, as shown in Fig. 4.9.b, ReSiPI consumes 25% less power in comparison with PROWAVES, which is due to two main reasons. First, ReSiPI can handle inter-chiplet traffic with lower bandwidth budget as bisection bandwidth between chipets and the interposer is more distributed across the chiplets. Second, utilizing the PCM-based couplers, ReSiPI intelligently power-gates some part of the photonic interposer and saves laser power consumption. The AWGR approach [80] has high power consumption because 1) one wavelength is required for each AWGR port (gateway) and 2) AWGR's optical loss is high (1.8 dB loss based on [80]). Energy analysis is also shown in Fig. 4.9.c, where ReSiPI offers a remarkable reduction across all the applications.

Fig. 4.9.a shows that ReSiPI imposes a small average latency overhead compared to the ReSiPI variant with all gateways activated. This is because ReSiPI intelligently accepts a small latency overhead to considerably save on the power consumption, as shown in Fig. 4.9.b. As we discussed in Section 4.4.2, we chose $L_m$ while accepting 10% overhead in the average latency to save on the power consumption in the design trade-off. Selecting a smaller $L_m$ slightly improves the average latency while imposing high power consumption overhead. Therefore, compared to when all the gateways are activated, ReSiPI greatly minimizes energy, as shown in Fig. 4.9.c.

### 4.4.5 Adaptivity analysis

To contrast the adaptive behavior in ReSiPI and the best performing 2.5D network from prior work, PROWAVES (e.g., when traffic load changes), we simulated three applications in a sequence. Each application was executed for 100 million cycles (100 intervals). We measured latency and power of each reconfiguration interval to observe the adaptation behavior with ReSiPI and PROWAVES across reconfiguration intervals. For this analysis, we selected the applications with the highest load: Blackscholes, the lowest load: Facesim, and the median load: Dedup, respectively. Fig. 4.10 shows the performance of ReSiPI and PROWAVES in terms of the average delay and the average power during the reconfiguration intervals. For the first 100 reconfiguration intervals, when Blackscholes is executing, which is the application with the highest load, ReSiPI can handle the traffic load and offers a low average latency. As shown in Fig 4.10.c, ReSiPI activates the maximum number of gateways ($4\times4+2=18$) in most of the cases to handle the traffic load with a small power overhead. During the Blackscholes application, although PROWAVES runs at its maximum bandwidth capacity with the maximum number of wavelengths (see Fig. 4.10.d), it is unable to adequately handle the traffic because the bandwidth is increased on the single gateway on each chiplet, rather than in a distributed manner across gateways in ReSiPI. Switching from Blackscholes to Facesim, ReSiPI adapts to the new traffic within three reconfiguration intervals only, whereas PROWAVES is unstable for five reconfiguration intervals. During the execution of Facesim, ReSiPI switches to a smaller number of active gateways and significantly reduces power consumption. ReSiPI imposes a small average latency overhead when executing Facesim. This is because ReSiPI finds the traffic load low and deactivates some unnecessary gateways. For the third application (Dedup), ReSiPI is again able to efficiently adapt to the traffic and manage the number of active gateways to achieve low power consumption. We also use the Dedup traffic to show the bandwidth distribution of ReSiPI next.

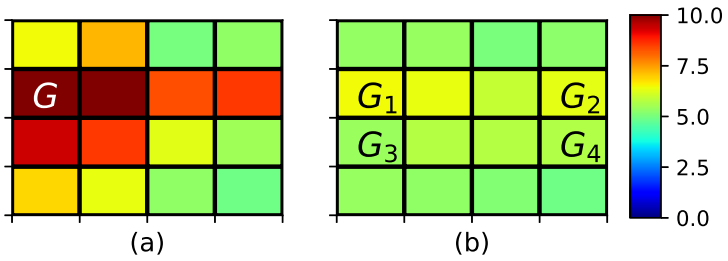



**Figure 4.11:** Average residency of flits on the routers in the first chiplet in (a) PROWAVES and (b) ReSiPI.

### 4.4.6 Bandwidth distribution analysis

To further explain the performance differences between PROWAVES and ReSiPI, we monitored the residency of flits, which is the average time (in cycles) that flits stay in the router for both architectures. Fig. 4.11 shows the average residency of one of the chiplets when using PROWAVES and ReSiPI. We do not show all the chiplets as the trend is similar in other chiplets. Although PROWAVES increases the bandwidth of gateways by increasing wavelengths, there is high congestion on the router connected to the gateway as shown in Fig. 4.11.a (router $G$ in the figure). Moreover, the high congestion on the routers leads to back-pressure in the entire chiplet, creating high network congestion. On the other hand, as the load is more efficiently distributed among different routers in ReSiPI, the average residency of routers is low (see Fig. 4.11.b). In ReSiPI, two gateways are often activated, which are connected to the routers at $G_1$ and $G_2$ in Fig. 4.11.b. The distributed bandwidth enhancement in ReSiPI thus significantly improves network congestion over PROWAVES.

## 4.5 Conclusion

This chapter presented ReSiPI which is a PCM-based reconfigurable silicon-photonic interposer network architecture for improving energy-efficiency in 2.5D chiplet systems. ReSiPI monitors the traffic load on the interposer and dynamically activates/deactivates gateways in the network. Activation of a larger number of gateways improves the average latency, while increasing the power consumption, and vice versa. ReSiPI's controller intelligently manages this latency-power trade-off and, therefore, can achieve 53% improvement in network energy, in comparison

with the best state-of-the-art 2.5D photonic network. Results with real application traffic indicate that ReSiPI is a promising solution for an energy-efficient interposer network in emerging 2.5D chiplet platforms.

# Chapter 5

# Machine Learning Accelerators in 2.5D Chiplet Platforms with ReSiPI-based Silicon Photonic Interposer

Machine learning (ML) accelerators outperform CPUs and GPUs in energy-efficient ML processing. However, the progress of electronic accelerators is constrained by fundamental limits arising from limited bandwidth and high-latency metallic interconnects. Therefore, in this dissertation, we investigate the utilization of optical interconnects to enhance communication in ML accelerators. This chapter presents the implementation of a 2.5D Chiplet system utilizing an optical interposer to facilitate a new class of scalable ML hardware accelerators. The interposer architecture is based on ReSIPI interposer technology, aimed at dynamically enhancing inter-chiplet bandwidth and improving energy efficiency in communication within ML accelerators. Additionally, the next chapter delves into exploring a novel switch design for optical interposers.

## 5.1 Introduction

The current use of GPUs to accelerate DNN execution is limited due to high power consumption, increasing area overhead, and memory bandwidth limitations. To tackle these problems and to effectively accelerate modern DNNs in a scalable manner, 2.5D architectures are actively being considered. However, inter-chiplet metallic interconnects pose a major challenge to system performance due to excess latency and energy consumption. To overcome these limitations, in this chapter, silicon photonic interconnects are being explored as an alternative option for energy effecient communication in 2.5D chiplet platforms for DNN acceleration.

The organization of this chapter is structured as follows: In the upcoming section, we delve deeper into our case study, which focuses on a silicon photonic 2.5D DNN accelerator. Following

that, in Section 5.3, we present our simulation results. Finally, Section 5.4 concludes the chapter and discusses the open challenges.

## 5.2 Case study: Silicon photonic 2.5D DNN accelerator

To explore the implications of accelerating DNNs on 2.5D interposer platforms, we present a case study that involves extending the CrossLight [99] photonic DNN accelerator to the 2.5D chiplet platform. CrossLight is a neural network accelerator designed to perform high speed multiply and accumulate (MAC) operations in the photonic domain. However, the original monolithic CrossLight architecture suffers from low scalability and relatively low energy efficiency. We propose to use a ReSiPI-based photonic interposer architecture to design a more scalable and energy-efficient 2.5D CrossLight implementation. A high-level overview of our chiplet-based 2.5D CrossLight accelerator with a photonic interposer is shown in 5.1. Several chiplets are packaged on a silicon photonic interposer substrate. We consider different types of chiplets as part of a heterogeneous architecture. Such a heterogeneous design allows system-on-chip (SoC) designers to utilize appropriate off-the-shelf chiplets and create diverse 2.5D packages to meet their design targets [100].

The chiplets in the proposed architecture consist of various computational and memory chiplets. One or more chiplets consist of an optically-interfaced memory architecture, such as high bandwidth memory (HBM; shown in 5.1), with a dedicated gateway to communicate with the rest of the system. Each compute chiplet (e.g., chiplets 1-4 in 5.1) hosts several photonic MAC units and has its local gateway(s) to read data from the memory chiplets and write data to them through the interposer network. Each gateway has two main parts: electronic circuitry on the chiplet and a Microring Resonator Group (MRG) on the interposer. The electronic part of a gateway is connected to the microrings of an MRG using the microbump technology. The photonic MAC units is based on CroosLight [], which employ MRs to perform multiply operations between parameters, and photodetectors to obtain the sum of products. The proposed architecture employs heterogeneous MAC unit sizes (size referring to the size of the vectors that can be deployed) across different

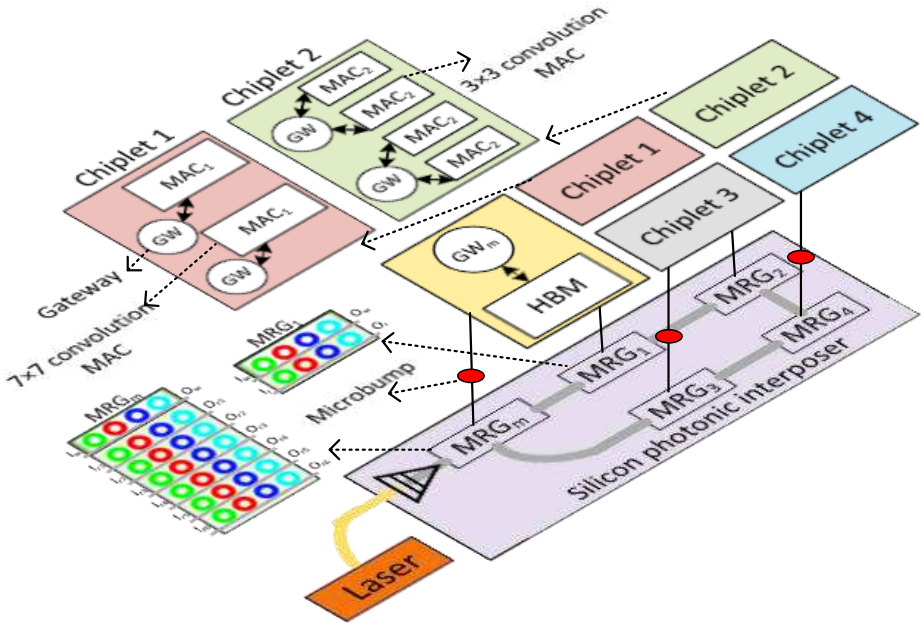



**Figure 5.1:** Overview of proposed 2.5D interposer chiplet-based DNN accelerator architecture.

chiplets to cater to the different kernel sizes and to handle the large-scale MAC operations needed for the fully connected layers. For example, in Fig.3, Chiplet 1 includes 3×3 convolution MACs, while Chiplet 2 contains 7×7 convolution MACs. Moreover, as footprint of MACs with various sizes are different, the number of MACs per chiplet can vary for each chiplet.

An example of the optical interface and communication on the interposer in this architecture is shown in 5.2. In this example, MACs are reading data from the HBM on a separate chiplet. For successful communication, a writer gateway, including buffers to store and forward data, is utilized in the HBM chiplet, and similarly, a reader gateway is utilized on the chiplet with MAC units. The stored data in the buffers of the writer gateway is modulated on the optical signals which are generated by an off-chip laser. Different colors of modulators show that they are used to modulate different optical signals on different wavelengths. As discussed earlier, employing several optical signals with different wavelengths enables our network to transmit more data at the same time on the same waveguide, to improve the communication bandwidth. Several MR filters are also connected to the reader gateways. Each MR filter is tuned at a specific wavelength to filter and drop the specified optical signal. After this step, the optical signal is converted to

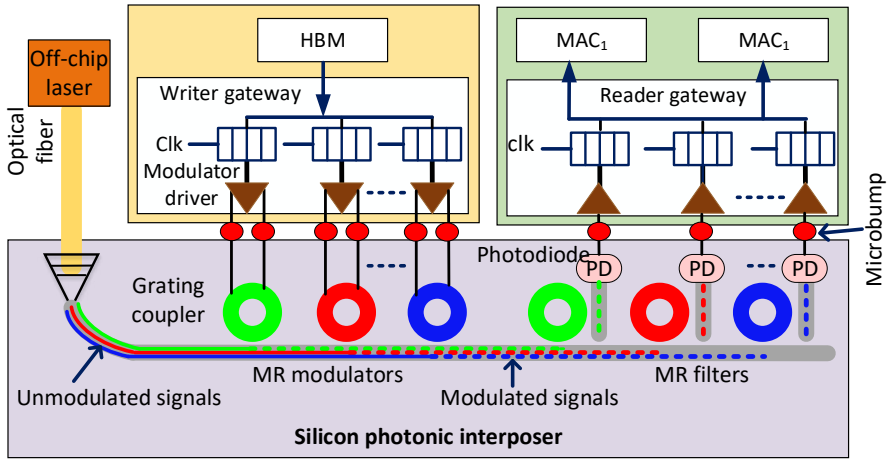



**Figure 5.2:** Example of optical communication on interposer: MACs are reading data from memory.

an electronic signal using a photodiode, and this signal is delivered to the reader chiplet using microbumps. The reader gateway converts the electronic signal to digital data, and stores the received data in its buffer. Finally, the data will be forwarded to the MACs. Such a protocol for optical communication, where a reader is receiving data from a writer using a waveguide, is called the single writer single reader (SWSR) protocol. Similarly, if several readers are receiving data from a writer using a waveguide, the protocol is referred to as single writer multiple reader (SWMR) protocol.

In our architecture, we have two types of traffic between the chiplets: 1) reading weights and inputs needed by MACs from memory, and 2) writing MAC outputs to the memory. As a result, from the memory chiplet to the compute chiplets, we utilize the SWMR protocol to perform reads from memory. Moreover, from the compute chiplets to the memory, we use the SWSR protocol. Therefore, the MRG of the memory chiplets requires several sets of MR filters (each set of MR filters is a row of the MRG shown in 5.1) to receive data from the compute chiplets. On the other hand, a compute chiplet requires only one set of MR filters as it only receives data from the memory. Both compute and memory chiplets require one set of MR modulators to send data.

An example of our 2.5D CrossLight with the integrated ReSiPI interposer is shown in 5.3. Although this example is shown with six gateways (associated with one memory chiplet and five

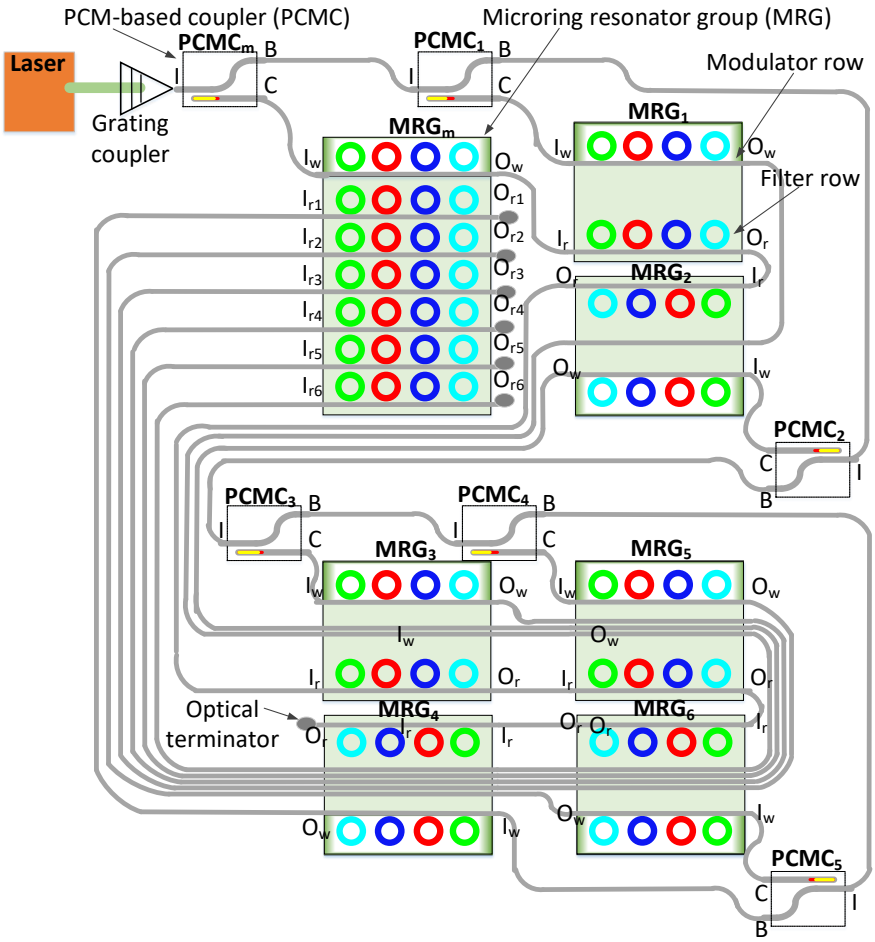



**Figure 5.3:** Silicon photonic network in our 2.5D chiplet-based DNN accelerator. Each MRG is connected to a gateway on a chiplet.

compute chiplets), the interposer design can be extended to a larger system without loss of generality. As shown in 5.3, the MRG of the memory chiplet (MRGm) has six filter rows to receive data from the six gateways of the compute chiplets (MRG1 – MRG6), while MRGm has one row of modulators to send data to all the gateways. The photonic interposer network architecture is a passive network to save energy. This means that there is a specific waveguide to transmit data from each writer gateway to each reader gateway and the route (waveguide) does not change.

## 5.3 Experimental results

We designed two variants of the 2.5D CrossLight architecture: with a ReSiPI-based interposer [25] (2.5D-CrossLight-SiPh-Interposer), and an electrical mesh interposer [23] (2.5D-CrossLight-

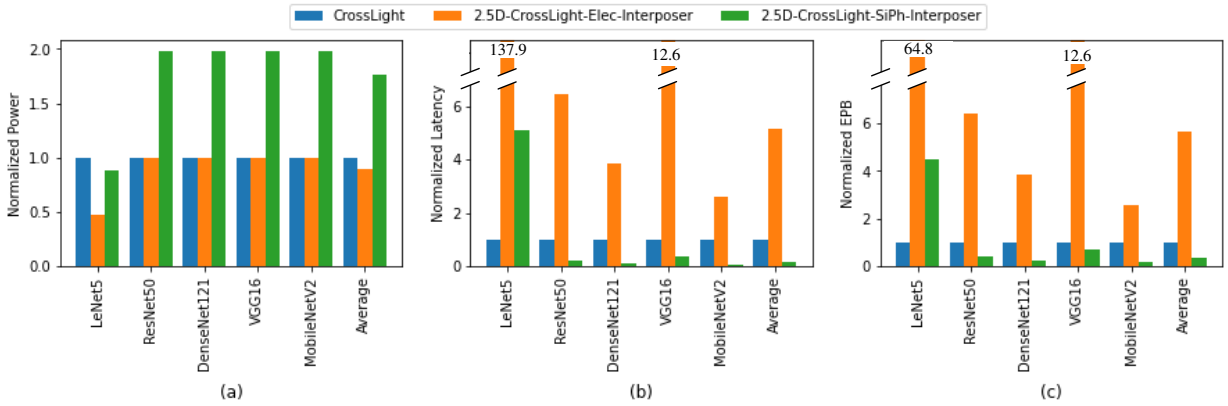



**Figure 5.4:** Performance analysis of CrossLight, 2.5D-CrossLight with electronic interposer, and 2.5D-CrossLight with silicon photonic interposer, (a) normalized power consumption, (b) normalized total latency, and (c) normalized energy-per-bit.

Elec-Interposer). We also compare the two 2.5D CrossLight variants with the original monolithic (single-chip) CrossLight architecture in terms of power, latency, and energy efficiency. The model parameters assumed in this study are summarized in Table 1. We also employ the power model and power parameters used in [79]. We consider one memory chiplet and eight compute chiplets in which two of the chiplets include dense-layer MACs and six of them include convolution layer MACs (3×3, 5×5 and 7×7 convolution MACs). We used various DNN models, summarized in Table 2, for our evaluation. The performance results are shown in 5.4. In general, 2.5D-CrossLight-SiPh-Interposer is able to achieve superior energy efficiency and latency across almost all models, except for very small ones (e.g., LeNet5). The heterogeneous chiplets and high bandwidth inter-chiplet photonic network enable more energy-efficient execution of DNNs than in the monolithic CrossLight case. 2.5D-CrossLight-SiPh-Interposer imposes a non-trivial power overhead as its photonic network consumes higher power for communication than an electronic network. However, 2.5D-CrossLight-SiPh-Interposer has lower power consumption in the smaller DNN models (e.g., LeNet5) as the ReSiPI controller reconfigures the photonic interposer and deactivates unnecessary gateways. Nonetheless, for the smaller models, where each layer only takes up a small fraction of the overall compute real estate, the 2.5D-CrossLight-SiPh-Interposer overheads become significant and adversely affect energy efficiency (e.g., LeNet5). For larger

models where multiple layers are mapped to chiplets, the 2.5D-CrossLight-SiPh-Interposer overheads in terms of power consumption are amortized across these mappings. The controller also activates gateways in the large models to cope with high traffic volumes, which helps to improve inter-chiplet latency. Although 2.5D-CrossLight-Elec-Interposer has lower power consumption, it suffers due to the significantly higher latency of metallic interconnects, especially for relatively long distances on large interposers. On average, in comparison with monolithic CrossLight, 2.5D-CrossLight-SiPh-Interposer shows 6.6× lower latency, which also results in 2.8× lower energy-per-bit (EPB). Compared to 2.5D-CrossLight-Elec-Interposer, 2.5D-CrossLight-SiPh-Interposer offers 34× lower latency and 15.8× lower EPB. Such significant improvement comes from the ability in 2.5D-CrossLight-SiPh-Interposer to select appropriate chiplets to map layers of each DNN model and tuning the required inter-chiplet bandwidth accordingly. As 2.5D-CrossLight-SiPh-Interposer performs well for larger models, such a photonics-based platform is scalable to support emerging large DNN model acceleration. We also compared 2.5D-CrossLight-SiPh-Interposer accelerator with state-of-the-art accelerators in terms of average power, latency (total latency of layers), and EPB (Table 3). 2.5D-CrossLight-SiPh-Interposer can be seen to outperform these accelerators in terms of latency and EPB.

## 5.4 Conclusion and open challenges

In this chapter, we presented a 2.5D chiplet platform-based photonic DNN accelerator where both communication on the interposer and computation on the chiplets employ silicon photonics. Compared to a monolithic photonic accelerator, a chiplet-based one not only improves fabrication yield and cost, but also reduces latency using a high-bandwidth photonic network on the interposer. Moreover, chiplets can be designed heterogeneously and off-the-shelf chiplets can be integrated in 2.5D packages to make various systems with different computation power budgets and capabilities. There are several open challenges in this field to design a more efficient silicon photonic DNN accelerator: 1) power consumption of the state-of-the-art photonic devices are relatively high, and there is a need for device-level efforts to design low-power devices; 2) designing an efficient elec-

tronic controller is essential to efficiently control the communication and computation operations with low latency; and 3) the silicon photonic 2.5D DNN accelerator architecture requires design-space exploration (e.g., in terms of the number of wavelengths, number of gateways per chiplet, and number of MACs per chiplet) to create an optimized architecture tailored to DNNs of interest.

# Chapter 6

# A Non-Blocking Silicon Photonic Switch-Based Interposer Network for 2.5D Machine Learning Accelerators

The surging demand for machine learning (ML) applications has emphasized the pressing need for efficient ML accelerators capable of addressing the computational and energy demands of increasingly complex ML models. However, the conventional monolithic design of large-scale ML accelerators on a single chip often entails prohibitively high fabrication costs. To address this challenge, this chapter proposes a 2.5D chiplet-based architecture based on a silicon photonic interposer, called SwInt, to enable high bandwidth, low latency, and energy-efficient data movement on the interposer, for ML applications. Existing silicon photonic interposer implementations suffer from high power consumption attributed to their inefficient network designs, primarily relying on bus-based communication. Bus-based communication is not scalable, as it suffers from high power consumption of the optical laser due to cumulative losses on the readers and writers when the bandwidth per waveguide (i.e., wavelength division multiplexing degree) increases or the number of processing elements in ML accelerators scales up. SwInt incorporates a novel switch-based network designed using Mach-Zehnder Interferometer (MZI)-based switch cells for offering scalable interposer communication and reducing power consumption. The designed switch architecture avoids blocking using an efficient design, while minimizing the number of stages to offer a low-loss switch. Furthermore, the MZI switch cells are designed with a dividing state, enabling energy-efficient broadcast communication over the interposer and supporting broadcasting demand in ML accelerators. Additionally, we optimized and fabricated silicon photonic devices, Microring Resonators (MRRs) and MZIs, which are integral components of our network architecture. Our

analysis shows that SwInt achieves, on average, 59.7% energy efficiency improvement compared to the state-of-the-art silicon photonic interposers specifically designed for ML accelerators.

## 6.1 Introduction

The rapid growth in demand for machine learning (ML) applications has highlighted the critical need for ML accelerators capable of efficiently performing the ever-growing computations required by ML models. These accelerators play a pivotal role in achieving energy efficiency and high-performance computing within the ML domain [101]. Therefore, there has been a surge of interest in the development of ML accelerator architectures that can meet these escalating demands [43].

Nevertheless, the conventional monolithic design of large-scale ML accelerators on a single chip often leads to prohibitively high fabrication costs, primarily due to low-fabrication yields [102]. Addressing this challenge requires innovative approaches, and one such promising strategy is the adoption of a 2.5D chiplet-based architecture [43, 45, 103–105]. In this paradigm, the large-scale accelerator is disintegrated into multiple smaller chiplets, and interconnected through an interposer [52]. These chiplets are linked to the interposer using microbump technology, with the interposer network serving as the vital communication backbone among them.

In the context of 2.5D ML accelerators, a common approach involves the utilization of Multiply-and-Accumulate (MAC) chiplets, alongside a Global Buffer (GLB) chiplet, as seen in state-of-the-art designs [43, 101, 102, 104]. The GLB chiplet plays a crucial role in storing the weights and activations required for extensive MAC operations. However, efficiently managing data exchange between the GLB chiplet and the MAC chiplets poses a significant challenge to interposer network design, mainly due to the substantial volume and time criticality of data involved.

There are three types of interposers: passive, active, and SiPh interposers. The first two are metallic/electronic interposers, while the latter is an optical interposer designed to support high bandwidth and low latency communication. Passive interposers [106] consist of simple substrates with only wiring layers, offering simpler connectivity between chiplets with lower power consumption but with limited capabilities for long-distance communication. On the other hand, active

interposers [23,58] incorporate complex functions, enabling diverse chiplet communication despite yield and thermal challenges. However, both passive and active electronic interposers suffer from low bandwidth and high latency, especially when scaling up the system [25, 50, 107]. Therefore, electronic interposers are not suitable candidates for large-scale future ML accelerators where high bandwidth is required due to communication-intensive applications, and high latency might be imposed due to the large-scale system and one-to-many and many-to-one communication patterns. SiPh interposers not only offer the high bandwidth and low latency requirements for ML accelerators but also support broadcast communication, where one source can send the same data to all destinations and vice versa, in an energy-efficient manner [103, 104]. However, SiPh interposers designed for ML accelerators [103, 104] employ bus-based communication, which is not energy efficient when supporting a large-scale system.

To overcome this challenge, this chapter proposes the integration of switch based SiPh interposers, offering attributes such as low latency, high bandwidth, and energy-efficient communication between the GLB chiplet and the MAC chiplets. While SiPh interposers have exhibited the potential to enhance communication efficiency, recent implementations [101, 103, 104] have suffered from heightened power consumption, primarily stemming from inefficient network design. In response to this issue, this chapter introduces SwInt, a non-blocking Switch-based SiPh Interposer for 2.5D ML Accelerators. The main contributions of this chapter are as follows:

- We develop a Butterfly-based interposer network topology to minimize the switch stages and improve the energy efficiency of 2.5D ML accelerators.
- We design new techniques to remove conflicts in the switch and avoid blocking.
- We propose a broadcast architecture with minimal laser power overhead.
- We optimize photonic devices to improve the energy efficiency of SwInt's switch and support the broadcast architecture.
- We explore and compare a switch-based architecture versus a bus-based one and offer a hybrid architecture.

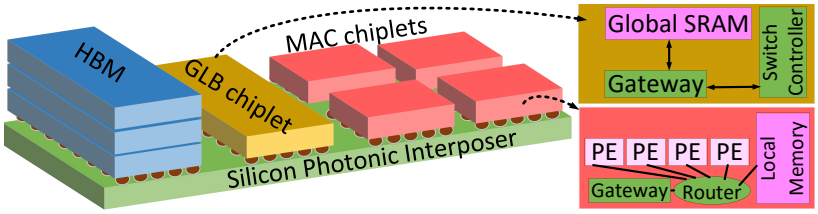



**Figure 6.1:** Chiplet system architecture considered in SwInt.

The organization of this chapter is structured as follows: In the upcoming section, we delve deeper into the background and related work. Following that, in Section 6.3, we present our proposed architecture. Section IV involves a comprehensive design space exploration of the switch architecture and the MZI switch cell. Our simulation results are showcased in Section 6.5. Additionally, in Section Section 6.6, we explore a hybrid architecture of SwInt with switch and bus. Finally, Section 6.7 concludes the chapter and highlights the significance of our contributions.

## 6.2 Background and Related Work

### 6.2.1 Silicon Photonic Communication

In Fig. 6.2, we illustrate a bidirectional SiPh link setup. The GLB chiplet utilizes modulators to modulate electrical signals on optical ones generated by a laser. These modulators can be designed based on microring resonators (MRRs), tuned to resonance at desired wavelengths [13]. The modulated signal is transmitted through a dedicated bus waveguide (depicted in dark green) to the MAC chiplet. The MAC chiplet retrieves the optical signal from the bus waveguide using optical filters, which are custom-designed with MRRs tuned to filter the modulated wavelength. The same process occurs when the MAC chiplet communicates with the GLB chiplet. However, this specific bus-based setup, recognized as a single-writer single-reader (SWSR) configuration, lacks scalability. As the number of writers and readers increases, substantial power is required from the laser source to compensate for optical losses incurred during transmission, including through loss, waveguide crossing losses, and bending losses. To improve scalability, optical switches have been used to devise more scalable network architectures. The switche cell typically feature four ports and can establish either a Bar or Cross connection between two input and output ports (see our

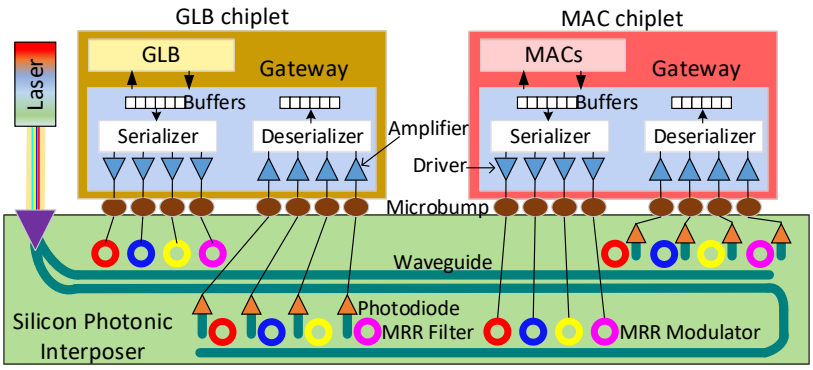



**Figure 6.2:** A bidirectional silicon photonic link with four wavelengths to provide optical communication between the GLB chiplet and a MAC chiplet.

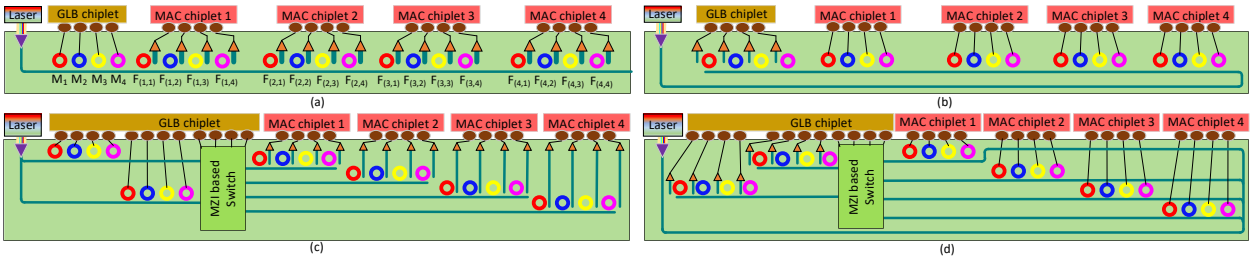



**Figure 6.3:** (a) GLB to MAC chiplets communication via a bus-based approach, (b) MAC chiplets to GLB communication via a bus-based approach, (c) SwInt facilitating GLB to MAC chiplets communication using a switch-based interposer, and (d) SwInt enabling MAC chiplets to GLB communication.

designed MZI switch later in Fig. 6.7). In the Bar state, each input is linked to a symmetrical output port, whereas in the Cross-state, as the name implies, each port connects to the crossing output port. By connecting these four-port switch cells in a specific topology, large-scale switches can be designed to offer high bandwidth while accommodating a greater number of inputs and outputs.

### 6.2.2 Silicon Photonic Interposers

SiPh interposers have received attention in the context of emerging many-core architectures. Fotouhi et al. [80] introduced an interposer design based on Arrayed-Waveguide Grating Routers (AWGRs) to address the latency issues associated with electronic interposers. AWGRs, functioning as passive optical devices that route data by wavelength, offer cost-effectiveness and higher bandwidth compared to their electronic counterparts. However, it is worth noting that achieving high bandwidth with AWGR-based interposers necessitates a significant number of wavelengths, which can result in less power efficiency. PROWAVES [79] and ReSiPI [25] utilize bus-based

communication with a single-writer-multiple-reader (SWMR) protocol for inter-chiplet communication while dynamically managing bandwidth. However, as we will show in Section 6.3, such bus-based communication is power efficient only for small-scale systems with a small number of writers and readers. Another work, FLUMEN [108], introduces in-network computing within photonic interposer networks, combining communication and computation. While it enables parallel linear computation during low network loads, this approach may not be ideal for ML applications due to their high data communication demands on the interposer network.

SPRINT [104] is another SiPh interposer network designed to facilitate inter-chiplet communication for 2.5D Convolutional Neural Network (CNN) accelerators. It relies on point-to-point SiPh links to establish high-bandwidth, low-latency communication channels between the GLB chiplet and individual MAC chiplets. However, SPRINT's scalability is hindered by the need for separate optical links for each receiver, potentially leading to inefficiencies. While it offers dynamic reconfiguration for broadcast communication, this feature comes at the cost of high laser power tuning and increased latency. SPACX [103] and ASCEND [101], propose SiPh interposer networks tailored for neural network accelerators, that address SPRINT's scalability limitations. Although these networks achieve improved scalability, the proposed solutions introduce challenges related to laser power scalability due to waveguide sharing. To compensate for losses incurred when optical signals traverse multiple receivers, higher optical power is required, resulting in increased energy consumption. In Section 6.3.2, we delve into the energy consumption implications of SiPh bus-based communication in large-scale ML accelerators and motivate our SiPh interposer network architecture.

To improve bus-based architectures, in [51], we introduced TRINE as a silicon photonic interposer network based on the SiPh switches, aimed at facilitating energy-efficient ML acceleration. However, TRINE utilizes multiple sub-networks with tree topologies to handle traffic between the GLB chiplet and the MAC chiplets, and vice versa. Although the idea of employing multiple sub-networks is straightforward, it suffers from limitations in flexibility and efficiency of communication. The inherent constraints of the tree topology result in restricted bandwidth between GLB

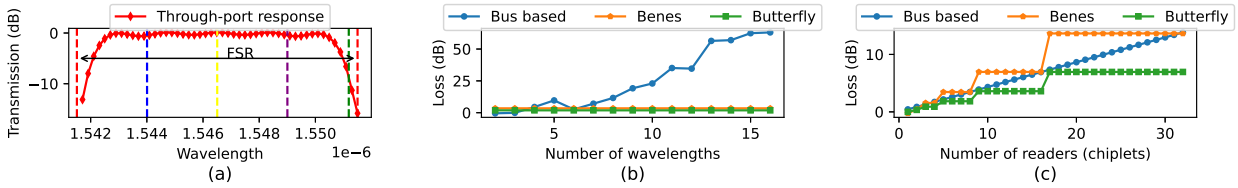



**Figure 6.4:** (a) MRR's Through-port response and optical loss under different wavelengths. (b-c) power loss of filters in bus-based network compared to power loss of Benes and Butterfly networks.

and MAC chiplets, and the utilization of several sub-networks further constrains flexibility as communication is confined within local sub-networks. Additionally, TRINE is incapable of efficiently supporting broadcast communication.

## 6.3 Proposed Interposer Network: SwInt

### 6.3.1 Architecture of the 2.5D Accelerator in SwInt

The 2.5D accelerator architecture depicted in Fig. 6.1 serves as the basis for SwInt. It consists of multiple MAC chiplets, a GLB chiplet, and an HBM main memory. This design aligns with prior research efforts [102–104], where an SRAM-based GLB is strategically employed to store the weights and activations required by all MAC chiplets. Each chiplet features a gateway responsible for data storage and forwarding between the GLB and the individual chiplets. These gateways play a pivotal role in controlling the modulator and filter MRRs on the interposer, to facilitate efficient optical communication. The MAC chiplets comprise several processing elements (PEs) interconnected through routers, with each PE housing a MAC unit capable of performing a set of parallel MAC operations. While our primary focus is to design an interposer network for low-latency and energy-efficient communication within this architecture, our network's design principles can be applied to other 2.5D ML accelerators without a loss of generality.

### 6.3.2 Motivation for Switch-based Network Architectures

As previously mentioned, state-of-the-art SiPh interposer networks have been predominantly designed around bus-based communication, a popular choice for small-scale Network-on-Chip

(NoC) configurations. However, numerous limitations become exacerbated in the context of large-scale interposer networks. While bus-based communication offers a smaller physical footprint, it exhibits energy inefficiency and lacks the necessary flexibility required for larger systems. For example, in Fig. 6.3(b), a bus network with SWMR necessitates the optical signal to traverse multiple readers to reach chiplet 4 (the worst-case scenario for defining the required laser power intensity). A similar situation arises in the multiple-writers single-reader (MWSR) case, as depicted in Fig. 6.3(b), where the writer of chiplet 4 must pass through several MRRs to reach the GLB. In contrast, our switch-based design, illustrated in Fig. 6.3(c) and Fig. 6.3(d), addresses these concerns. In the following, we discuss the challenges with bus-based networks in more detail and advocate for a scalable switch-based interposer network.

**Bandwidth and Adaptation**

Bus-based communication struggles to offer high bandwidth efficiently. One potential solution involves increasing the degree of wavelength-division multiplexing (WDM), enabling the transmission of a larger number of wavelengths over the same waveguide. However, this approach can introduce significant challenges, including higher optical power losses and crosstalk noise. As more wavelengths are transmitted in proximity, there is an increased likelihood of signal leakage to nearby filters with close resonant wavelengths, resulting in optical power loss and crosstalk.

**Power Inefficiency**

In Fig. 6.4(a), we present the frequency response of an MRR filter on the Through port. The y-axes depicts the transmission of the Through port relative to the input port. The resonant wavelength is indicated by the red lines. In an ideal scenario, 0-dB loss should occur on the Through port for wavelengths other than the resonant one (blue, yellow, and purple). In this scenario with a small number of wavelengths (i.e., four), all the three wavelengths, which are not the intended resonant ones, exhibit nearly 0-dB loss. However, as additional wavelengths are introduced to expand bandwidth, the resonance frequencies of these new wavelengths become closer to that of the red wavelength, potentially leading to power losses. For instance, the green wavelength expe-

riences significant loss in the case of using 32 wavelengths on the same waveguide. This concern exponentially worsens as the number of reader chiplets increases. However, as discussed next, our switch-based approach remains unaffected because the signal does not traverse numerous readers or writers.

We assess the bus-based network alongside two switch-based counterparts: Benes and Butterfly topologies. In our evaluation of the switch-based networks, we exclusively account for losses stemming from the switch itself, encompassing waveguide crossings and MZI switch cells. Conversely, for the bus-based network, we solely factor in the Through losses of MRRs. This delineation reflects the primary distinctions between these two communication paradigms. In Fig. 6.4(b), we investigate scenarios involving 6 reader chiplets while varying the number of wavelengths from 1 to 16. In Fig. 6.4(c), we maintain 6 wavelengths and alter the number of readers from 1 to 32. Our findings indicate that switch-based networks employing Benes and Butterfly topologies offer substantially enhanced scalability. Notably, the Butterfly topology exhibits lower power loss due to its minimized number of switch stages when compared to Benes. However, this power-saving advantage comes at the cost of increased blocking. In contrast, as we will show later, SwInt addresses this issue by minimizing blocking while retaining the same number of stages.

**Multicast and Broadcast Challenges**

Prior work [101, 103] assumed that broadcast operations in bus-based SiPh networks offer minimal energy overhead, but such networks present their own set of challenges when used in large-scale systems. Typically, passive filters are considered [25, 79, 101] and in both communication modes, unicast and broadcast, worst-case power loss is considered. However, tuning the laser power for the worst-case scenario (i.e., broadcast) imposes severe energy inefficiency on the laser when unicast mode is used. Considering active filters imposes latency and power overhead to tune filters accurately for all communication modes and routing scenarios. In SwInt, unicast and broadcast operations can be efficiently managed, while minimizing the laser power consumption. Even if considering active filters, in prior work [101, 101, 103], the laser power for broadcast and unicast is considered the same instead of dynamic laser power management. On the other hand, in

SwInt, dynamic power management is achieved via dynamic laser power tuning at the MZI splitter, which will be discussed in more detail in Section 6.3.5.

Furthermore, broadcasting in bus-based architectures presents challenges that result in high power consumption and area overhead. As illustrated in the example depicted in Fig. 6.3(a), $M_i$ sends data to $F_{(i,j)}$, where $i$ denotes the wavelength, and $j$ represents the reader (chiplet). For clarity, we focus on communication involving the red wavelength and consider two scenarios: 1) unicast, where $M_1$ modulates data for the first chiplet ($F_{(1,1)}$); and 2) broadcast, where all the chiplets receive the data.

To efficiently utilize laser power, each filter must be precisely tuned to deliver a specific portion of the optical signal to the photodetector, with the remainder passed on to other readers. In the unicast scenario, $F_{(1,1)}$ is simply tuned to capture the entire signal with the red wavelength. However, in the broadcast scenario, $F_{(1,1)}$ should be tuned to receive only $1/4$ of the signal, and the rest of the signal should be passed to the other readers. Thus, the filter should be tuned to receive $1/4$ of its input and pass $3/4$.

$F_{(2,1)}$, $F_{(3,1)}$, and $F_{(4,1)}$ should be tuned to receive $1/3$, $1/2$, and $1/1$ of their input, respectively. Regardless of area and controlling overheads, achieving exact tuning of the filters might not be possible due to process and thermal variations [86]. Therefore, laser power should be increased to compensate for such issues, which further results in energy inefficiency.

A similar concept applies to the multiple-writers single-reader (MWSR) protocol, as shown in Fig. 6.3(b). In our switch-based approach, multicast and broadcast operations can be efficiently managed, while minimizing laser power consumption. Therefore, as shown in our power analysis in Section 6.5.2, SwInt offers higher power efficiency when used in broadcast mode.

### 6.3.3 SwInt Switch Topology

In this chapter, our proposed switch topology is based on the Butterfly topology. The Butterfly-based topology of SwInt is the ideal choice for our large-scale ML accelerators, because of its minimized number of stages to improve the footprint and power consumption of the laser. When

considering the number of receivers as N, the stages in the Butterfly topology can be calculated as:

$$NS_{But} = \lceil \log_2 N \rceil \tag{6.1}$$

In contrast, topologies like Benes, with more switch stages, increase costs and power consumption. Additionally, the adaptability of Butterfly topology compared to a traditional bus-based network, though adding routing complexity, provides multiple routing paths between the GLB chiplet and the MAC chiplets, crucial for optimizing ML accelerator performance in dynamic workloads.

However, the original Butterfly switch may experience blocking for specific input-output request combinations. To address this, we optimize the switch topology to effectively reduce instances of blocking. The blocking insances can be improved by adding extra stages. However, adding an extra stage to the switch introduces losses, as each stage requires the signal to pass through an additional MZI, incurring inherent losses within the MZI components. This underscores our commitment to minimizing the number of stages to mitigate such losses and maintain overall efficiency.

In addition to the inherent loss within the switching elements (i.e., MZI), the number of waveguide crossings increases proportionally to the number of stages. The total number of waveguide crossings in the Butterfly topology can be calculated as follows:

$$C_{But} = \sum_{n=1}^{NS_{But}} 2^n + (2^n - 1) \tag{6.2}$$

Therefore, the worst-case loss of the switch from input ports to any output port is given by:

$$L_{But} = NS_{But}. \max(L_{MZI_{Cross}}, L_{MZI_{Bar}}) + C_{But}.L_{WG_{Cross}} \tag{6.3}$$

Here, $L_{MZI_{Cross}}$ represents the loss from an input port of MZI to the Cross power of MZI, while $L_{MZI_{Bar}}$ denotes the loss from an input port of MZI to the Bar power of MZI. Additionally,

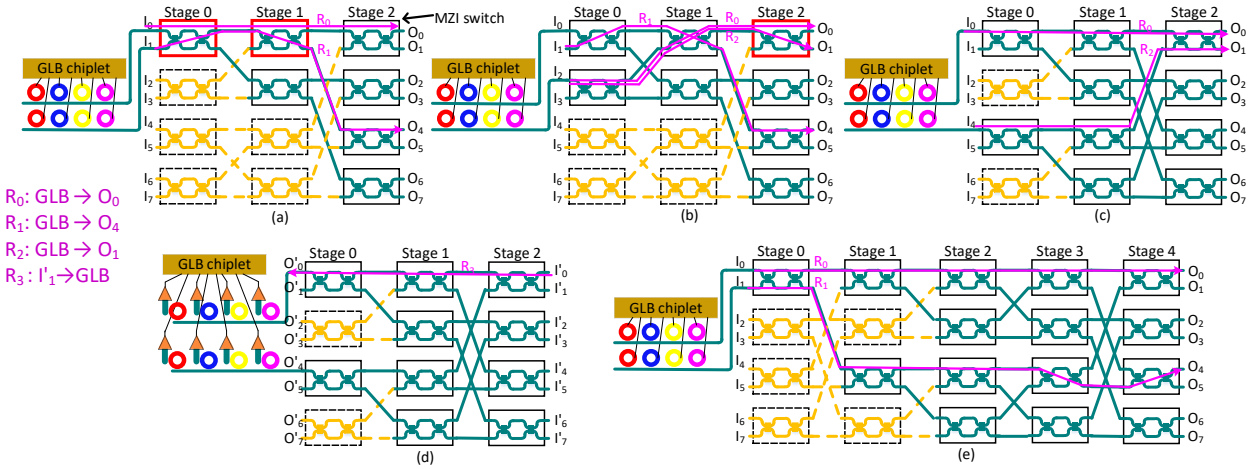



**Figure 6.5:** An example of (a-b) Butterfly topology with suboptimal input selection, (c) SwInt network for GLB to MAC chiplets communication, (d) SwInt network for MAC to GLB chiplets communication, and (e) Benes topology. Orange switches are removed compared to a typical Butterfly/Benes topology. Red switches suffer from blocking.

$L_{WG_{Cross}}$ represents the loss of waveguide crossing. A flattened butterfly to improve the crossings is also presented in [109].

Now, let's examine the key distinctions between the SwInt topology and the original Butterfly topology.

Our proposed switch topology, shown in Fig. 6.5(c) and Fig. 6.5(d), is based on the Butterfly topology. The Butterfly-based topology of SwInt is the ideal choice for our large-scale ML accelerators, because of its minimized number of stages to improve the footprint and power consumption of the laser. While the standard Butterfly topology, with its typical number of inputs and outputs, may encounter blocking issues, we optimize SwInt to minimize such blocking, thereby enhancing the overall performance. Blocking occurs when competing requests for the same switch cell result in conflicts, hindering the immediate fulfillment of those requests. The number of inputs is inherently smaller than that of a typical Butterfly switch, which reduces the occurrence of blocking scenarios. This is primarily due to the communication pattern within the chiplet-based ML accelerator, which is one-to-many (from the GLB to the MAC chiplets) and many-to-one. Please note that another switch is employed to facilitate communication from MAC chiplets to the GLB chiplet, utilizing the same topology but with inputs and outputs swapped. Due to brevity, we do

not delve into the details of this additional switch in this discussion. As shown in Fig. 6.5(e), the Benes topology offers non-blocking communication at the cost of a larger number of stages and power loss.

There are two factors considered in the SwInt topology to minimize blocking scenarios in comparison with the original Butterfly: 1) input swapping and 2) input selection. The *input swapping* essentially explores various input combinations to establish connections to outputs without causing blockages. This process incurs no performance overhead as all input lines connect to the same source. In Section 6.3.4, we will delve into our offline routing algorithm, which seeks to minimize blocking by evaluating different input swapping options. The *input selection* technique identifies the input combination that minimizes blocking. Fig. 6.5 shows examples of how SwInt reduces blocking with better input selection. In Fig. 6.5(a), blocking occurs when considering $R_0$ and $R_1$ due to conflicts in switch cells (red switch cells), while in Fig. 6.5(b), blocking arises when routing $R_0$ and $R_2$. However, in Fig. 6.5(c), SwInt's *input selection* technique is employed, rendering this topology example non-blocking. Our designed algorithm, shown in Algorithm 6, automatically selects the optimized inputs for various combinations of switch sizes and input numbers. Please note that in Fig. 6.5(d), our switch network supports communication from MAC chiplets to GLB chiplets, akin to the GLB-to-MAC communication but with fewer output links to GLB chiplets. The size of these two networks may vary depending on the bandwidth requirements dictated by the accelerator dataflow.

**Algorithm 6** Input selection to generate SwInt topology

$Con\_list$: List of connected inputs to the GLB
$L_{GLB}$: Optical communication line from GLB
$L_{SW}$: Communication lines of Butterfly = Number of gateways
$L \leftarrow 0$
**while** Length of $Con\_list < L_{GLB}$ **do**
  **for** $i$ in range $L_{SW}$ **do**
    **if** $i$ not in $Con\_list$ & length of $Con\_list < L_{GLB}$ **then**
      **if** $i\%(L_{SW}/(2^L)) == 0$ **then**
        Add $i$ to $Con\_list$
    $L \leftarrow L + 1$

### 6.3.4 Routing Strategies and Optimization in SWInt

Switch configuration for routing from inputs to outputs can be accomplished through online or offline methods. The online configuration offers flexibility to adapt to various applications but can become complex, especially for large-scale switches, even though the butterfly switch uses straightforward routing. Fortunately, ML applications typically feature predictable and predefined dataflow communication patterns. In SwInt, we have chosen to implement offline routing analysis, where we optimize switch configurations in advance and store them in a configuration table. During runtime, the switch configuration controller employs this table to configure the switch according to the dataflow. It is worth noting that we plan to explore the development of an online configuration controller for SwInt in future work, extending its versatility to support a broader range of applications. Our offline routing algorithms explore potential combinations of simultaneous outputs within the communication tasks of accelerator dataflow. Once a non-blocking configuration is successfully identified, considering the input swapping technique, it is recorded in the configuration table. In cases where the algorithm cannot find a non-blocking configuration for a particular combination, it is marked for further processing, with outputs being handled one at a time, although our evaluation in Section 6.5.2 shows no blocking even with a high GLB chiplet bandwidth.

### 6.3.5 Multicast and Broadcast Capabilities in SWInt

To enable multicast and broadcast functionality within our SiPh switch, we introduce a new state for our switch cell (our device design of the MZI is elaborated in Section 6.4). In addition to the Cross and Bar states, traditionally used in optical switches for unicast communication, this new state, called Divide state, allows the input signal to be split between Cross and Bar outputs. This Divide state empowers the switch to direct data to multiple intended outputs concurrently. As depicted in Fig. 6.6, several switches can be configured in a tree-like pattern, efficiently splitting input data to all the designated destinations. However, to support the receivers adequately, the laser power of the input line needs to be increased. To achieve this, we incorporate an MZI at the input

of the first line, serving as an active splitter. This active splitter directs the entire input power of the laser to this line, ensuring sufficient power for all intended receivers. Proper tuning of the laser power is also essential to achieve optimized power in this multicast and broadcast configuration. Algorithm 7 outlines the procedure for managing the SwInt interposer network. The splitting ratio in the split state is:

$$R_S = \frac{1}{L_{GLB} - 1}, \tag{6.4}$$

where $L_{GLB}$ is the number of communication lines at the GLB chiplet (switch inputs). We define $L_{GLB}$ to be matched with the GLB chiplet bandwidth:

$$L_{GLB} = \frac{B_{GLB}}{F_M \times N_w}. \tag{6.5}$$

Here, $B_{GLB}$ is GLB chiplet bandwidth. $F_M$ and $N_W$ are SiPh modulation frequency and number of wavelengths, respectively.

**Algorithm 7** SwInt network management procedure

```
for All communication tasks in the dataflow do
    if Broadcast or multicast then
        Laser power ← Multicast power (∝ number of readers)
        Switches of the broadcast path ← Divide state (50:50)
        Tunable splitter ← Split state (R_S)
    else
        Laser power ← Unicast power
        Tunable splitter ← Bar state
        Switch is configured according to the configuration table
```

## 6.4 Design and evaluation of SwInt's devices

### 6.4.1 MZS Design and Optimization

MZI-based switches (MZS) are one of the most well-known and extensively used devices for optical switching. An MZI switch consists of two 3-dB couplers that are connected to each other by two waveguide arms. By tuning the phase of either of the arms, the MZI can operate as a

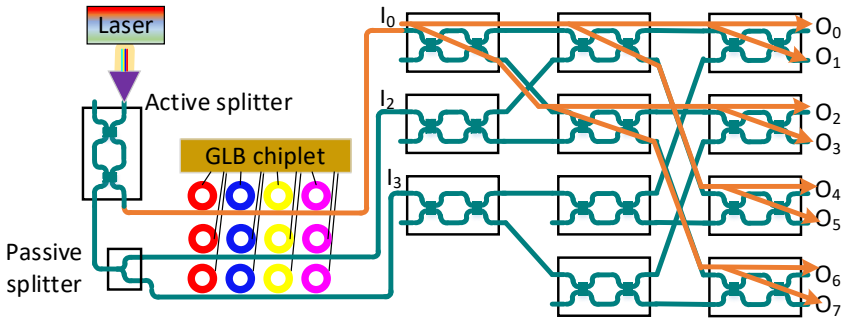



**Figure 6.6:** An example of a broadcast in SwInt, achieved using the Divide state in the MZI switch.

switch to route either of the two inputs to either of the two outputs. The 2×2 couplers in the MZI structure are the main source of wavelength dependency and can be realized through multi-mode interference (MMI) couplers since they have higher fabrication tolerance compared to directional couplers and can provide a more uniform 50:50 splitting ratio over the C-band($1.53 \sim 1.56$ $\mu$m) the fundamentals of MMI operation further discussed in details in [110]. Fig. 6.7(a) showcases our MZS structure. In this design, by doping silicon to form p-n junctions and applying a voltage across the junction, carriers (electrons and holes) can be either injected into or depleted from the silicon waveguide region. This process changes the carrier concentration in the silicon, which in turn modifies its refractive index through the plasma dispersion effect causing the light traveling through the injected region to experience a different optical path length compared to the light in the non-injected arm. This difference in optical path length leads to a phase shift, altering the interference pattern when the light waves are recombined at the output, and thereby modulating the output signal. Fig. 6.7(b,c) demonstrates an adiabatic microring resonator that is discussed in the following section.

Fig. 6.8(a-c) shows three different states of the MZS. In Fig. 6.8(a), the MZS is in the default cross state, and that light is switched to the opposite output port. Fig. 6.8(b) shows the bar state of the switch where light exits through the same output port as it would by inducing a $\pi$ phase shift. Fig. 6.8(c) shows a 50/50 dividing state where the light is evenly split between the two output ports, achieved by precise interference via inducing a $\pi/2$ phase shift. The intricate manipulation and control over these states are managed by optical network controllers [111].

MMI coupler operations are based on multimode interference, where, at specific lengths ($L_{MMI}$) and widths ($W_{MMI}$) of the MMI, the optical power can be equally divided between the two outputs [110]. Unlike traditional design methods that proceed from known design parameters to performance outcomes, we employed an inverse design approach for the MMI coupler. This method involves specifying the desired outcome—in our case, the maximization of the figure of merit (FOM)—and using computational algorithms to determine the optimal physical parameters (e.g., $L_{MMI}$, $W_{MMI}$) to achieve that outcome. Inverse design is crucial for our project because it enables the exploration of a broader design space, potentially uncovering innovative configurations that traditional methods might overlook. This approach is particularly beneficial for enhancing the performance of the MMI coupler, as it systematically identifies the design parameters that optimize the FOM, defined by:

$$FOM = -(|(T_{Cross} - 0.5)| + |(T_{Bar} - 0.5)|) \tag{6.6}$$

using a particle swarm optimization (PSO) technique and using Lumerical FDTD simulation where the $T_{Cross}$ and $T_{Bar}$ are the transmission of the Cross and Bar outputs of the MMI coupler. For better mode matching between the input waveguides and the MMI modes, the input waveguides are connected to the MMI structure using tapers, and their geometry is defined by:

$$w(x) = \alpha(L - x)^m + w_2. \tag{6.7}$$

In this equation, $m$ determines the curvature of the taper and $\alpha = (w_1 - w_2)/L^m$. Here, $w_1$ and $w_2$ denote the widths of the waveguides at the start and end of the taper section, respectively, while $L$ represents the taper's length. The optimized dimension for the $2 \times 2$ MMI is calculated to be: $L_{MMI} = 17.1\mu$m, $W_{MMI} = 2.2\mu$m, $L_{Taper} = 4.37\mu$m where L and W representing the length and width (see Fig. 6.7(a)), and the optimum values for parameters in Equation 6.7 are $m = 1.75$, $w_1 = 500$ nm, $w_2 = 1.08\mu$m , $gap = 680$ nm. We used Lumerical FDTD and DEVICE simulation tools to design and optimize our optical devices. Simulations showed a worst-case

transmission loss of a single MMI coupler with these specifications to be 0.02 dB with a power splitting imbalance of less than 0.14 dB over the entire C-band. The designed MZI based on this MMI coupler can achieve a worst-case low loss of 0.12 dB in the Cross state and 0.5 dB in the Bar state using the electro-optical (E-O) tuning method [86]. The lengths of the MZI arms are considered to be 200 $\mu m$ and the $V_\pi$ is calculated to be 1.3V with a switching time of $5.7ns$.

To validate our design, we fabricated our shape-optimized MMI couplers through Applied Nanotools Inc. (ANT) foundry utilizing their 220 nm silicon-on-insulator (SOI) technology. The fabrication process was meticulously planned and executed to ensure that the MMI couplers adhered to our precise design specifications, optimizing their performance capabilities. The collected fabrication results from the MMI couplers show a wide band response over the C-band and the measured insertion loss for the central wavelength of operation is calculated to be 0.17 dB. The measured response of the fabricated device is shown in Fig. 6.9(a) over the C-band central wavelength. It can be observed that the best performance of the MMI is achieved around 1550 nm where the device is optimized and the least power imbalance happens at the same wavelength range. However, the shape optimization of the MMI could provide an outstanding imbalance between the two channels over a wide range of wavelengths. Further comparative analysis among the various fabricated devices was performed to ascertain the consistency and reliability of our MMI designs. A focal point of this evaluation was the determination of the maximum tolerance for power imbalance between the two outputs of the MMI. Our findings indicated a maximum imbalance tolerance of $<$0.5 dB, underscoring the robustness and precision of our design in maintaining uniform output levels under fabrication variations.

This crosstalk resulting from the power imbalance of our optimized MMI is more pronounced as the systems get larger. However, since the number of stages does not increase linearly and our device optimizations have minimized the imbalance, the total loss due to imbalance does not introduce a large overhead to the laser power. In Fig. 6.10, we show the imbalance effect of the "Divide" state on the scalability of broadcasting. In this figure, the total loss imposed by imbalance is compared as the size of the switch increases, and it can be seen that it does not increase linearly.

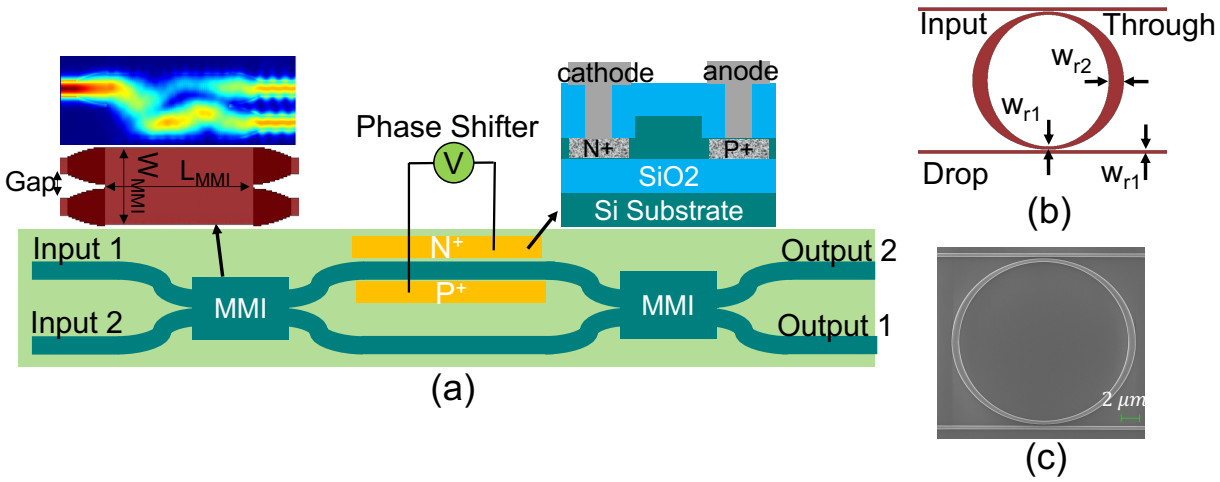



**Figure 6.7:** (a) An MZS including the MMI couplers and PN junction phase shifter. (b) an adiabatic MRR ($w_{r2} > w_{r1}$), and (c) SEM image of our fabricated Adevice.

Moreover, the introduction of the 'Divide' state in the MZI switch, allowing for simultaneous transmission to multiple outputs, enhances broadcasting capabilities in an energy-efficient manner.

### 6.4.2 Micro Ring Resonator Design and Optimization

MRRs find extensive application in diverse fields, including filtering, switching, and modulation. Nevertheless, conventional MRR designs are susceptible to process variations, which can unexpectedly shift their resonance frequencies. Addressing these deviations necessitates corrective measures that, unfortunately, come at the expense of increased power consumption and more complex system designs. To overcome these challenges, [112] has designed adiabatic structures to improve the MRRs tolerance toward fabrication errors resulting in smaller resonance wavelength shift per ring ($\Delta\lambda_R$) (see Fig. 6.7(b)). Adiabatic microring resonators (AMRs) offer significant advantages over traditional microring resonators due to their unique design and operational principles. By incorporating an interior wall design, AMRs effectively cut off higher-order modes that are conventional designs. This design feature ensures that only the lowest-order radial mode propagates, eliminating the adverse effects of higher-order radial modes [113]. Besides, AMRs demonstrate a wide, uncorrupted FSR that is beneficial for applications requiring a broad optical bandwidth.

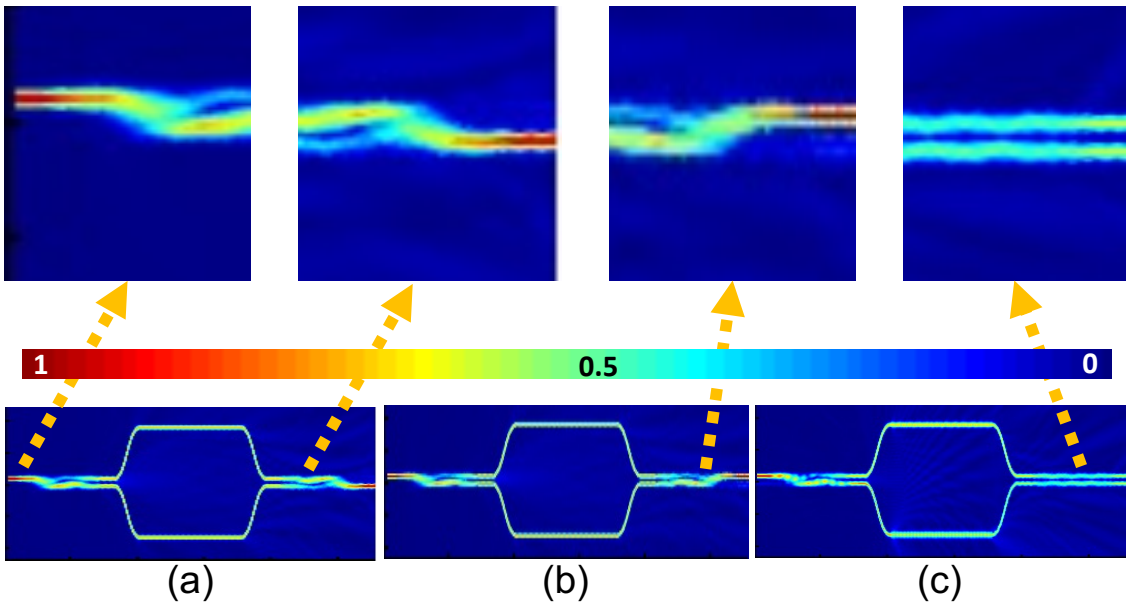



**Figure 6.8:** (a) Field distribution of Cross-state (b) Bar-state, and (c) Divide-state.

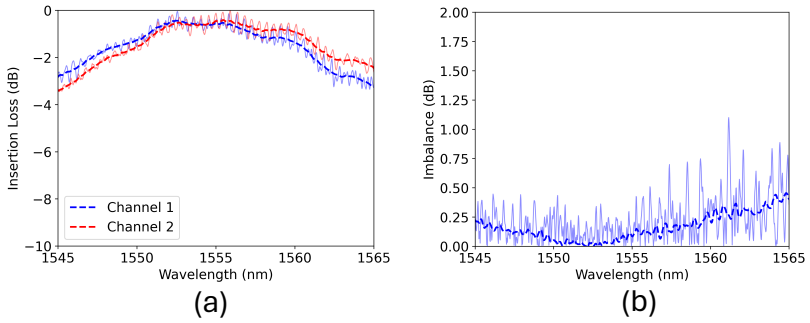



**Figure 6.9:** (a) The response of the fabricated MMI. (b) The power imbalance between the two outputs of the MMI.

To enhance our understanding and validation of AMRs, we conducted a comprehensive analysis that included both simulations and experimental fabrication. The device, as depicted in Fig. 6.7(c), revealed a drop loss of $0.5$ dB, attributable to additional losses encountered during testing, whereas our simulations predicted a slightly lower loss of $0.3$ dB. The dimensions of the waveguide widths denoted as $w_{r1}$ and $w_{r2}$ and illustrated in Fig. 6.7(b), were carefully chosen to be $500$ nm and $850$ nm, respectively. Moreover, the design incorporated a waveguide-ring gap of $100$ nm, and the ring itself featured a radius of $10\mu$m. This strategic configuration resulted in a significant enhancement in power efficiency within the switching interface (SwInt) per MRR,

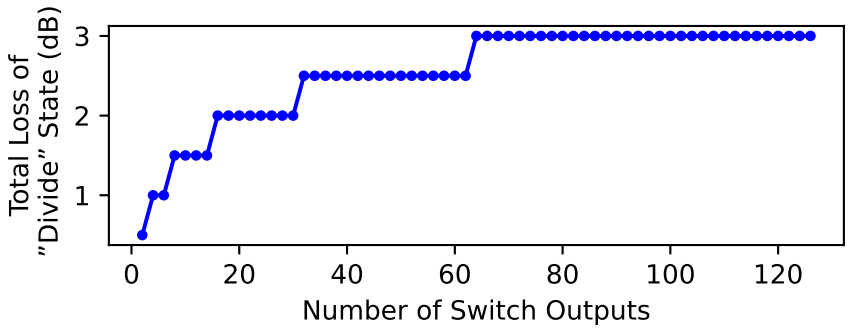



**Figure 6.10:** Power imbalance effect of the "Divide" state on the scalability of broadcasting.

**Table 6.1:** Simulation setup of SwInt.

| | | | |
|---|---|---|---|
| Modulation freq. | 12 GHz [25] | MAC chiplets | 8 |
| Wavelengths | 16 [79] | MACs per Gateway | 4 |
| PD sensitivity | 9 dBm [114] | PEs per chiplet | 16 [43] |
| Bending loss (90°) | 0.01 dB [81] | Data resolution | 8 bits [43] |
| Coupler loss | 4.55 dB [115] | Vector MAC Width | 8 [43] |
| Y-splitter loss | 0.2 dB [103] | Vector MACs (per PE) | 8 |
| Laser efficiency | 10% | Weight(per PE) | 32 KiB [43] |
| Gateway freq. | 2 GHz | Input buffer(per PE) | 8 KiB [43] |
| Gateways | 4 (per chiplet) | Output buffer(per PE) | 3 KiB [43] |

demonstrating a 50% reduction in the average wavelength shift ($\Delta\lambda_R$). This strategic optimization notably influences the tuning power required for adjusting the resonant frequencies of the rings, leading to enhanced energy efficiency during device calibration [111]. Tuning in this context refers to the precise adjustment of the resonant frequencies of the AMRs to match specific wavelengths, optimizing the device's performance. Furthermore, our AMR design achieved a FSR of $9.9$ nm, complemented by a channel spacing of $0.62$ nm across 16 wavelengths, as detailed in Section 6.5.1. This meticulous design ensured that the average Through-port loss per AMR was maintained at a remarkably low level of $0.02$ dB, thereby illustrating the efficiency and effectiveness of our AMR in optical signal processing applications.

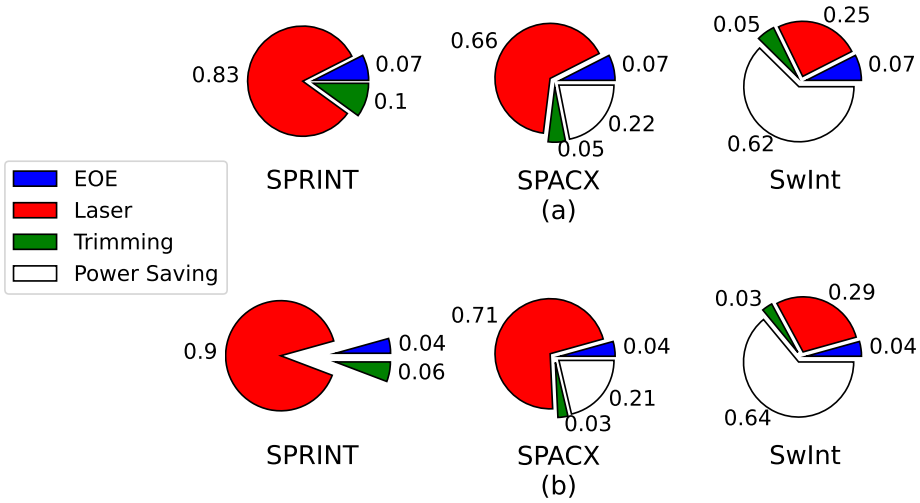



**Figure 6.11:** Power analysis: (a) Unicast state, and (b) Broadcast state. White portion in SPACX and SwInt shows power saving.

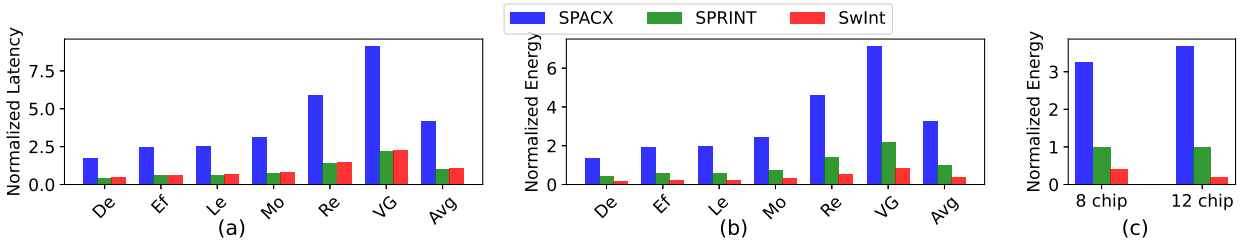



**Figure 6.12:** (a) Latency analysis, (b) energy analysis (the acronym represents models' name listed in Section 6.5.1), and (c) scalability analysis. The results are normalized to average case (Avg) in SPRINT

## 6.5 SwInt's Architecture Evaluation

### 6.5.1 Simulation Setup

We conduct a comprehensive comparison involving SwInt against SiPh interposer networks tailored for ML acceleration: SPRINT [104] and SPACX [103]. Our analysis encompasses a range of neural network models, employing Tensorflow 2.13.0 alongside Qkeras for model representation. We evaluate six DNN models for the ImageNet dataset, including DenseNet121, EfficientNetB0, LeNet5, MobileNetV2, ResNet50, and VGG16. To ensure consistency, we adhere to a weight stationary dataflow paradigm, where weights remain within vector MAC registers, facilitating reuse across iterations [43]. Our power modeling aligns with the approach described in [116] for laser power estimation and assumed thermo-optic tuning [81] for trimming power considera-

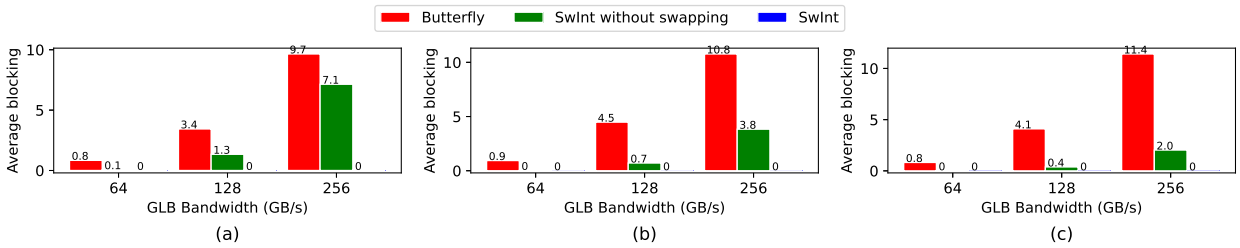



**Figure 6.13:** Analyse blocking instances of Butterfly in comparison with SwInt. (a) 4 chiplets, (b) 8 chiplets, and (c) 12 chiplets.

tions. We considered the power model and parameters used in [79] for Electrical-to-Optical (E-O) and Optical-to-Electrical (O-E) conversions. Our latency and energy modelings also align with the analysis principles of MAESTRO [117].

Our chiplet design closely resembles that of Simba [43], featuring sixteen MAC PEs per chiplet. However, in contrast to Simba's utilization of an electronic network-on-chip (NoC), we assume four gateways per chiplet. Four PEs are connected to a router and the router is connected to a gateway, facilitating communication through the SiPh interposer network.

Furthermore, we opt for SRAM technology with size 13 MiB for the Global Buffer (GLB) in line with [102]. Additionally, we set a maximum chiplet-interposer bandwidth of 100 GB/s per chiplet, a constraint determined by the microbump area [43]. For more details on our simulation setup and modeling parameters, please refer to the summary provided in Table 6.1. Please note that the parameters of the devices extracted from our evaluation in Section 6.4 are not re-reported in the Table.

### 6.5.2 Simulation Results

In Fig. 6.11, we analyze SwInt's power consumption in comparison to other SiPh interposer networks. SwInt demonstrates a significant improvement in power efficiency compared to SPRINT [104] and SPACX [103], primarily attributable to its remarkable reduction in laser power requirements. The lower laser power intensity requirement is due to the fact that the optical signal generated from the laser does not pass through many writers/readers. SPRINT relies on individual photonic links for each Reader, resulting in the incorporation of numerous MRR modulators. Con-

sequently, SPRINT consumes more power for trimming processes compared to SPACX, while SwInt maintains a small trimming power since the number of MRRs are minimized to match the switch bandwidth with the system bandwidth (see equation 6.5). As depicted in Fig. 6.11(b), in the broadcast state, bus-based architectures (e.g., SPRINT and SPACX) consume more laser power due to inefficient broadcasting. As discussed in Section 6.3.2, achieving broadcast in bus-based architectures requires precise tuning of filters to receive a portion of the signal and pass the rest to other receivers. However, accurate tuning of filters is affected by process variations of MRRs. Consequently, laser power must be increased to compensate for this variability, resulting in power inefficiency in broadcasting for bus-based architectures. In this analysis, we also considered the imbalanced signal division in the MZI divide state of SwInt. This resulted in a 0.5 dB loss for the worst-case imbalance between the two MZI outputs at each stage, as determined by our device-level analysis of the fabricated MMIs. The white portions in Fig. 6.11 represent power savings, where SwInt demonstrates significant power improvement of 63% in the unicast state and 88% in the broadcast state compared to the SPRINT architecture.

As shown in Fig. 6.12(a) SwInt introduces only a minor increase in latency when compared to SPRINT, primarily due to the switch reconfiguration delay. We use the first two letters of the DNN models in the figure. On average, the latency is 7.3% higher than SPRINT and 59% lower than SPACX. Also, we estimated the footprint of the SwInt switch, according to our MZS design, which is $0.8mm^2$ (13.3% of our chiplet size).

In Fig. 6.12(b), we present our energy efficiency results. On average, SwInt achieves an impressive 87.6% reduction in energy consumption compared to SPACX and a substantial 59.7% reduction compared to SPRINT. This significant energy efficiency can be attributed to SwInt's streamlined network architecture. SwInt significantly improves power consumption while introducing only a minimal latency, resulting in minimized energy consumption. Additionally, its efficient broadcast approach, combined with dynamic laser power management, further enhances energy efficiency. SwInt optimizes network topology ensuring that the interposer network's offered bandwidth aligns with the system's bandwidth requirements. Scalability analysis is also

presented in Fig. 6.12(c), demonstrating a substantial increase in energy efficiency for SwInt as the number of chiplets scales from 8 to 12. This finding underscores the scalability of SwInt's design for future large-scale accelerators.

In Fig. 6.13, we compare the blocking instances between Butterfly and SwInt. This analysis includes the average number of blocking instances with varying GLB chiplet bandwidth and the number of MAC chiplets. As can be observed, SwInt (which uses both input selection and input swapping) offers a significant improvement over Butterfly (no blocking in all cases), effectively removing blocking instances. In the SwInt without swapping scenario, where only input selection is used, blocking instances still occur, but they are notably reduced compared to Butterfly. SwInt without swapping is suitable when aiming to use online routing for switch configuration, reducing complexity. However, in our study, we focus on offline routing, where the swapping technique does not introduce any additional overhead to the configuration process.

## 6.6 SwInt with Bus Integration

### 6.6.1 Switch-based and bus-based architectures

In advocating for the adoption of a switch-based network architecture in the preceding sections, it is essential to acknowledge that the choice between switch-based and bus-based architectures is context-dependent. While our p roposal emphasizes the advantages of a switch-based network for efficient silicon photonic interposers, with increased Wavelength WDM degrees and a large number of MAC chiplets, it is prudent to recognize scenarios where a bus-based may offer more practical solutions.

In small-scale systems, the simplicity and lower overhead of bus-based communication system might outweigh the complexities associated with a switch-based alternative. The reduced hardware requirements and streamlined communication paths can make bus-based architectures a more cost-effective and straightforward solution for such small systems.

Similarly, in instances where the WDM degree is limited, and the overall network complexity is constrained, a bus-based architecture may be a viable choice. The simplicity of bus-based com-

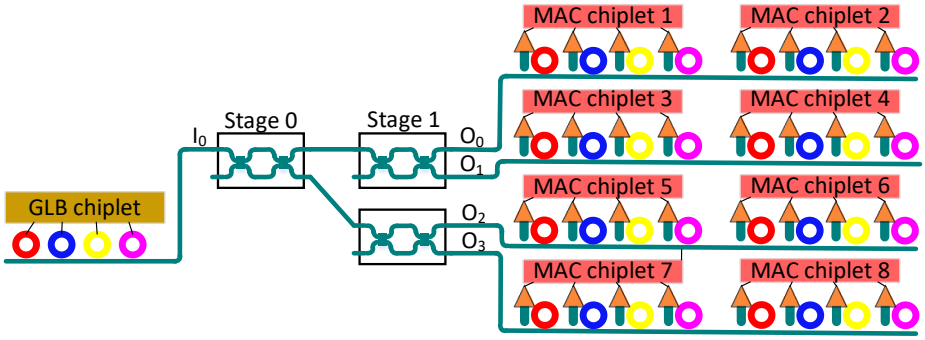



**Figure 6.14:** SwInt with Bus, with 2 readers at each output ($B = 2$).

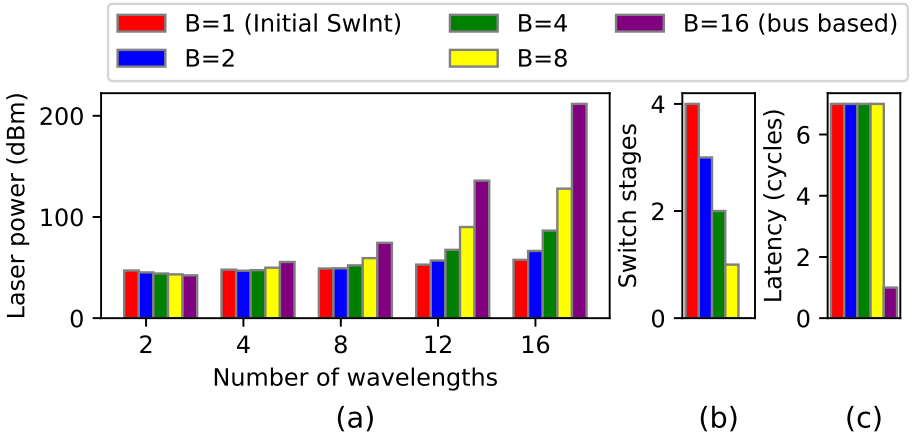



**Figure 6.15:** Exploring optimal $B$ (number of readers at each output of SwInt) in SwInt with Bus design under different number of wavelengths.

munication can be advantageous when dealing with scenarios where the benefits of a switch-based network may not be fully realized.

### 6.6.2 SwInt with Bus Integration: Design Trade-offs

As shown in Fig. 6.14, a hybrid architecture can be used for a small-scale network or with a limited WDM degree, where a bus is employed at each port of the switch. This strategic augmentation provides enhanced flexibility, allowing for multiple readers at the output of GLB to MAC switch or several writers at the input of the MAC to GLB switch. In this case, the number of switch stages is reduced. Considering $B$ as the number of readers on each switch output, the number of switch stages is

$$NS_{But_B} = \lceil \log_2 N/B \rceil \tag{6.8}$$

A smaller number of switch stages results in smaller switch loss (see Equation 6.3). However, since there are more filters on each output of the switch, losses on microrings increase, which increases laser power. Therefore, depending on the number of wavelengths and number of readers, the optimum architecture may require a different number of readers on each output of the switch. In Fig. 6.15(a), we compare laser power for different numbers of wavelengths with varying numbers of readers on each switch output. Since laser power dominates the total power of the system (see Fig. 6.11), minimizing it aids in reducing the overall power consumption. In this analysis, the total number of readers is sixteen. For more than four wavelengths, one reader at each output is the optimum architecture (initial SwInt design). However, in the case of four wavelengths, two readers on each output are optimum, while for the two-wavelength case, having sixteen readers on each output, i.e., removing the switch and having a bus architecture, is preferable. As a result, using a bus on switch outputs is efficient only when the number of wavelengths is small, whereas when the number of wavelengths exceeds four, a pure switch architecture is recommended.

However, reducing the number of readers on switch outputs increases switch size, which can potentially increase area overhead and reconfiguration latency. Therefore, in Fig. 6.15(b), we explore the number of switch stages under different numbers of readers on each switch output. As can be seen, increasing the number of readers on each output ($B$) decreases the number of switch stages. However, considering the latency and area overhead, the advantage of having a larger switch under a high-bandwidth network with a large number of wavelengths is beneficial. The reason is that, as mentioned in the last section, the switch size is small compared to the size of the interposer. Moreover, since the switch cells can be configured in parallel, the switch size does not significantly affect the reconfiguration time. We further evaluated the configuration controller of TRINE's switches by implementing it in Verilog and analyzing the area using Cadence Genus under 15 nm technology. Our analysis indicates that for all the switch sizes shown in the figure, the configuration decision can be made in a single cycle at 2 GHz frequency. Therefore, as demonstrated in Fig. 6.15(c), the switch does not impose a high latency compared to its power efficiency benefits.

## 6.7 Conclusion

This chapter proposed an innovative SiPh interposer network architecture for large-scale ML accelerators. We developed an energy efficient Butterfly-based interposer network topology to minimize switch stages while preventing blocking. Our architecture improved energy efficiency by 59.7% on average, by utilizing a low loss switch with an efficient broadcast technique. SwInt's MZI-based switch also significantly reduces laser power consumption. We introduced a novel "divide" state for our fabricated MZIs alongside existing cross and bar states to facilitate our energy-efficient broadcast communication designed to minimize laser power usage. Additionally, we employ active splitters and dynamic laser power management to tune laser power for both unicast and broadcast modes, effectively reducing energy consumption. These innovations offer a significant step forward in addressing the growing demand for efficient ML accelerators, emphasizing the importance of interposer communication in chiplet-based ML accelerators.

# Chapter 7

# SCRIPT: A Multi-Objective Routing Framework for Securing Chiplet Systems against Distributed DoS Attacks

## 7.1 Introduction

The chiplet system architecture is becoming a cost-effective solution for building large-scale systems. Using off-the-shelf chiplets for modular integration significantly reduces design time and costs, as these foundational building blocks can be procured from third-party entities. However, as the employment of 2.5D chiplet systems grows further, *unique and new security challenges arise* [118]. For example, when handling chiplets from potentially untrusted sources, the security of Network-on-Interposer (NoI) communication is at risk. The vulnerability arises as malicious chiplets may focus traffic on the NoI or Vertical Links (VL), jeopardizing the interposer's service availability. This situation leads to Denial-of-Service (DoS) attacks.

Detecting and preventing DoS attacks originating from single or multiple sources (IPs) within a chip has been addressed in several prior works [119, 120], but the challenges *differ* when various *chiplets* coordinate to execute a Distributed Denial-of-Service (DDoS) attack [121] at the NoI level. Moreover, the traditional methods of avoiding DoS attacks through routing obfuscation at the IP level may inadvertently degrade chiplet-based network performance.

The intrinsic modularity of chiplet systems renders them more susceptible to DDoS attacks. Notably, the NoI is shared among multiple chiplets, a departure from traditional monolithic chips. The availability of the NoI is pivotal, as the entire system relies on its functionality for inter-chiplet communication. Furthermore, VLs represent a costly and inherently critical network path. As a result, attackers consider them particularly valuable. For instance, an attack scenario is illustrated in Fig. 7.1 where multiple (two in this example) untrusted chiplets maliciously transmit packets

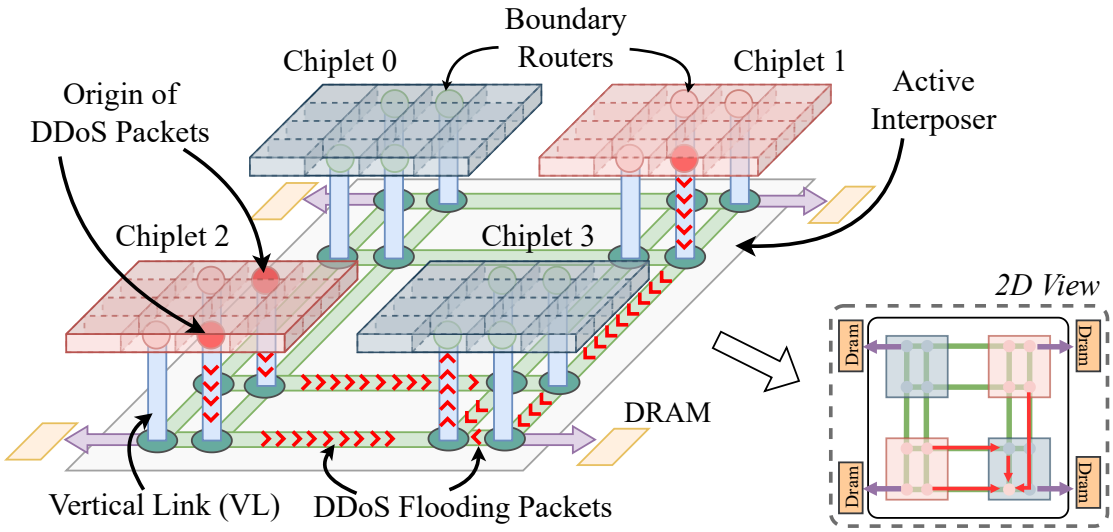



**Figure 7.1:** High-level overview of a 2.5D network with 4 chiplets, subdivided into 16 IPs. Each chiplet contains four boundary routers interface with the interposer. Chiplet 1 & 2 are malicious and transmit DDoS flood packets to Chiplet 3.

to a destination chiplet, seeking to concentrate the traffic load on a specific VL in an attempt to induce a DDoS condition.

This chapter presents a novel routing-based framework, called SCRIPT, developed to mitigate DDoS attacks in 2.5D chiplet systems by intentionally obscuring the predictable paths typically exploited by attackers. In contrast to conventional routing techniques vulnerable to targeted congestion, our proposed solution incorporates elements of obfuscation and randomization, thereby disrupting the predictability underlying orchestrated attacks. Within this new framework, the network dynamically generates randomized paths, confounding potential adversaries and enhancing the network resilience against coordinated traffic designed to compromise NoI service availability. Furthermore, our framework considers the trust levels of chiplets as a measure for quantifying the threat. It introduces a multi-objective optimization to improve the network's reliability against DDoS attacks, while concurrently minimizing performance degradation. The main contributions of this chapter are summarized below:

- We propose obfuscated routing on the NoI to ensure system security, even in the presence of malicious chiplets.

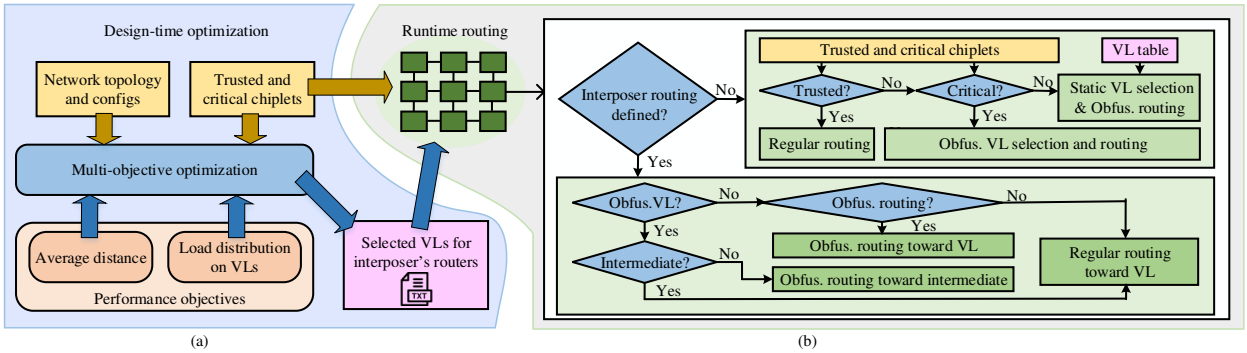



**Figure 7.2:** SCRIPT framework: (a) Design-time optimization takes into account the trust level and criticality of chiplets, along with performance objectives, to propose a list of Vertical Links (VLs) for use in the runtime routing. (b) In routing, a VL is selected, and the routing mode (obfuscated or regular) is determined to deliver the packet to the specified VL.

- We mitigate DDoS attacks on VLs originating from one or multiple malicious chiplets in a distributed manner.
- We present multi-objective optimization for source-destination average-distance and load distribution on VLs to avoid performance penalty due to DDoS attacks while achieving security.

The organization of this chapter is structured as follows: In the upcoming section, we delve deeper into the background and related work on NoC's security. Following that, in Section 7.3, we discuss our threat model. In Section 7.4, we present our proposed performance- and security-aware interposer. Section 7.4 also involves an exploration of the design space for performance and security. Our simulation results are showcased in Section 7.5, and finally, in Section 7.6, we draw conclusions and highlight the significance of our contributions.

## 7.2 Background and Related Work

A significant challenge in chiplet systems lies in designing the NoI to provide reliable and low-latency inter-chiplet communication. While routing algorithms have been proposed to achieve low hardware costs [52, 54] and high flexibility in VL selection [23], they often neglect the security challenges inherent in chiplet systems. Previous work on security has primarily focused on

designing routing algorithms for conventional Networks-on-Chip (NoCs) [122, 123] but has not specifically addressed the unique requirements of NoIs in chiplet systems. This lack of specificity renders them inefficient when applied to NoI.

A major threat to NoCs is DoS attacks where the aim is to block access to network services by overwhelming the system. These attacks involve malicious IPs flooding the NoC with packets, impacting the availability of on-chip resources [121]. For instance, memory-intensive applications can lead to heavy traffic on routers connected to memory controllers, and if a malicious IP targets the same node, it results in significant degradation of NoC performance. The flooding can cause congestion in routers, leading to severe delays in responses. Various works, including one with Hardware Trojans embedded in routers, highlighted the performance degradation caused by flooding the network with additional packets [124].

Furthermore, DoS attacks can be carried out through packet corruption [125, 126] and traffic flow manipulation [127]. Regardless of the attack type, these efforts either explore methods of employing such attacks, their implications, or ways of *detecting* attacks. [122] proposed a zone-based routing algorithm to separate secure regions from insecure ones, enhancing defense against DoS and timing attacks. However, the zone-based routing approach, despite its complexity, constrains resource utilization and the flexibility of the network particularly when dealing with dynamic workloads.

Related work primarily focuses on DoS attacks. However, in the case of DDoS attacks originating from multiple chiplets, the detection mechanisms may prove ineffective without profiling application behavior as a priori [121], particularly when dealing with small packet injection rates that go undetected. For instance, in a recent study [119], the minimum flit injection rate assumed for attacks is 0.3. In contrast, we will demonstrate that a DDoS can occur in a chiplet system with an injection rate $10\times$ smaller than that.

## 7.3 Threat Model and Assumptions

Our threat model assumes the following: similar to Fig. 7.1, we assume a system with multiple chiplets connected to routers within an NoI. Each chiplet is composed of multiple IPs/cores (e.g., sixteen in Fig. 7.1), each with its own router that is part of an NoC, whereas only a few (e.g., four in Fig. 7.1) of these routers serve as *boundary routers* which connect to the interposer's routers (NoI) via VL.

We assume that there can be multiple malicious chiplets (e.g., two in Fig. 7.1). Furthermore, our model assumes the NoI is *trusted*. It is also assumed the attacker of a compromised chiplet has full control over its routers, with the ability to send as many packets as desired. However, the attacker *does not* possess full knowledge of the underlying router architecture on the NoI. Furthermore, we assume that various chiplets in the system have different *criticality* and *trust levels*. Specifically, we assume some chiplets are critical hence their availability should be protected. Further, we define trust level as whether or not a chiplet can be trusted. For example, from NoI's perspective, chiplets from vendor A are trusted while chiplets from vendor B are not (i.e., due to supply chain issues).

In a DoS scenario, a malicious chiplet can inundate a critical chiplet with a high volume of traffic, jeopardizing its connectivity to the NoI. For a DDoS attack, *multiple* malicious chiplets collaborate to coordinate an attack. In both cases, the goal is to concentrate a load on a specific part of the NoI or a VL associated with a critical chiplet. Detecting DDoS attacks poses significant challenges as a low injection rate from multiple chiplets can collectively generate a high volume of traffic, complicating the identification of malicious activity. Additionally, the collaboration of two or more malicious chiplets can cause congestion by reciprocally forwarding traffic, leading to congestion on a specific path of the NoI.

## 7.4 SCRIPT: Proposed Performance- and Security-aware Routing Framework

### 7.4.1 Overview

The SCRIPT framework, shown in Fig. 7.2, introduces a comprehensive approach to enhance the security and performance of 2.5D chiplet systems. This framework includes two key components: (a) Design-time optimization, and (b) Secure and performance-preserved routing. The design-time optimization is a pivotal aspect of SCRIPT, where the trust level and criticality of individual chiplets are taken into account. This optimization process aims to strike a balance between enhancing security in the runtime routing and ensuring that it does not introduce substantial performance overhead. Additionally, performance objectives are taken into account during this phase. The outcome is the generation of an optimized set of VLs, which serves as the regular VL selection for the subsequent runtime routing of the network. The runtime routing process in SCRIPT involves the dynamic selection of VLs based on the optimized list generated during design time and a random VL selection for obfuscation. Furthermore, it offers the flexibility to choose between obfuscated and regular routing. The obfuscated routing mode adds an additional layer of security by intentionally introducing complexity into the routing path, contributing to the framework's overall resilience against potential security threats.

### 7.4.2 SCRIPT Routing Process

In chiplet systems, the routing algorithm utilizes boundary routers as intermediate routers to deliver packets from one chiplet to another. A boundary router is specifically defined as a router on a chiplet that is directly connected to the NoI through a VL. The routing process for sending a packet from a source chiplet to a destination chiplet involves five distinct steps: (1) the packet is routed from the source router to a boundary router on the source chiplet; (2) vertical routing occurs as the packet moves from the boundary router on the source chiplet to the NoI; (3) the packet is routed across the NoI to the VL connected to the destination chiplet; (4) vertical routing occurs

from the NoI to the boundary router on the destination chiplet; and (5) the packet is routed from the boundary router to its ultimate destination within the destination chiplet. Our primary focus is on steps 3 and 4, where attacker chiplets can potentially concentrate traffic on a specific path on the NoI or a VL to initiate an attack. Note that attack mitigation within a chiplet (i.e., IP-level DoS attacks) is beyond the scope of this chapter, as there is substantial existing research on this topic.

In step 3, concerning intra-interposer routing, we aim to implement obfuscated routing for untrusted packets. A comprehensive explanation of our obfuscated routing approach is provided in Section 7.4.5. For the VL selection employed in step 4 of the routing process, we enhance network security against DDoS by improving VL selection, as detailed in Section 7.4.3. Nevertheless, this security enhancement can have a significant impact on network performance. Therefore, we introduce a multi-objective optimization strategy, discussed in Section 7.4.4. Please note that the first VL selection on the source chiplet (i.e., step 2) is beyond the scope of this chapter and we consider the source VL-selection strategy used in prior work (e.g., see DeFT [23]) in our experiments (see Section 7.5).

### 7.4.3 Enhancing Security in VLs

To enhance the security of 2.5D chiplet systems, it is crucial to address the risk of attacks on VLs that connect critical chiplets to NoI. VLs (i.e., microbumps) are limited resources compared to metal links on chips, making them susceptible to DDoS attacks due to their scarcity and relatively smaller bandwidth. The combination of their limited availability and bandwidth amplifies their vulnerability to congestion, which malicious chiplets can exploit—i.e., to deliberately congest a specific path or VL, triggering a DDoS attack. Therefore, one of the key objectives in SCRIPT is to distribute the load across various VLs of chiplets to prevent congestion.

Note that we assume the NoI is trusted, thus we maintain control over the routing of the NoI. Hence, the main feature of our routing algorithm is the strategic NoI routing. The selection of VLs (on the destination chiplet) as part of the NoI routing process is crucial in this context. An attacker might attempt to send multiple packets from different sources to a victim chiplet, inducing

congestion on a specific VL at the victim chiplet's input. Current routing algorithms for 2.5D chiplet systems often employ a static VL selection [23, 52, 54], making them susceptible to attacks where the attacker sends packets to the destinations using the same VL, leading to congestion. For instance, an attacker could strategically distribute packets from multiple sources of untrusted chiplets to multiple destinations near a specific VL, causing congestion on that VL. To counter this, we randomly select destination VLs for packets originating from an untrusted source chiplet and destined for a critical chiplet. However, it is important to note that this random selection has a notable impact on network performance.

We assess the average distance overhead introduced by this approach in Fig. 7.3. Here we assume a four-chiplet system under uniform traffic, where each chiplet is a $4 \times 4$ mesh. As depicted, the average distance increases with the growing number of untrusted chiplets. A larger average distance necessitates packets to traverse a greater number of routers, leading to increased latency and energy consumption in low-traffic scenarios. Also, it raises the risk of congestion. Consequently, we opt for dynamic random destination VL selection exclusively for packets originating from untrusted chiplet sources and destined to critical chiplets. We term these packets "VL obfuscated packets (VOP)" due to their routing in an obfuscated manner on the NoI, in addition to their random VL selection.

Given that the random VL selection for the VOP packet is contingent on the trust level of the source chiplet, we define $T_c$ as the trust factor of chiplet $c$, with a trust factor of 0 denoting that the chiplet is completely untrusted, and a trust factor of 1 indicating full trust in the chiplet. In our framework, we utilize this trust factor to prioritize performance enhancement for trusted chiplets, while focusing on improving routing to enhance security against DDoS attacks for untrusted chiplets. Similarly, we define $Cr_c$ as the criticality level of a chiplet as a victim chiplet of DDoS attacks. Therefore, we label a packet as an VOP packet given that the packet generated from

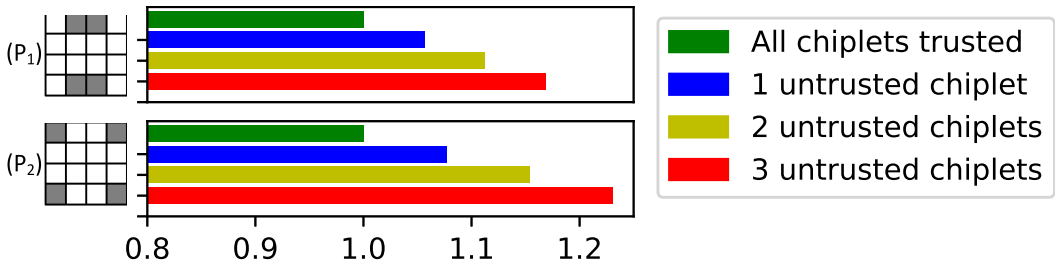



**Figure 7.3:** Normalized average distance overhead with two different patterns of chiplet-connected VLs ($P_1$ and $P_2$).

source $s$ and forwarded to destination $d$ with probability:

$$Ov_d^s = \begin{cases} T_{c_s}, & \text{if } Cr_{c_d} = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{7.1}$$

Here, $T_{c_s}$ is the trust factor of the source chiplet and $Cr_{c_d}$ is the criticality level of the destination chiplet.

To alleviate the impact on system performance under SCRIPT's secure VL selection, we conduct design-time optimization for static VL selection (predefined VLs selected during design time), which is employed by trusted packets. These optimizations are geared towards enhancing performance while taking into account the considerations of our secure VL selection methodology.

### 7.4.4 Multi-Objective Optimization

As previously mentioned, in addition to the dynamic random VL selection for VOP packets, we employ a static VL selection for regular packets. An optimized set of VLs is pre-stored in routers for runtime usage (cost is analyzed in Section 5.4). Note that static VL selection is only used for regular packets, and not for VOP packets that are from untrusted sources. This section delves into the optimization of the selected VL set, with a specific emphasis on enhancing both the security and performance aspects of the network.

While random online selection is suitable for obfuscation purposes, it is inefficient for performance due to the necessity for the NoI's router to search among available VLs and make real-time

selections. To address this, we opt for offline selection for regular packets, a method commonly employed in state-of-the-art routing algorithms [23, 52, 54], as described below.

Let $NC$ represent the total number of chiplets, $N_c$ represent the number of nodes in chiplet $c$, and $NV_c$ represents the number of VLs in chiplet $c$. We also define, $L_{d_v}$ as the load on vertical link $v$ connected to destination chiplet $d$. We calculate $L_v$ based on the load generated by trusted chiplets. Note that the load generated by untrusted chiplets is already distributed among VLs since an online random selection is used. Therefore, the load on vertical link $v$ based on the trust level of chiplets is:

$$L_{d_v} = \sum_{c=1}^{NC} \sum_{i=1}^{N_c} \sum_{j=1}^{N_d} W_{(i,j)} \times (1 - Ov_j^i) \times U_{v_{(i,j)}}. \tag{7.2}$$

In this equation, $W_{(i,j)}$ denotes the weight of communication between node $i$ and $j$. $Ov_j^i$, which we introduced in (7.1), is based on the trust factor of the source chiplet and the criticality factor of the destination chiplet. We use this term to exclude the VOP packets from the load, as we have already distributed the load coming from untrusted chiplets on VLs. The variable $U_{v_{(i,j)}}$ signifies whether communication between node $i$ and node $j$ utilizes vertical link $v$ or not. It is defined as:

$$U_{v_{(i,j)}} = \begin{cases} 1, & \text{if } v \text{ is used to send a packet from } i \text{ to } j \\ 0, & \text{otherwise.} \end{cases} \tag{7.3}$$

We use utilization variance of $L_{d_v}$ in our design-time optimization to mitigate potential congestion on $s$ [40, 128, 129]. This objective enhances potential congestion for the network's performance and also secures the network against DDoS attacks. We define utilization variance on the VLs of chiplet $d$ as:

$$UtVar_d = \frac{1}{NV_d} \sum_{v=1}^{NV_d} (L_v - \mu_d)^2, \tag{7.4}$$

where $\mu_d$ denotes the average load on all VLs of chiplet $d$:

$$\mu_d = \frac{1}{NV_d} \sum_{v=1}^{NV_d} L_v. \tag{7.5}$$

Now that we have utilization variance for every chiplet, we want to calculate to total utilization variance on VLs as the objective in our optimization. Thus:

$$UtObj = \sum_{d=1}^{NC} UtVar_d. \tag{7.6}$$

The number of hop counts between the source and destination significantly impacts the network performance. Large hop counts create more traffic and worsen congestion. It also increases the latency of each packet and results in higher energy consumption during packet transmission. To address this issue, we incorporate *weighted distance* as one of our performance objectives during optimization. Weighted distance is defined as the sum of source-destination distances weighted by the average communication frequency (i.e., communication weight) between the source and destination. We define the weighted distance objective as:

$$DisObj = \sum_{c=1}^{NC} \sum_{i=1}^{N_c} \sum_{j=1}^{N_d} W_{(i,j)} d_{(i,j)} \times IC_{(i,j)} \times (1 - Ov_j^i), \tag{7.7}$$

where $d_{(i,j)}$ is the source-destination hop count between node $i$ and $j$. We use $IC_{(i,j)}$ to exclude source-destination pairs on the same chiplet since they are not affected by our VL selection optimization. Therefore, $IC_{(i,j)} = 0$ indicates that routers $i$ and $j$ are on the same chiplet, while $IC_{(i,j)} = 1$ indicates that routers $i$ and $j$ are on different chiplets. Similar to the utilization variance objective, we use the term $(1 - Ov_j^i)$ to exclude VOP packets because they are not affected by static VL optimization. We use $DisObj$ and $UtObj$ in our optimization of static VL selection. We employ a simulated-annealing-based multi-objective optimization algorithm [26] for the optimization. More details of our optimization and solution selection process are presented in Section 7.5.2.

### 7.4.5 Obfuscated Deadlock-Free Routing

In addition to dynamic random VL selection, we employ obfuscated routing on the NoI for "route obfuscated packets (ROP)" with probability $OR_d^s = T_{c_s}$. To enhance unpredictability and

**Algorithm 8** SCRIPT virtual channel selection

```
if Reset then
    Round_Robin ← 0
if current is the first router on NoI then
    if Packet is ROP then
        Virtual channel ← 0
    else
        if Round_Robin is TRUE then
            Virtual channel ← 1
            Round_Robin ← FALSE
        else
            Virtual channel ← 0
            Round_Robin ← TRUE
else if current is the intermediate and packet is ROP then
    Virtual channel ← 1
```

deter potential attacks, we randomly select an intermediate router for ROP packets. However, applying obfuscated routing to all packets from untrusted chiplets is not performance-efficient. Therefore, we implement obfuscated routing for untrusted packets if their path traverses a *critical region* of the NoI. We define the critical region of the NoI as the area where routers are directly connected to a critical chiplet.

We implement our obfuscated routing using the virtual channel (VC) concept [23,53] but without any extra VCs compared to VCs already used for deadlock prevention in chiplet systems. VCs are employed to separate traffic flows in virtual networks, preventing cyclic dependencies and avoiding deadlock. Employing the deadlock-freedom solution in the DeFT routing algorithm [23], packets are switched to the second VC across the NoI. This implies that VC switching is permitted when a packet enters from the source chiplet into the NoI and should occur before the packet exits the NoI. We leverage this opportunity to develop an obfuscated routing algorithm for the NoI. Packets for which we intend to obfuscate (i.e., ROP packets) are initially routed to an intermediate router using the first VC, and from there, they are directed to the VL using the second VC. To guarantee livelock freedom, a 1-bit flag is employed in the header flit. This flag marks packets that have already been routed to an intermediate router, ensuring that the packet will be routed minimally to the VL after visiting the intermediate router.

For regular packets, we utilize a round-robin approach at the first router on the NoI, evenly distributing the load across VCs. As a result, 50% of the packets undergo switching upon entering the NoI, while the remaining 50% undergo switching upon exiting the NoI. However, for ROP packets at the first node in the NoI, the first VC is employed, and the packet switches to the second VC upon reaching the intermediate router. The details of the VC-selection process are illustrated in Algorithm 8. Note that following deadlock freedom rules in DeFT, the packets are initially routed in the first VC from a source chiplet.

Algorithm 9 also outlines our routing approach on the NoI. In this algorithm, we utilize West-first routing (adaptive for routing to the east direction) in the first VC and East-first routing (adaptive for routing to the west direction) in the second VC. West-first routing is introduced in [60], and East-first routing is created by exchanging west and east directions in West-first routing. The reason for employing partially adaptive routing in two opposite directions is that each routing strategy provides a better load distribution in a specific direction. Choosing two opposite directions allows us to achieve a higher overall load distribution. Moreover, these partially adaptive routing strategies not only contribute to traffic distribution but also enhance network security. This is because it becomes less predictable for potential attackers to determine the exact path for a routing scenario.

**Algorithm 9** SCRIPT routing on interposer

```
if packet arrived at destination vertical link then
    Route vertically to up
else if packet is ROP then
    if the intermediate is not visited then
        Route West-first toward intermediate
    else
        Route East-first toward the vertical link
else if packet is regular then
    if Virtual channel is 0 then
        Route West-first toward the vertical link
    else
        Route East-first toward the vertical link
```

## 7.5 Evaluation and simulation results

### 7.5.1 Simulation Setup

We extend the capabilities of the Noxim simulator [75] to accommodate chiplet systems. The simulation framework is augmented to facilitate a comparative analysis with DeFT [23] and MTR [52] routing strategies. Although DeFT and MTR are not specifically proposed for security purposes, we compare with these well-known routing approaches as there is currently no other work proposing a routing algorithm to enhance the security of chiplet systems. For the simulation, we adopt the same configuration parameters as employed in DeFT and MTR for a consistent evaluation. The chiplet system consists of four chiplets, each with a $4 \times 4$ mesh network configuration. Similarly, the NoI is modeled as a $4 \times 4$ mesh network. The interconnection between the chiplets and the NoI involves four routers per chiplet, designated as boundary routers, which are linked to the NoI. The arrangement of boundary routers is the same as the pattern used in DeFT and MTR, represented also as $P_1$ in Fig. 7.3. Each router is equipped with buffers at each port with space for 4 flits, 2 virtual channels, and a 128-bit link and flit width. This configuration ensures a comparable performance evaluation of SCRIPT to DeFT and MTR routing algorithms. We also employ partially adaptive routing (West-first and East-first) on NoI for DeFT and MTR to have a fair comparison.

We incorporate real application traffic by utilizing a set of widely recognized PARSEC applications [42]: *facesim, bodytrack, fluidanimate, streamcluster, swaptions, dedup, and canneal*. To emulate these benchmarks, we employ Gem5 [38] in full-system simulation mode. The simulated system architecture comprises 64 x86 cores, each equipped with a private L1 cache. Additionally, the system incorporates four L2 cache banks to manage shared resources efficiently. This configuration reflects a realistic representation of a multi-core processing environment, allowing us to evaluate the performance and behavior of the proposed enhancements under conditions that closely resemble practical scenarios.

### 7.5.2 Multi-Objective Optimization Results

In our design-space exploration, we implemented the Simulated Annealing based Multi-Objective Optimization Algorithm (AMOSA) [26] to optimize the performance of our NoI. As detailed in Section 7.4.4, our optimization considers two primary objectives related to performance, with additional considerations for the trust and criticality levels of chiplets. The Pareto front generated by AMOSA, shown in Fig. 7.4.a, showcases various solutions, each representing a trade-off between optimized weighted distance and utilization variance on VLs. The input to this optimization is the average traffic over all the PARSEC applications. From this Pareto front, we selected several solutions for further evaluation and simulated them using our enhanced Noxim simulator. Fig. 7.4.b illustrates the average network latency comparison for these selected solutions.

Solution 1, in particular, has been selected due to its minimized average latency. It is worth noting that the optimal solution can vary depending on the nature of the traffic. In scenarios with low congestion, solutions emphasizing smaller average distances are more efficient. Conversely, in highly congested situations, a solution that minimizes utilization variance proves beneficial.

It is important to highlight that our optimization approach does not tailor the traffic to each application individually; rather, we adopt an average over all the traffic types as our optimization input, making a pessimistic assumption. However, considering the traffic of each application in the design-time optimization could lead to further improvements, but might not be realistic.

Our primary contribution lies in the flexibility of our proposed framework, which is not tied to any specific multi-objective optimization method. Our framework can be implemented with any multi-objective optimization technique without loss of generality. We opted for AMOSA due to its simplicity and efficiency. While it is acknowledged that simulated annealing-based search may be slower compared to other optimization techniques, the optimization process in our case is conducted offline, and we simplify objectives using our formalization, mitigating concerns about speed. Indeed, in our evaluation, the optimization was completed in less than an hour. In our implementation of AMOSA, we utilized specific parameters to optimize the exploration of solution space. We initialized the temperature at a relatively high value of $1.0e5$ degrees, allowing

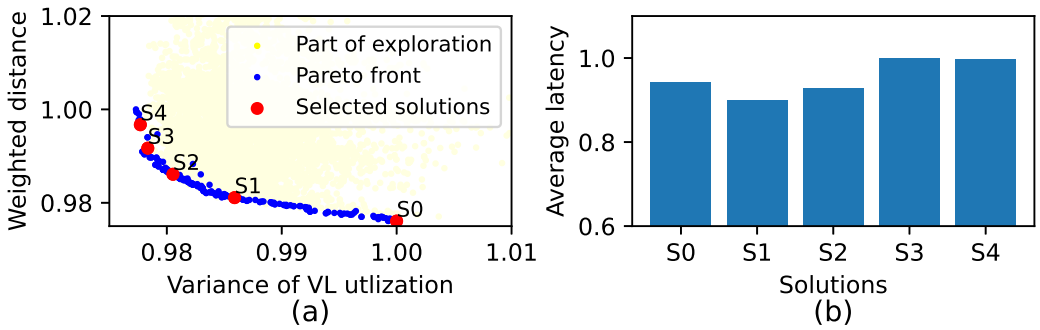



**Figure 7.4:** Multi-objective optimization and solution selection process. (a) Pareto front under weighted distance and utilization variance objectives. Five solutions on the Pareto front are selected for evaluation. (b) Latency results of the simulation under the selected solutions.

for a wide exploration range in the early stages of the algorithm. This high starting temperature encourages acceptance of uphill moves, which helps in escaping local optima. To ensure convergence towards an optimal solution, we set the stopping temperature to a significantly low value of $1.0e-5$ degrees. This stringent criterion enables the algorithm to focus on exploiting promising regions as it approaches convergence. Moreover, we incorporated an annealing schedule with a cooling factor (alpha) of 0.95. This factor determines the rate at which the temperature decreases, balancing between exploration and exploitation throughout the iterations. By gradually reducing the temperature, we control the balance between exploration and exploitation, promoting deeper exploration initially while fine-tuning towards optimal solutions later in the process. To manage computational resources efficiently, we conducted a fixed number of iterations set at 100, ensuring a reasonable trade-off between solution quality and computational cost.

### 7.5.3 Security and Latency Analysis

We assess the average latency under real application scenarios without any attacks, as depicted in Fig.7.5.a. For this evaluation, we assume one out of the four chiplets is untrusted, while another is critical. SCRIPT introduces a minor latency overhead by employing obfuscated routing and VL selection for certain packets, enhancing security against DDoS attacks. This overhead is slightly higher in high-load applications such as *streamcluster*, *dedup*, and *canneal*. On average, SCRIPT exhibits a 14% increase in the average latency.

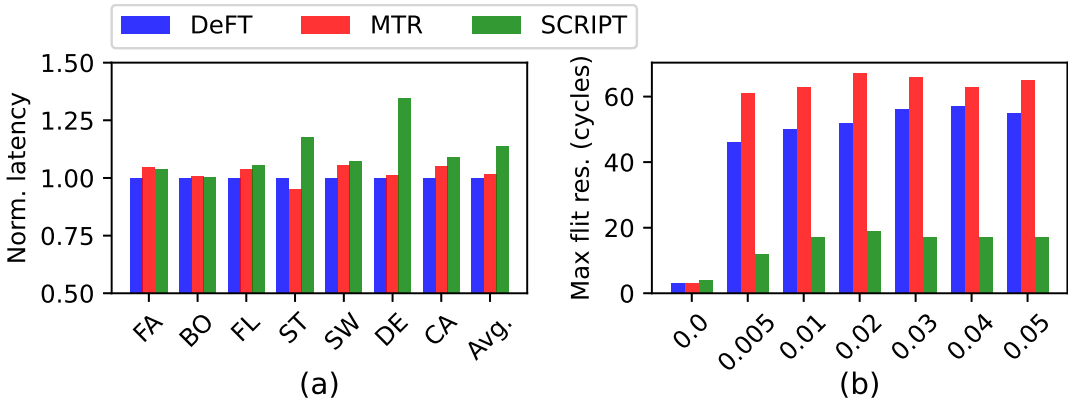



**Figure 7.5:** (a) Normalized average latency under real application scenarios without attacks. X-axis shows the initial two letters of each application. (b) Maximum average flit residency in a router among all routers of NoI under different attack scenarios (results are averaged among all applications).

To evaluate the reliability of SCRIPT under DDoS attacks, we consider the maximum average flit residency in a router across all NoI's routers under various attack scenarios, shown in Fig.7.5.b. This assessment is conducted across all real applications. Flit residency refers to the average time a flit remains in a router, experiencing blocking. A prolonged waiting time results in a DoS, as packets cannot be delivered in real-time. In this experiment, we investigate the impact of four malicious nodes attacking the critical chiplet with varying rates of malicious flit injection (flits/cycle/node). Even with low injection rates from malicious nodes, the effects are significant on DeFT and MTR, due to the distributed nature of the attack. This highlights the potential benefits of SCRIPT as a security-aware routing solution. Traditional attack detection approaches may struggle to identify attacks with low injection rates. For instance, in [119], the minimum assumed attack injection rate is 0.3. However, we show that under an injection rate of, for example, 10$\times$ smaller (i.e., 0.03 in Fig. 7.5.b), a flit residency of 56 cycles is imposed with an insecure routing like DeFT. Therefore, our SCRIPT framework can be used in conjunction with an attack detection mechanism to handle undetected DDoS attacks. SCRIPT effectively handles such attacks and reduces flit residency by at least 64% in the given attack scenarios.

We further assess the performance under an increased number of untrusted chiplets, as illustrated in Fig. 7.6 (1/4 means one chiplet out of four chiplets). Assuming half of the chiplets are

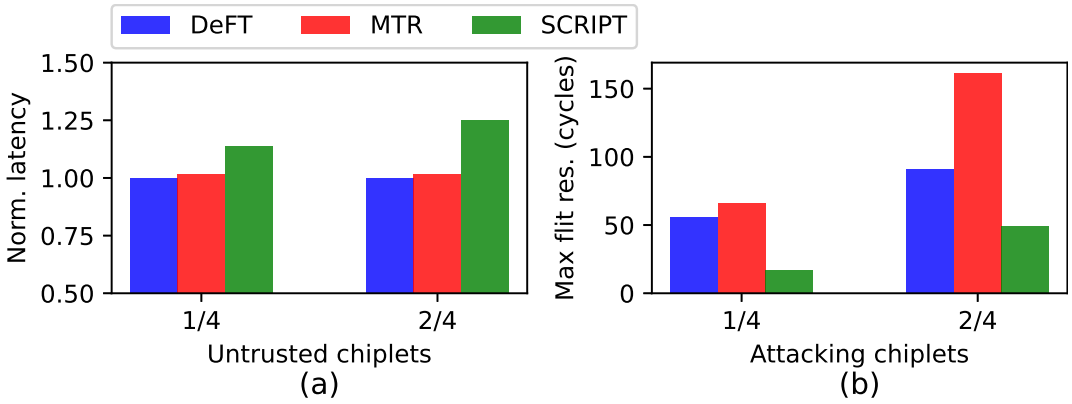



**Figure 7.6:** Impact of an increased number of untrusted and attacker chiplets on (a) average latency, and (b) flit residency.

untrusted, SCRIPT incurs a latency overhead of at most 26%, while concurrently reducing flit residency by at least 46%.

### 7.5.4 Area and Power Analysis

We implement the SCRIPT routing using Cadence Genus [39] under 15-nm technology node, considering a 128-bit flit width, 1V voltage, 2GHz frequency, and a 5-port router configuration. The results of the area and power analysis are summarized in Table 7.1, revealing negligible overhead in both aspects. Compared to DeFT, the area and power overhead of SCRIPT are 1.5% and 0.8%, respectively.

Compared to MTR and DeFT, our approach needs only four VL addresses per chiplet, facilitating a round-robin (RR) selection for obfuscated VLs without significant impact. Moreover, to optimize the selection of random intermediates and minimize overhead, we adopt a RR approach. Eight random routers are statically selected and stored in the router, facilitating a RR intermediate selection process. This strategy ensures the robustness of SCRIPT while keeping area and power overhead at a minimal level. Similarly, we used RR with the precision of 3 bits (8 levels) for implementing the probability of obfuscation (see Equ. 7.1). Also, SCRIPT uses 4 bits of header flit to store the address of an intermediate destination, which is negligible compared to 128-bit flit width.

**Table 7.1:** Area and power: SCRIPT vs. DeFT and MTR

| Routing | DeFT | MTR | SCRIPT |
|---|---|---|---|
| Router area ($\mu m^2$) | 9635.7 | 9496.4 | 9782.7 |
| Router power (mW) | 9.47 | 9.44 | 9.48 |

## 7.6 Conclusion

This chapter introduced SCRIPT, a routing framework tailored to improve reliability and performance against DDoS attacks in 2.5D chiplet systems. By using obfuscation routing techniques on the NoI, SCRIPT addressed security challenges associated with modular integration. Considering chiplet trust levels and implementing a multi-objective optimization, SCRIPT enhanced network resilience against both single-source DoS attacks and DDoS scenarios. Our new framework enhances the security of NoI by at least 64% under DDoS attacks, while minimizing performance degradation.

# Chapter 8

# Summary and future work

## 8.1 Summary

This dissertation has presented several contributions towards improving the performance, fault tolerance, energy efficiency, and security of emerging 2.5D and 3D NoCs.

Chapter 2 presents a lightweight adaptive congestion- and energy-aware elevator-selection algorithm, called AdEle+, for partially connected 3D NoCs. AdEle+ addresses traffic congestion on elevators while delivering packets with less hop counts in low traffic circumstances. By employing a set of elevators instead of one elevator for each source router, AdEle+ is able to adapt to runtime traffic loads and select the best elevator during runtime. Results show that AdEle+ improves the network latency in PC-NoCs and prevents network hot-spots, thus helping in designing low-latency and energy-efficient networks for high performance 3D manycore systems.

Chapter 3 presents ReD, which is the first deadlock-free and fault-tolerant routing algorithm for 2.5D chiplet systems. ReD offers VC-based deadlock-freedom and efficiently balances VC utilization. Freedom in VL-selection allows ReD to tolerate any pattern of VL faults. Compared to the state-of-the-art routing algorithms, simulation results show that ReD improves network reachability by up to 75%, reduces the network latency by up to 40% for multi-application execution scenarios, and minimizes energy consumption by up to 10.7% with less than 2% area overhead. These results highlight the promise of ReD for improving emerging chiplet systems.

Chapter 4 presents ReSiPI, which is a PCM-based reconfigurable silicon-photonic interposer network architecture for improving energy-efficiency in 2.5D chiplet systems. ReSiPI monitors the traffic load on the interposer and dynamically activates/deactivates gateways in the network. ReSiPI's controller intelligently manages this latency-power trade-off and, therefore, can achieve 53% improvement in network energy, in comparison with the best state-of-the-art 2.5D photonic

network. Results with real application traffic indicate that ReSiPI is a promising solution for an energy-efficient interposer network in emerging 2.5D chiplet platforms.

Chapter 5, extends ReSiPI architecture and presents a 2.5D chiplet platform-based photonic DNN accelerator where both communication on the interposer and computation on the chiplets employ silicon photonics. Compared to a monolithic photonic accelerator, a chiplet-based one not only improves fabrication yield and cost, but also reduces latency using a high-bandwidth photonic network on the interposer.

Chapter 6 introduces a novel silicon photonic interposer network architecture tailored for large-scale ML accelerators, introducing an energy-efficient Butterfly-based topology to streamline switch stages and eliminate blocking. By incorporating a low-loss switch with an optimized broadcast technique, our design achieves an average 59.7% improvement in energy efficiency. Furthermore, our MZI-based switch within SwInt substantially reduces laser power consumption, featuring a newly introduced "divide" state alongside traditional cross and bar states to support our energy-efficient broadcast communication strategy, aimed at minimizing laser power utilization. Additionally, active splitters and dynamic laser power management techniques are employed to fine-tune laser power for both unicast and broadcast operations, effectively curbing overall energy consumption. These advancements mark significant progress in meeting the escalating demand for efficient ML accelerators, underscoring the pivotal role of interposer communication in chiplet-based ML acceleration architectures.

Chapter 7 introduces a solution named SCRIPT as a routing framework designed specifically to improve the security and performance of 2.5D chiplet systems against Distributed Denial-of-Service (DDoS) attacks. SCRIPT aims to mitigate security concerns inherent in heterogeneous chiplet integration by employing obfuscation routing techniques on the NoI. By factoring in chiplet trust levels and utilizing a multi-objective optimization approach, SCRIPT enhances network resilience not only against single-source Denial-of-Service (DoS) attacks but also in DDoS scenarios. Through evaluations, it is demonstrated that SCRIPT significantly improves NoI security by at least 64% during DDoS attacks, while minimizing any performance degradation.

## 8.2 Future Work

This dissertation has explored and addressed several critical challenges in the design of NoCs for future high-performance computing systems, with a particular focus on 3D and 2.5D chiplet integration. The proposed solutions offer significant improvements in terms of latency, energy efficiency, fault tolerance, and security. However, there are still exciting opportunities for further research in this domain:

### 8.2.1 Integration of AdEle+ and ReD

We suggest exploring the integration of AdEle+ with ReD to create a routing solution for PC-NoCs. This combined approach could offer improved adaptability to traffic patterns and fault tolerance while maintaining the benefits of both individual algorithms. Since AdEle+ does not consider faulty VLs, incorporating faulty VL in the optimization will make AdEle+ a comprehensive routing framework for PC-NoCs.

### 8.2.2 Optimize ReSiPI Architectures to Address Non-Ideal Frequency Response of PCMs

We suggest exploring the optimization of ReSiPI architectures to address the non-ideal frequency response of PCM couplers. The frequency response of the PCM-based devices exhibits varying coupling ratios across different wavelengths. To achieve a high bandwidth, it is essential to evaluate the characteristics of the PCM couplers and improve the ReSiPI architecture accordingly. This optimization aims to enhance energy efficiency while accommodating a wide range of wavelengths to support high-bandwidth communication.

### 8.2.3 On-chip Network Monitoring and Machine Learning

We propose to investigate the application of machine learning techniques and prediction models for on-chip network monitoring and dynamic traffic management. This could enable real-time

optimization of routing strategies like ReD and ReSiPI to adapt to changing workloads in real-time.

### 8.2.4 Co-design of Communication Architectures and Machine Learning Workloads

We suggest exploring the co-design of chiplet-based communication architectures and emerging ML workloads. This co-design could lead to more efficient network topologies and communication protocols specifically tailored for the communication patterns of different machine learning algorithms, potentially surpassing the performance of general-purpose NoC designs. Moreover, considering the joint optimization of machine learning workload mapping and switch blocking on the photonic interposer could be an interesting challenge to solve.

### 8.2.5 Security Enhancements in SCRIPT

Further research on SCRIPT could focus on enhancing its security features. This might involve exploring techniques for dynamic trust evaluation of chiplets, integration with hardware security modules, and investigating mechanisms for detecting and mitigating novel DDoS attack vectors in chiplet systems.

### 8.2.6 Scaling SwInt for Large-scale Systems

We suggest investigating techniques for scaling SwInt to support even larger-scale machine-learning accelerators. This could involve exploring hierarchical network topologies and multi-layer interposer designs to accommodate the growing communication demands of increasingly complex DNN models.

### 8.2.7 Efficient electronic controllers

We propose to investigate designing and implementing efficient electronic controllers that can manage the photonic network and optimize its operation for different DNN communication pat-

terns. This could involve employing machine learning techniques or hardware accelerators within the controllers.

# Bibliography


[1] Florentine Dubois et al. Elevator-first: A deadlock-free distributed routing algorithm for vertically partially connected 3D-NoCs. *IEEE TC*, 62(3):609–615, 2011.

[2] Ying Ping Zhang, Taikyeong Jeong, Fei Chen, Haiping Wu, Ronny Nitzsche, and Guang R Gao. A study of the on-chip interconnection network for the ibm cyclops64 multi-core architecture. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, pages 10–pp. IEEE, 2006.

[3] Luca Benini and Giovanni De Micheli. Networks on chips: A new SoC paradigm. *computer*, 35(1):70–78, 2002.

[4] Mohamad FallahRad, Ahmad Patooghy, Hesamedin Ziaeeziabari, and Ebadollah Taheri. Cirket: A performance efficient hybrid switching mechanism for noc architectures. In *2016 Euromicro Conference on Digital System Design (DSD)*, pages 123–130. IEEE, 2016.

[5] Abbas Sheibanyrad, Frédéric Pétrot, Axel Jantsch, et al. *3D integration for NoC-based SoC Architectures*. Springer, 2011.

[6] Alexandre Coelho et al. Fl-runs: A high-performance and runtime reconfigurable fault-tolerant routing scheme for partially connected three-dimensional networks on chip. *IEEE Transactions on Nanotechnology*, 18:806–818, 2019.

[7] Aqeeb Iqbal Arka, Srinivasan Gopal, Janardhan Rao Doppa, Deukhyoun Heo, and Partha Pratim Pande. Making a case for partially connected 3D NoC: NFIC versus TSV. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 16(4):1–17, 2020.

[8] Maryam Bahmani, Abbas Sheibanyrad, Frédéric Pétrot, Florentine Dubois, and Paolo Durante. A 3D-NoC router implementation exploiting vertically-partially-connected topologies. In *2012 IEEE Computer Society Annual Symposium on VLSI*, pages 9–14. IEEE, 2012.

[9] Ebadollah Taheri, Karim Mohammadi, and Ahmad Patooghy. On–off: a reactive routing algorithm for dynamic thermal management in 3d nocs. *IET Computers & Digital Techniques*, 13(1):11–19, 2019.

[10] Ebadollah Taheri, Ahmad Patooghy, and Karim Mohammadi. Cool elevator: A thermal-aware routing algorithm for partially connected 3d nocs. In *2016 6th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 111–116. IEEE, 2016.

[11] Yiwen Shen, Xiang Meng, Qixiang Cheng, Sébastien Rumley, Nathan Abrams, Alexander Gazman, Evgeny Manzhosov, Madeleine Strom Glick, and Keren Bergman. Silicon photonics for extreme scale systems. *Journal of Lightwave Technology*, 37(2):245–259, 2019.

[12] Xiaolu Wang, Huaxi Gu, Yintang Yang, and Kun Wang. A group-based laser power supply scheme for photonic network on chip. *IEEE Photonics Journal*, 11(3):1–14, 2019.

[13] Asif Mirza, Febin Sunny, Peter Walsh, Karim Hassan, Sudeep Pasricha, and Mahdi Nikdast. Silicon photonic microring resonators: A comprehensive design-space exploration and optimization under fabrication-process variations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[14] Asif Mirza, Febin Sunny, Sudeep Pasricha, and Mahdi Nikdast. Silicon photonic microring resonators: Design optimization under fabrication non-uniformity. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 484–489. IEEE, 2020.

[15] Fengjuan Wang, Zhangming Zhu, Yintang Yang, Xiangkun Yin, Xiaoxian Liu, and Ruixue Ding. An effective approach of reducing the keep-out-zone induced by coaxial through-silicon-via. *IEEE Transactions on Electron Devices*, 61(8):2928–2934, 2014.

[16] T Frank, C Chappaz, P Leduc, L Arnaud, F Lorut, S Moreau, A Thuaire, R El Farhane, and Lorena Anghel. Resistance increase due to electromigration induced depletion under TSV. In *2011 International Reliability Physics Symposium*, 2011.

[17] Ashkan Eghbal, Pooria M Yaghini, Nader Bagherzadeh, and Misagh Khayambashi. Analytical fault tolerance assessment and metrics for TSV-based 3D network-on-chip. *IEEE Transactions on computers*, 64(12):3591–3604, 2015.

[18] Ebadollah Taheri, Mihailo Isakov, Ahmad Patooghy, and Michel A Kinsy. Addressing a new class of reliability threats in 3-D network-on-chips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(7):1358–1371, 2020.

[19] Behrad Niazmand et al. Logic-based implementation of fault-tolerant routing in 3D network-on-chips. In *NOCS*, pages 1–8. IEEE, 2016.

[20] Yuxiang Fu, Chuan Zhang, Wenqing Song, Qinyu Chen, Hui Chen, Minghao Zhou, and Li Li. Optimizing vertical link placement and congestion aware dynamic elevator assignment for partially connected 3d-nocs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(10):1957–1970, 2020.

[21] Sahar Foroutan, Abbas Sheibanyrad, and Frederic Petrot. Assignment of vertical-links to routers in vertically-partially-connected 3-D NoCs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(8):1208–1218, 2014.

[22] Yuxiang Fu, Qinyu Chen, Guoqiang He, Kai Chen, Zhonghai Lu, Chuan Zhang, and Li Li. Congestion-aware dynamic elevator assignment for partially connected 3D-NoCs. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019.

[23] Ebadollah Taheri, Sudeep Pasricha, and Mahdi Nikdast. DeFT: A deadlock-free and fault-tolerant routing algorithm for 2.5 d chiplet networks. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1047–1052. IEEE, 2022.

[24] Ronak Salamat, Misagh Khayambashi, Masoumeh Ebrahimi, and Nader Bagherzadeh. LEAD: An adaptive 3D-NoC routing algorithm with queuing-theory based analytical verification. *IEEE Transactions on Computers*, 67(8):1153–1166, 2018.

[25] Ebadollah Taheri, Sudeep Pasricha, and Mahdi Nikdast. ReSiPI: A reconfigurable silicon-photonic 2.5d chiplet network with pcms for energy-efficient interposer communication. In *41st IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2022.

[26] Sanghamitra Bandyopadhyay, Sriparna Saha, Ujjwal Maulik, and Kalyanmoy Deb. A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation*, 12(3):269–283, 2008.

[27] Minmin Jiang, Ioannis A Papistas, and Vasilis F Pavlidis. Cost modeling and analysis of tsv and contactless 3D-ICs. In *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, pages 519–524, 2020.

[28] Dimitrios Velenis, Michele Stucchi, Erik Jan Marinissen, Bart Swinnen, and Eric Beyne. Impact of 3D design choices on manufacturing cost. In *2009 IEEE International Conference on 3D System Integration*, 2009.

[29] Bing Li, Xiaohang Wang, Amit Kumar Singh, and Terrence Mak. On runtime communication and thermal-aware application mapping and defragmentation in 3D NoC systems. *IEEE Transactions on Parallel and Distributed Systems*, 30(12):2775–2789, 2019.

[30] Ronak Salamat et al. A resilient routing algorithm with formal reliability analysis for partially connected 3D-NoCs. *IEEE TC*, 65(11):3265–3279, 2016.

[31] Thomas Canhao Xu, Gert Schley, Pasi Liljeberg, Martin Radetzki, Juha Plosila, and Hannu Tenhunen. Optimal placement of vertical connections in 3d network-on-chip. *Journal of Systems Architecture*, 59(7):441–454, 2013.

[32] Amir Charif, Alexandre Coelho, Masoumeh Ebrahimi, Nader Bagherzadeh, and Nacer-Eddine Zergainoh. First-last: A cost-effective adaptive routing solution for TSV-based three-dimensional networks-on-chip. *IEEE Transactions on Computers*, 67(10):1430–1444, 2018.

[33] Jinho Lee, Kyungsu Kang, and Kiyoung Choi. REDELF: An energy-efficient deadlock-free routing for 3D-NoCs with partial vertical connections. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 12(3):1–22, 2015.

[34] Fatemeh Vahdatpanah, Mahdi Elahi, Somayeh Kashi, Ebadollah Taheri, and Ahmad Patooghy. 3DEP: a efficient routing algorithm to evenly distribute traffic over 3d network-on-chips. In *27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 237–241. IEEE, 2019.

[35] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[36] Kai-Yuan Jheng, Chih-Hao Chao, Hao-Yu Wang, and An-Yeu Wu. Traffic-thermal mutual-coupling co-simulation platform for three-dimensional network-on-chip. In *Proceedings of 2010 International Symposium on VLSI Design, Automation and Test*, 2010.

[37] Vincenzo Catania, Andrea Mineo, Salvatore Monteleone, Maurizio Palesi, and Davide Patti. Noxim: An open, extensible and cycle-accurate network on chip simulator. In *26th international conference on application-specific systems, architectures and processors (ASAP)*, pages 162–163, 2015.

[38] Nathan Binkert et al. The Gem5 simulator. *ACM SIGARCH*, 2011.

[39] Cadence Genus synthesis tool, 2015.

[40] Ebadollah Taheri et al. Adele: An adaptive congestion-and-energy-aware elevator selection for partially connected 3d nocs. In *DAC*, 2021.

[41] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The SPLASH-2 programs: Characterization and methodological considerations. *ACM SIGARCH computer architecture news*, 23(2):24–36, 1995.

[42] Christian Bienia and Kai Li. *Benchmarking modern multiprocessors*. Princeton University Princeton, NJ, 2011.

[43] Yakun Sophia Shao, Jason Clemons, Rangharajan Venkatesan, Brian Zimmer, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, et al. SIMBA: Scaling deep-learning inference with multi-chip-module-based architecture. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 14–27, 2019.

[44] Ebadollah Taheri, Mohammad Amin Mahdian, Sudeep Pasricha, and Mahdi Nikdast. Swint: A non-blocking switch-based silicon photonic interposer network for 2.5 d machine learning accelerators. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2024.

[45] Ajaykumar Kannan, Natalie Enright Jerger, and Gabriel H Loh. Enabling interposer-based disintegration of multi-core processors. In *Proceedings of the 48th international symposium on Microarchitecture*, pages 546–558, 2015.

[46] Ebadollah Taheri et al. Addressing a new class of reliability threats in 3-D network-on-chips. *IEEE TCAD*, 39(7):1358–1371, 2019.

[47] Jinwoo Kim, Gauthaman Murali, Heechun Park, Eric Qin, Hyoukjun Kwon, Venkata Chaitanya, Krishna Chekuri, Nihar Dasari, Arvind Singh, Minah Lee, et al. Architecture, chip, and package co-design flow for 2.5D IC design enabling heterogeneous IP reuse. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[48] Gabriel H Loh, Samuel Naffziger, and Kevin Lepak. Understanding chiplets today to anticipate future integration opportunities and limits. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 142–145. IEEE, 2021.

[49] Yenai Ma et al. TAP-2.5D: A thermally-aware chiplet placement methodology for 2.5D systems. In *DATE*, 2021.

[50] Febin Sunny et al. Machine learning accelerators in 2.5 d chiplet platforms with silicon photonics. In *DATE*, 2023.

[51] Ebadollah Taheri et al. Trine: A tree-based silicon photonic interposer network for energy-efficient 2.5 d machine learning acceleration. In *NocArc*, 2023.

[52] Jieming Yin, Zhifeng Lin, Onur Kayiran, Matthew Poremba, Muhammad Shoaib Bin Altaf, Natalie Enright Jerger, and Gabriel H Loh. Modular routing design for chiplet-based systems. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 726–738. IEEE, 2018.

[53] Masoumeh Ebrahimi and Masoud Daneshtalab. EbDa: A new theory on design and verification of deadlock-free interconnection networks. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 703–715, 2017.

[54] Pritam Majumder, Sungkeun Kim, Jiayi Huang, Ki Hwan Yum, and Eun Jung Kim. Remote control: A simple deadlock avoidance scheme for modular systems-on-chip. *IEEE Transactions on Computers*, 70(11):1928–1941, 2020.

[55] Ebadollah Taheri, Pooya Aghanoury, Sudeep Pasricha, Mahdi Nikdast, and Nader Sehatbakhsh. Script: A multi-objective routing framework for securing chiplet systems against distributed dos attacks. In *Proceedings of the Great Lakes Symposium on VLSI 2024*, pages 78–85, 2024.

[56] Ran Wang et al. Pre-bond testing of the silicon interposer in 2.5D ICs. In *DATE*, 2016.

[57] Haibo Zhu, Partha Pratim Pande, and Cristian Grecu. Performance evaluation of adaptive routing algorithms for achieving fault tolerance in noc fabrics. In *2007 IEEE International Conf. on Application-specific Systems, Architectures and Processors (ASAP)*, pages 42–47. IEEE, 2007.

[58] Pascal Vivet, Eric Guthmuller, Yvain Thonnart, Gael Pillonnet, César Fuguet, Ivan Miro-Panades, Guillaume Moritz, Jean Durupt, Christian Bernard, Didier Varreau, et al. Intact:

A 96-core processor with six chiplets 3d-stacked on an active interposer with distributed interconnects and integrated power management. *IEEE Journal of Solid-State Circuits*, 56(1):79–97, 2020.

[59] Perceval Coudrain, J Charbonnier, A Garnier, P Vivet, Rémi Vélard, A Vinci, F Ponthenier, A Farcy, R Segaud, P Chausse, et al. Active interposer technology for chiplet-based advanced 3d system architectures. In *2019 IEEE 69th Electronic Components and Technology Conference (ECTC)*, pages 569–578. IEEE, 2019.

[60] Christopher J Glass et al. The turn model for adaptive routing. *ACM SIGARCH Computer Architecture News*, 20(2):278–287, 1992.

[61] Yibo Wu, Liang Wang, Xiaohang Wang, Jie Han, Jianfeng Zhu, Honglan Jiang, Shouyi Yin, Shaojun Wei, and Leibo Liu. Upward packet popup for deadlock freedom in modular chiplet-based systems. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 986–1000. IEEE, 2022.

[62] Srikant Bharadwaj et al. Kite: A family of heterogeneous interposer topologies enabled via accurate interconnect modeling. In *DAC*, 2020.

[63] Mahdi Hasanzadeh, Ebadollah Taheri, Mansour Shafaei, and Ahmad Patooghy. Fastest: A concurrent strategy to test components of 3d network-on-chips. In *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 65–68. IEEE, 2019.

[64] Junshi Wang, Masoumeh Ebrahimi, Letian Huang, Xuan Xie, Qiang Li, Guangjun Li, and Axel Jantsch. Efficient design-for-test approach for networks-on-chip. *IEEE Transactions on Computers*, 68(2):198–213, 2018.

[65] Masoumeh Ebrahimi et al. Fault-tolerant routing algorithm for 3D NoC using hamiltonian path strategy. In *DATE*, 2013.

[66] Masoumeh Ebrahimi, Masoud Daneshtalab, and Juha Plosila. High performance fault-tolerant routing algorithm for noc-based many-core systems. In *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 462–469. IEEE, 2013.

[67] Yong Zou and Sudeep Pasricha. NARCO: Neighbor aware turn model-based fault tolerant routing for NoCs. *IEEE ESL*, 2(3):85–89, 2010.

[68] Sudeep Pasricha, Yong Zou, Dan Connors, and Howard Jay Siegel. Oe+ ioe: A novel turn model based fault tolerant routing scheme for networks-on-chip. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on hardware/software codesign and system synthesis*, pages 85–94, 2010.

[69] Sudeep Pasricha and Yong Zou. Ns-ftr: A fault tolerant routing scheme for networks on chip with permanent and runtime intermittent faults. In *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*, pages 443–448. IEEE, 2011.

[70] Vahid Janfaza and Elaheh Baharlouei. A new fault-tolerant deadlock-free fully adaptive routing in noc. In *2017 IEEE East-West Design & Test Symposium (EWDTS)*, pages 1–6. IEEE, 2017.

[71] Yan Li et al. *3D microelectronic packaging: From fundamentals to applications*, volume 57. Springer, 2017.

[72] Hsiang Hsiao et al. Electromigration reliability and morphologies of Cu pillar with microbump under high current density stressing. In *EPTC*, 2015.

[73] Ebadollah Taheri et al. Adele: An adaptive congestion-and-energy-aware elevator selection for partially connected 3D NoCs. In *DAC*, 2021.

[74] Kshitij Bhardwaj, Koushik Chakraborty, and Sanghamitra Roy. An milp-based aging-aware routing algorithm for nocs. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 326–331. IEEE, 2012.

[75] Vincenzo Catania et al. Cycle-accurate network on chip simulation with noxim. *ACM TOMACS*, 27(1):1–25, 2016.

[76] Sudeep Pasricha. Exploring serial vertical interconnects for 3D ICs. In *DAC*, 2009.

[77] Andrew B Kahng et al. ORION3.0: A comprehensive NoC router estimation tool. *IEEE Embedded Systems Letters*, 7(2):41–45, 2015.

[78] Yvain Thonnart, Stéphane Bernabé, Jean Charbonnier, Christian Bernard, David Coriat, César Fuguet, Pierre Tissier, Benoît Charbonnier, Stéphane Malhouitre, Damien Saint-Patrice, et al. Popstar: A robust modular optical noc architecture for chiplet-based 3d integrated systems. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1456–1461. IEEE, 2020.

[79] Aditya Narayan, Yvain Thonnart, Pascal Vivet, and Ayse K Coskun. Prowaves: Proactive runtime wavelength selection for energy-efficient photonic nocs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(10):2156–2169, 2020.

[80] Pouya Fotouhi, Sebastian Werner, Roberto Proietti, Xian Xiao, and SJ Ben Yoo. Enabling scalable disintegrated computing systems with awgr-based 2.5 d interconnection networks. *Journal of Optical Communications and Networking*, 11(7):333–346, 2019.

[81] Febin Sunny et al. ARXON: A framework for approximate communication over photonic networks-on-chip. *IEEE TVLSI*, 2021.

[82] Asif Mirza et al. Opportunities for cross-layer design in high-performance computing systems with integrated silicon photonic networks. In *DATE*, 2020.

[83] Sai Vineel Reddy Chittamuru, Dharanidhar Dang, Sudeep Pasricha, and Rabi Mahapatra. Bignoc: Accelerating big data computing with application-specific photonic network-on-chip architectures. *IEEE Transactions on Parallel and Distributed Systems*, 29(11):2402–2415, 2018.

[84] Sai Vineel Reddy Chittamuru, Srinivas Desai, and Sudeep Pasricha. Swiftnoc: a reconfigurable silicon-photonic network with multicast-enabled channel sharing for multicore architectures. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(4):1–27, 2017.

[85] Sebastian Werner, Javier Navaridas, and Mikel Luján. A survey on optical network-on-chip architectures. *ACM Computing Surveys (CSUR)*, 50(6):1–37, 2017.

[86] Sudeep Pasricha and Mahdi Nikdast. A survey of silicon photonics for energy-efficient manycore computing. *IEEE Design & Test*, 37(4):60–81, 2020.

[87] Hao Zheng, Ke Wang, and Ahmed Louri. A versatile and flexible chiplet-based system design for heterogeneous manycore architectures. In *ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[88] Jieying Zhang, Jiajiu Zheng, Peipeng Xu, Yanqun Wang, and Arka Majumdar. Ultra-low-power nonvolatile integrated photonic switches and modulators based on nanogap-enhanced phase-change waveguides. *Optics Express*, 28(25):37265–37275, 2020.

[89] Peipeng Xu, Jiajiu Zheng, Jonathan K Doylend, and Arka Majumdar. Low-loss and broadband nonvolatile phase-change directional coupler switches. *Acs Photonics*, 6(2):553–557, 2019.

[90] Ting Yu Teo, Milos Krbal, Jan Mistrik, Jan Prikryl, Li Lu, and Robert Edward Simpson. Comparison and analysis of phase change materials-based reconfigurable silicon photonic directional couplers. *Optical Materials Express*, 12(2):606–621, 2022.

[91] Parya Zolfaghari, Joel Ortiz, Cédric Killian, and Sébastien Le Beux. Non-volatile phase change material based nanophotonic interconnect. In *IEEE/ACM Design, Automation and Test in Europe (DATE)*, pages 1053–1058. IEEE, 2022.

[92] Ebadollah Taheri, Sudeep Pasricha, and Mahdi Nikdast. Deft: A deadlock-free and fault-tolerant routing algorithm for 2.5 d chiplet networks. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1047–1052. IEEE, 2022.

[93] Hosam Mekawey, Mohamed Elsayed, Yehea Ismail, and Mohamed A Swillam. Optical interconnects finally seeing the light in silicon photonics: Past the hype. *Nanomaterials*, 12(3):485, 2022.

[94] Matthias Wuttig, Harish Bhaskaran, and Thomas Taubner. Phase-change materials for non-volatile photonic applications. *Nature Photonics*, 11(8):465–476, 2017.

[95] Aditya Narayan, Yvain Thonnart, Pascal Vivet, César Fuguet Tortolero, and Ayse K Coskun. Waves: Wavelength selection for power-efficient 2.5 d-integrated photonic nocs. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 516–521. IEEE, 2019.

[96] Ishan G Thakkar, Sai Vineel Reddy Chittamuru, and Sudeep Pasricha. Run-time laser power management in photonic nocs with on-chip semiconductor optical amplifiers. In *International Symposium on Networks-on-Chip (NOCS)*, pages 1–4. IEEE, 2016.

[97] Robert Polster, Yvain Thonnart, Guillaume Waltener, Jose-Luis Gonzalez, and Eric Cassan. Efficiency optimization of silicon photonic links in 65-nm cmos and 28-nm fdsoi technology nodes. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(12):3450–3459, 2016.

[98] Kentaro Kato, Masashi Kuwahara, Hitoshi Kawashima, Tohru Tsuruoka, and Hiroyuki Tsuda. Current-driven phase-change optical gate switch using indium–tin-oxide heater. *Applied Physics Express*, 10(7):072201, 2017.

[99] Febin Sunny, Asif Mirza, Mahdi Nikdast, and Sudeep Pasricha. Crosslight: A cross-layer optimized silicon photonic neural network accelerator. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 1069–1074. IEEE, 2021.

[100] Jinwoo Kim, Gauthaman Murali, Heechun Park, Eric Qin, Hyoukjun Kwon, Venkata Chaitanya Krishna Chekuri, Nael Mizanur Rahman, Nihar Dasari, Arvind Singh, Minah Lee, et al. Architecture, chip, and package codesign flow for interposer-based 2.5-d chiplet integration enabling heterogeneous ip reuse. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(11):2424–2437, 2020.

[101] Yuan Li, Ke Wang, Hao Zheng, Ahmed Louri, and Avinash Karanth. ASCEND: A scalable and energy-efficient deep neural network accelerator with photonic interconnects. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(7):2730–2741, 2022.

[102] Robert Guirado, Hyoukjun Kwon, Sergi Abadal, Eduard Alarcón, and Tushar Krishna. Dataflow-architecture co-design for 2.5 d dnn accelerators using wireless network-on-package. In *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 806–812. IEEE, 2021.

[103] Yuan Li, Ahmed Louri, and Avinash Karanth. SPACX: Silicon photonics-based scalable chiplet accelerator for dnn inference. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 831–845. IEEE, 2022.

[104] Yuan Li, Ahmed Louri, and Avinash Karanth. SPRINT: a high-performance, energy-efficient, and scalable chiplet-based accelerator with photonic interconnects for CNN inference. *IEEE Transactions on Parallel and Distributed Systems*, 33(10):2332–2345, 2021.

[105] Ebadollah Taheri, Sudeep Pasricha, and Mahdi Nikdast. 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*, 2024.

[106] Pete Ehrett et al. Sipterposer: A fault-tolerant substrate for flexible system-in-package design. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 510–515. IEEE, 2019.

[107] Febin Sunny, Ebadollah Taheri, Mahdi Nikdast, and Sudeep Pasricha. Silicon photonic 2.5 d interposer networks for overcoming communication bottlenecks in scale-out machine learning hardware accelerators. In *2024 IEEE 42nd VLSI Test Symposium (VTS)*, pages 1–4. IEEE, 2024.

[108] Kyle Shiflett, Avinash Karanth, Razvan Bunescu, and Ahmed Louri. Flumen: Dynamic processing in the photonic interconnect. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–13, 2023.

[109] John Kim, James Balfour, and William Dally. Flattened butterfly topology for on-chip networks. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*, pages 172–182. IEEE, 2007.

[110] Mohammad Amin Mahdian et al. Thz multimode interference power divider based on groove gap waveguide configuration. *TNANO*, 2022.

[111] Mohammad Amin Mahdian, Ebadollah Taheri, and Mahdi Nikdast. Pars: A power-aware and reliable control plane for silicon photonic switch fabrics. In *2023 International Conference on Photonics in Switching and Computing (PSC)*, pages 1–3. IEEE, 2023.

[112] Jared C Mikkelsen, Wesley D Sacher, and Joyce KS Poon. Adiabatically widened silicon microrings for improved variation tolerance. *Optics Express*, 22(8):9659–9666, 2014.

[113] Michael R Watts. Adiabatic microring resonators. *Optics Letters*, 35(19):3231–3233, 2010.

[114] Cheng Li et al. Silicon photonic transceiver circuits with microring resonator bias-based wavelength stabilization in 65 nm cmos. *IEEE JSSC*, 2014.

[115] Mikael Antelius et al. An apodized soi waveguide-to-fiber surface grating coupler for single lithography silicon photonics. *Optics express*, 2011.

[116] Yaoyao Ye et al. A torus-based hierarchical optical-electronic network-on-chip for multiprocessor system-on-chip. *ACM JETC*, 2012.

[117] Hyoukjun Kwon, Prasanth Chatarasi, Vivek Sarkar, Tushar Krishna, Michael Pellauer, and Angshuman Parashar. Maestro: A data-centric approach to understand reuse, performance, and hardware cost of dnn mappings. *IEEE micro*, 40(3):20–29, 2020.

[118] Mohammed Nabeel, Mohammed Ashraf, Satwik Patnaik, Vassos Soteriou, Ozgur Sinanoglu, and Johann Knechtel. 2.5D root of trust: Secure system-level integration of untrusted chiplets. *IEEE Transactions on Computers*, 69(11):1611–1625, 2020.

[119] Mitali Sinha, Setu Gupta, Sidhartha Sankar Rout, and Sujay Deb. Sniffer: A machine learning approach for dos attack localization in noc-based socs. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 11(2):278–291, 2021.

[120] Subodha Charles, Yangdi Lyu, and Prabhat Mishra. Real-time detection and localization of dos attacks in noc based socs. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019.

[121] Subodha Charles, Yangdi Lyu, and Prabhat Mishra. Real-time detection and localization of distributed dos attacks in noc-based socs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):4510–4523, 2020.

[122] Ramon Fernandes, César Marcon, Rodrigo Cataldo, Jarbas Silveira, Georg Sigl, and Johanna Sepúlveda. A security aware routing approach for noc-based mpsocs. In *2016 29th Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–6. IEEE, 2016.

[123] Ahmad Patooghy, Mahdi Hasanzadeh, Amin Sarihi, Mostafa Abdelrehim, and Abdel-Hameed A Badawy. Securing network-on-chips against fault-injection and crypto-analysis attacks via stochastic anonymous routing. *ACM Journal on Emerging Technologies in Computing Systems*, 19(3):1–21, 2023.

[124] Dabin Fang, Huikai Li, Jun Han, and Xiaoyang Zeng. Robustness analysis of mesh-based network-on-chip architecture under flooding-based denial of service attacks. In *2013 IEEE*

*Eighth International Conference on Networking, Architecture and Storage*, pages 178–186. IEEE, 2013.

[125] Jonathan Frey and Qiaoyan Yu. A hardened network-on-chip design using runtime hardware trojan mitigation methods. *Integration*, 56:15–31, 2017.

[126] Manoj Kumar Jyv, Ayas Kanta Swain, Sudeendra Kumar, Sauvagya Ranjan Sahoo, and Kamalakanta Mahapatra. Run time mitigation of performance degradation hardware trojan attacks in network on chip. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 738–743. IEEE, 2018.

[127] Rajesh JS, Dean Michael Ancajas, Koushik Chakraborty, and Sanghamitra Roy. Runtime detection of a bandwidth denial attack from a rogue network-on-chip. In *Proceedings of the 9th International Symposium on Networks-on-Chip*, pages 1–8, 2015.

[128] Ebadollah Taheri, Ryan G Kim, and Mahdi Nikdast. Adele+: An adaptive congestion-and-energy-aware elevator selection for partially connected 3d nocs. *IEEE Transactions on Computers*, 2023.

[129] Sirui Qi, Yingheng Li, Sudeep Pasricha, and Ryan Gary Kim. Moela: A multi-objective evolutionary/learning design space exploration framework for 3d heterogeneous manycore platforms. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2023.