Networks-on-Chip: from the Optimization of Traditional Electronic NoCs to the Design of Emerging Optical NoCs

Author
Marta ORTÍN OBÓN

Supervisors
Dr. Víctor VIÑALS YÚFERA
Dr. María VILLARROYA GAUDÓ

DISSERTATION
Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the University of Zaragoza

Grupo de Arquitectura de Computadores
Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Instituto de Investigación en Ingeniería de Aragón
Universidad de Zaragoza

February 2016

Acknowledgements

I want to start by thanking my advisors, Víctor and María, for their help and guidance during all these years. Thank you very much for your knowledgeable advice at every step of the way. Also Darío and Cruz, who have worked with me with the same dedication even though they were very far away. I extend my acknowledgement to all the members of the Computer Architecture Group, who have not hesitated to assist me with everything I have needed. Very special thanks to Jorge for his invaluable help at the beginning of the thesis, you always had a smile and a joke for me, no matter how many questions I asked every day.

During the thesis I was lucky to spend a few months at the University of Ferrara. Davide, thank you very much for being so helpful from the first email and welcoming me as one more of your students from the very first day. I am tremendously grateful for our collaboration, I would not have been able to get to where I am now if it had not been for you. Thanks also to Luca and Marco, who spent countless hours teaching me new stuff, it has been a pleasure to work closely with you. And of course to the rest of the guys in the lab (Hervé, Gabriele, Lorenzo, and Alberto) for making my experience there very enjoyable, showing me how cooperative a lab could be, and sharing several late nights of work with me.

I also had a wonderful internship at Munich, for which I have to thank my manager Wolfgang, thank you for making me a part of the team and valuing my work as you did. Francisco, thank you for making my internship possible and for all your help in every aspect while I was in Germany. I remember also the rest of the members of the VP team, who always had kind words towards me: Rauf, Renate, Richard, and Gerald, I really enjoyed working with you. And of course thanks to everyone else in the group (David, Julio, Arthur, Matthias, and many more I am forgetting), who created a great work atmosphere and were always great lunch and snack buddies.

I would also like to thank my fellow PhD students with whom I have shared joy and frustration: Sergio, Xandra, Edu, Oli, María Astón, and Joaquín, the best moments of every day were always those that I spent with you. And of course I include other friends I can always trust to have the best and funniest times: Bea, Rubén, Laura, Gorka, Cere, Isma, Jose, and many others, thanks guys for spending your free time with me. Thanks also to Quique, you always find the bright side in everything and have motivated me to give my best in every aspect of my life. Finally, thanks to my family for their unconditional support and always pointing me in the right direction.

Project Framework

This thesis has been developed in the Computer Architecture Group of the University of Zaragoza, within the Instituto de Investigación en Ingeniería de Aragón, in the framework of the following projects: Interconnection and Memory on Scalable Computers (TIN2010-21291-C02-01), and Memoria, Interconexión y Aplicaciones para Computadores Eficientes, Jerarquía de Memoria y Aplicaciones (TIN2013-46957-C2-1-P). It has also been funded by the Gobierno de Aragón and the Spanish Ministry of Education, Culture, and Sports (FPU12/02553).

Part of this work is the result of an internship with professor Davide Bertozzi at the University of Ferrara (Italy) funded by the European Network of Excellence HiPEAC (October 2013 - January 2014), which led to a fruitful collaboration. During the elaboration of the thesis, there was also an internship at Intel Mobile Communications GmbH in Munich under the supervision of Wolfgang Pauli (November 2014 - March 2015), which did not lead to any publications due to confidentiality of the work, but complemented the research training with a practical approach in a leading company.

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Instituto Universitario de Investigación en Ingeniería de Aragón
Universidad Zaragoza

università di ferrara
DA SEICENTO ANNI GUARDIAMO AVANTI.

Intel Mobile Communications

Networks-on-Chip: from the Optimization of Traditional Electronic NoCs to the Design of Emerging Optical NoCs

Executive Summary

As technology improves, memories and processors become faster, smaller, cheaper, and more energy-efficient, enabling computer architects to include more of them in a single chip. Now that Moore's Law is reaching its limit, the replication of simple cores is being used to continue improving performance while minimizing fabrication costs. As a consequence, performance now faces a bottleneck not only in computing power and memory access, but also in the communication of the chip elements. In this context, interconnection networks have emerged to replace buses as the prevailing solution to provide fast, cost-effective, and scalable communications. They are the key for the success of future digital systems, both chip multiprocessors composed of tens of identical cores and heterogeneous systems-on-chip. In the last decade, there has been an extensive research effort towards optimizing the networks-on-chip (NoCs) from low-level physical aspects all the way up to system-level and application-related issues, and NoCs have now reached a mature level of development with their integration as a fundamental component in many successful commercial products.

In this thesis, we start by analysing the state-of-the-art of electronic networks-on-chip and detect that, even though the fundamental purpose of the interconnect is to exchange information among processors and memories, it is often designed and optimized without taking those essential components into consideration. We revisit the comparison of several well-known topologies from a comprehensive point of view: running realistic applications on a detailed model of the processors, the caches, and the interconnect. This study identifies the dominant impact of the network latency and designates the concentrated mesh as the most cost-effective topology. Based on those observations, we propose a mechanism called Reactive Circuits that successfully leverages the information provided by the coherence protocol to reduce power and area, and improve performance. It uses the requests that travel across the network to dynamically build circuits for their replies, and is able to remove unnecessary buffer space and coherence messages.

As multiprocessors continue to scale, it is more challenging for these electronic networks-on-chip to meet their communication demands within the power budget. In consequence, a new technology is coming to the forefront with the objective of providing higher bandwidth and shorter latencies with reduced energy consumption. Although optical networks have already proved themselves useful for long distances and chip-to-chip communications, their establishment as on-chip networks still requires an intensive research effort, both in designing the new required mechanisms and devices, and in successfully proving their superiority against their electronic counterparts to compensate for the cost of migrating to the new technology.

In this work, we present an algorithm to automatically generate the communication matrices for optical rings with any number of nodes and minimum number of waveguides and wavelengths, and calculate their power consumption. We also introduce the first complete network interface architecture for optical networks and demonstrate it is responsible for most of the complexity of the optical NoC, both in latency and power. We then analyse the feasibility of the optical interconnect technology when integrated into industry-relevant objects: a chip multiprocessor and a general purpose multicore accelerator. By performing an accurate crossbenchmarking against an optimized electronic NoC, we determine that the optical network is better in latency and energy-per-bit, but still needs to be optimized in terms of power. With the objective of tackling that issue as well as adapting the optical networks to the virtualization paradigm frequent in multicore accelerators, we design the first algorithm to partition an optical network while minimizing the number of used wavelengths, thus reducing power.

Networks-on-Chip: from the Optimization of Traditional Electronic NoCs to the Design of Emerging Optical NoCs

Resumen ejecutivo

A medida que la tecnología mejora, las memorias y procesadores se vuelven más rápidos, pequeños, baratos y eficientes, permitiendo a los arquitectos de computadores incluir más en un mismo chip. Actualmente, la Ley de Moore está alcanzando su límite, por lo que la replicación de cores simples está siendo utilizada para mejorar el rendimiento al mismo tiempo que se minimizan los costes de fabricación. En consecuencia, la computación y el acceso a memoria no son ya los únicos cuellos de botella que debemos optimizar, debemos añadir también la comunicación de los elementos dentro del chip. En este contexto, las redes de interconexión han surgido reemplazando a los buses como la solución predominante para proporcionar comunicaciones rápidas, eficientes y escalables. Estas redes son la clave del éxito de los sistemas digitales futuros, tanto multiprocesadores en chip compuestos por decenas de cores idénticos, como sistemas heterogéneos. En la última década, se ha producido un importante esfuerzo investigador hacia la optimización de las redes en chip en todos los niveles: desde aspectos físicos de bajo nivel hasta el nivel de sistema y aplicaciones. Estas redes han alcanzado un nivel de desarrollo muy maduro y han sido integradas como un componente fundamental en productos comerciales exitosos.

En esta tesis comenzamos analizando el estado del arte de las redes en chip electrónicas. Detectamos que, a pesar de que el objetivo fundamental de la red es intercambiar información entre procesadores y memorias, es muy frecuente que se diseñe y optimice sin considerar estos componentes esenciales. Revisitamos la comparación de varias topologías muy conocidas desde un punto de vista amplio: ejecutamos aplicaciones realistas en un modelo detallado de los procesadores, las caches y la red. En este estudio identificamos el gran impacto de la latencia de la red, e indicamos que la malla concentrada es la topología más efectiva en cuanto a rendimiento y coste. Basándonos en esas observaciones, proponemos un mecanismo llamado Circuitos Reactivos que aprovecha la información que proporciona el protocolo de coherencia para reducir la energía y el área, y mejorar el rendimiento. Utiliza las peticiones que viajan por la red para construir dinámicamente circuitos para sus respuestas, y es capaz de eliminar almacenamiento y mensajes de coherencia innecesarios.

Conforme los procesadores continúan escalando, satisfacer las demandas de comunicación dentro del presupuesto energético supone un reto cada vez mayor para las redes en chip electrónicas. En consecuencia, la comunicación óptica está adquiriendo importancia con el objetivo de proporcionar mayor ancho de banda y menores latencias con consumo energético reducido. La utilidad de las redes ópticas ya ha sido demostrada en distancias largas y comunicaciones entre chips, pero su uso en redes en chip todavía necesita un esfuerzo investigador intensivo. Es necesario diseñar nuevos mecanismos y dispositivos, y probar la superioridad de las redes ópticas frente a su equivalente electrónico para compensar el coste de migrar a la nueva tecnología.

En este trabajo presentamos un algoritmo que genera automáticamente las matrices de comunicación para anillos ópticos con cualquier número de nodos, minimizando el número de guías y longitudes de onda utilizadas, y permite calcular su consumo energético. También mostramos la primera arquitectura completa de una interfaz de red para redes ópticas y demostramos que es responsable de la mayor parte de la complejidad de la red óptica, tanto en latencia como en energía. Después analizamos viabilidad de las redes ópticas en plataformas relevantes para la industria: un multiprocesador en chip y un acelerador de varios cores de propósito general. Llevamos a cabo una comparación precisa con una red electrónica optimizada, y determinamos que la red óptica es mejor en latencia y energía-por-bit, pero todavía necesita optimización en cuanto a potencia consumida. Con el objetivo de abordar este problema al mismo tiempo que adaptamos las redes ópticas al paradigma de virtualización frecuente en los aceleradores de varios cores, diseñamos el primer algoritmo para crear particiones en una red óptica minimizando el número de longitudes de onda utilizadas y, por lo tanto, la energía.

Contents

List of Figures

List of Tables

Part I

Preliminaries

The first part of the dissertation motivates the use of interconnection networks and introduces some useful concepts about electronic and optical networks-on-chip. It also lists the contributions of the thesis and outlines the rest of this document.

Chapter 1
Introduction

Summary

This chapter introduces the basics about electronic and optical interconnection networks and outlines the contributions of this thesis.

1.1. Context and Background

The landscape of digital electronic systems has radically changed in the past 40 years: from the first available chips containing a single core and no memory, they now include dozens of cores and specialized hardware. The scaling in the number of transistors predicted by Moore's Law is now reaching its limits, forcing computer architects to come up with innovative techniques to continue improving performance and reducing power consumption. A prevailing trend to achieve that objective consists of replicating many simple nodes on a single chip, which also minimizes fabrication cost. In this context, data movement is becoming a bottleneck that may severely restrain the computational power of these platforms. As the number of communication actors escalates, dedicated wires or buses prove themselves insufficient and it becomes clear that a cost-effective communication fabric has to be designed. Interconnection networks come into place as the perfect alternative to provide high bandwidth and low latency with reduced power consumption. In this work we explore the optimization opportunities in networks-on-chip (NoCs) when designing them along with the coherence protocol, taking into consideration the communication patterns and the reactive nature of the traffic they generate.

Electronic networks-on-chip have been widely used during the past decades, but it is now becoming infeasible to accomodate the increasingly demanding requirements imposed on them. Therefore, new technologies are being explored in order to take NoCs one step further, namely carbon nanotubes, graphene nanoribbons, wireless, and optics. In this thesis, we focus on optical networks-on-chip as a strong potential candidate for the implementation of future energy-efficient NoCs.

1.1.1. Basics about Electronic NoCs

Electronic networks-on-chip (ENoCs) use shared wires (or links) and routers to build a communication infrastructure that connects all the nodes of the system, and are defined by three aspects: topology, routing, and flow control [32]. The *topology* of the network is the static arrangement of nodes and channels. The *routing* defines the path messages will follow on that topology given their source and destination with the objective of minimizing the number of routers a message has to go through to reach its destination, while balancing the use of the network resources. Finally, the *flow control* determines how resources (such as buffers and links) can be accessed by the messages that are using the network. For this purpose, messages are normally divided into smaller pieces called *flow control units* or *flits*.

Routers are composed of buffers, logical units, and a crossbar to implement routing and flow control in the network, and are typically pipelined. Based on the coherence protocol that generates the messages that will travel on the network, we may need several virtual networks in order to avoid deadlocks. A *deadlock* occurs in an interconnection network when messages are blocked waiting for others to free resources generating a circular dependency. *Virtual networks* split buffers into different classes or subsets so that messages that may generate interdependencies are not forced to contend for the same resources. Inside each virtual network, it is common to include several *virtual channels*, which are additional buffering that help avoid *head-of-line blocking*. This type of blocking takes place when a message that could obtain all the resources it needs to continue is blocked behind another one that must wait.

Depending on the flow control mechanism, networks can be classified into two categories: circuit-switching and packet-switching. In *circuit-switching* networks the path between two nodes must be set up first, and resources remain reserved for it until the communication finishes. In contrast, in *packet-switching* a message must contend for resources at every hop in the network.

Electronic networks-on-chip are frequent to connect nodes in multicore commercial processors such as Tilera's TILE*Pro*64 [144], Intel Xeon Phi [58], Intel 48-core processor [53], or IBM Power8 [138]. Many research efforts in the field of networks-on-chip have focused too closely on the problem at hand and have not considered the characteristics of the platform in which the NoC would be implemented and the real traffic it would have to support. In this work we characterise several network-on-chip topologies from a comprehensive point of view: modelling at the same time the processors and memory subsystem and running realistic applications. From the observations of this preliminary analysis, we propose a sensible

mechanism to optimize the network by leveraging the reactive nature of the traffic.

1.1.2. Basics about Optical NoCs

Optical Networks-on-chip (ONoCs) are gaining momentum as a way to improve energy consumption and bandwidth scalability in next generation multi and many-core systems. The recent remarkable advances in silicon photonics pave the way for the implementation of optical networks-on-chip (ONoCs) as an enabling technology for the integration of hundreds of cores onto the same silicon die [43]. Compared with their electronic counterparts, they offer larger bandwidth, lower latencies, and reduced energy consumption [67].

Even though some traditional NoC concepts such as topology and routing are still valid, the basic building blocks and design challenges for ONoCs are radically different. There are several devices required to implement an optical network: waveguides, microring resonators, modulators, and photodetectors [17]. *Waveguides* are the photonic equivalent of a wire. Several *wavelengths* can travel inside of a single waveguide at the same time, thus transporting several streams of data. This is called *wavelength-division multiplexing (WDM). Microring resonators* are waveguides in the shape of a ring that, located next to another waveguide, have the capability of either letting the light continue on the waveguide (when they are off-resonance) or coupling it into the ring (when they are on-resonance). Combining these elements we can build different networks and implement the routing of messages. *Modulators* translate the electrical signal into an optical signal so that it can be transmitted on the optical medium, while *photodetectors* are used to perform the reverse conversion. The *insertion loss* is the attenuation the photonic signal experiences as it goes through a device (for example, a ring resonator or a waveguide crossing). The addition of such losses along a given path will determine the amount of power the laser source needs to inject into the waveguides in order to transmit information successfully.

3-D stacking is a promising scenario for cost-effective integration of optical NoCs and electronic devices. We consider a 3-D stacked design with the processors in an electronic layer located at the bottom and the optical network in another layer vertically stacked on top of it. Through-Silicon Vias (TSVs) connect the electronic network interface in the bottom layer with the corresponding optical one (referred to as *hub*).

Even though this technology is still in a very early stage of development, it has already been proven useful for chip-to-chip communication [73], especially in the context of the network-in-package paradigm [8]. The current research frontier for on-chip interconnection networks consists of assessing the feasibility of the optical interconnect technology. A number of benchmarking efforts in the open literature are trying to investigate whether the photonic integration of future multi- and many-core systems is worth doing from a performance and power viewpoint. This thesis contributes to this work by designing a power-efficient optical ring and a network interface architecture for optical communications, which is essential but typically overlooked. Then we introduce the optical networks in two realistic platforms: a chip multiprocessor and a general purpose multicore accelerator, and compare the results with highly optimized electronic networks.

1.2. Contributions

This thesis includes the following contributions:

- Analysis and comparison of three common electronic networks-on-chip: mesh, torus, and ring, and their concentrated versions. Considering a cache-coherent shared-memory chip multiprocessor, we analyse network-specific metrics such as network latency, hop count, and link utilization, and their effect on system-level performance, power, and area. We determine that latency is the most relevant and that the concentrated mesh is the most cost-effective topology.
- We present a new mechanism for CMPs to optimize electronic networks-on-chip called Reactive Circuits. It consists of using cache coherence requests to reserve the path for their replies. A careful implementation of the proposal allows us to remove unnecessary buffers and coherence messages, greatly reducing network power consumption and area while improving performance.

- We present an algorithm to generate optical ring configurations with minimal number of waveguides and/or wavelengths to connect any number of nodes. We include physical-layout aware power calculations considering the laser distribution network.
- To the best of our knowledge, we design the first complete network interface architecture for optical networks and analyse the effect of varying its parameters. We demonstrate that it is the most significant component of an optical network in terms of both performance and power.
- We integrate an optical network on a CMP and compare its performance and power with an optimized electronic interconnect. While the optical network is superior in latency and energy-per-bit, it still has higher power consumption than its electronic counterpart.
- We integrate an optical network on a general purpose programmable accelerator following two different approaches: on one hand, we present a hybrid interconnect that consists of a global optical NoC and a local electronic NoC; on the other hand, we implement a local optical NoC by introducing the first algorithm to dynamically partition an optical NoC with the objective of saving power by switching off unused wavelengths.

1.3. Thesis Organization

This document is organized in four parts, the first one has introduced the thesis with Chapter 1. The second part is dedicated to electronic networks-on-chip and contains the following chapters: Chapter 2 describes the CMP architecture and the simulation methodology used for our work on electronic NoCs; Chapter 3 analyses several electronic NoC topologies and chooses the most cost-efficient option; Chapter 4 describes and evaluates the new Reactive Circuits mechanism to save electronic network power and improve performance. The third part of the thesis contains the work on optical networks-on-chip organized in several chapters: Chapter 5 motivates the use of optical interconnection networks and defines some useful concepts; Chapter 6 introduces an algorithm to generate optical rings with minimum number of waveguides and wavelengths and calculate their power; Chapter 7 describes the first complete network interface architecture for optical networks; Chapter 8 evaluates the use of an optical interconnection network on a chip multiprocessor; Chapter 9 shows two alternatives to include the optical network on a general purpose programmable accelerator, including the first algorithm to partition an optical network while saving power. Finally, the last part concludes the dissertation with Chapter 10.

Part II

Electronic Network-on-Chip Optimization

This part of the thesis is entirely dedicated to electronic networks-on-chip. It starts by analysing and characterising several well-known topologies from a full-system point of view to find out which network features have a higher impact on system performance and power. It then uses that information to propose the Reactive Circuits mechanism, which uses requests to build circuits on demand for replies in order to reduce network power and latency.

Chapter 2
CMP Architecture and Simulation Methodology

Summary

This chapter presents the architecture framework and simulation methodology used for the analysis and optimization of electronic networks-on-chip. We run full-system simulations of 16 and 64-core chips, carefully modelling the cores, caches, and interconnect, and simulate realistic parallel applications and multiprogrammed workloads.

2.1. Introduction

Nowadays, a single chip may contain multiple processors and a significant amount of memory. A popular trend consists of interconnecting several nodes, each of them with a core and one or more levels of private and/or shared cache memories. Nodes communicate through an interconnection network that allows them to exchange coherence messages and cache blocks, and has a major impact on overall performance, energy consumption, and area. We focus on general purpose chip multiprocessors (CMPs), where both high-performance and low-power are required in equal shares.

Only a few works study the interconnect by modelling in detail the processors, memory hierarchy, and interconnection network. Those analysis are often performed with synthetic traffic or application traces that do not entirely capture the behaviour of a real execution [24, 72, 88, 12]. This work simulates both parallel and multiprogrammed workloads with real applications, carefully modelling all the components above-mentioned. This allows us to study the effect of the interconnection network configuration on the whole system and the real interactions between the memory subsystem and the interconnect, which also lead to interesting optimization opportunities.

2.2. CMP Architecture Framework

This section presents the modelled CMP architecture and a detailed description of the interconnection network configuration, which will be used to perform a detailed topology analysis and as a baseline for comparison with our Reactive Circuits proposal.

2.2.1. General System Architecture

This work focuses on a homogeneous CMP where each tile is composed of a core with private first level cache (L1) split into data and instructions, and a bank of the shared second-level cache (L2), both connected directly to the router. Four tiles in the edges of the chip also include a memory controller. Figure 2.1 depicts the block diagram of the chip and a tile with memory controller. It also includes the connections between the elements in the tile and the router. Table 2.1 presents the key parameters of the architecture. To model it we based our design on other systems with similar characteristics, both from academia [155, 135, 13] and industry (Tilera's TILE*Pro*64 [144], Intel Xeon Phi [58], and Intel 48-core processor [53]). To size our L2 cache (which is our last level cache) we have taken a configuration very frequently used in academia [2, 4, 61] that is also a nice compromise among the sizes of shared last level caches in high and low-end commercial platforms. For example, the AMD Opteron processor has a shared L3 cache of 6 MB for 6 cores [31]; IBM Power8 has 8 to 12 cores with 8 threads per core, and includes an L3 cache with 64 to 96 MB, as well as an L4 cache with 32 to 64 MB [49]; Intel Xeon D has 1.5 MB of L2 cache per core [65]; Sparc M7 has 32 cores and 64 MB of shared L3 cache [107]. An interesting design trend for CMPs is to integrate a large number of simple cores. That is why we are modelling systems with 16 and 64 Ultrasparc III Plus single-thread in-order cores. However, we also consider the effect of multithreading by simulating a configuration with 16 cores with 4 threads each.

We use a directory-based MESI coherence protocol. All the traffic that traverses the interconnection network is a direct consequence of the memory activity, either to move cache lines (instructions or data) among tiles or for coherence management. Table 2.2 details the messages exchanged by the coherence protocol. Therefore, it is important to model the caches realistically, even though our main interest lies in the interconnect [78, 133]. This MESI protocol allows direct data transfer between L1 caches, as opposed to a simpler version that always forced to use the L2 as an intermediary.

2.2.2. Network-on-Chip and Router Architecture

The baseline NoC that connects all tiles is built with simple 4-stage routers, XY routing and wormhole flow control, running at 2 GHz (same clock frequency as the processors). Table 2.3 shows the detailed configuration of the baseline NoC and Figure 2.2 depicts the router architecture. We use two separate

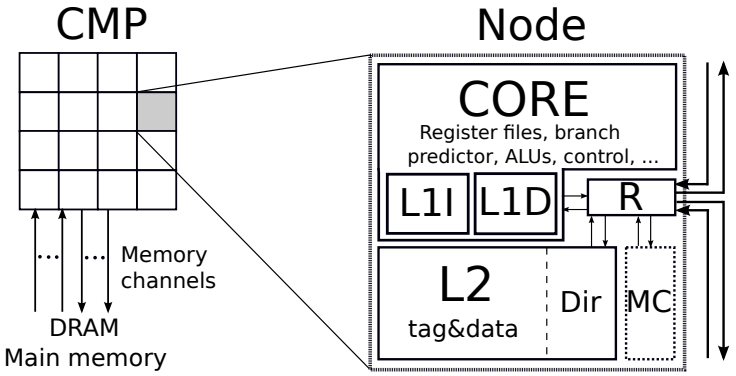


Figure 2.1: Block diagram including a chip and the components of a tile. MC stands for memory controller, R is the router, and Dir is the directory, which is included in the L2 cache. This example router has two input and two output ports connected to neighbouring tiles.

Table 2.1: Main characteristics of the chip multiprocessor.

Cores	16 single and multithreaded cores, and 64 single-threaded cores, Ultrasparc III Plus, in order, 1 instruction/cycle and thread, 2GHz frequency
Coherence protocol	Directory-based, MESI, directory distributed among L2 cache banks
Consistency model	Sequential
Private L1 cache	32KB data and instruction caches, 4-way set associative, 2-cycle hit access time, 64B line size, pseudo-LRU replacement policy
Shared L2 cache	Distributed, 1 bank/tile, 1MB per bank, 16-way set associative, 64B line size Pseudo-LRU replacement policy, inclusive, interleaved by line address 7-cycle hit access time
Memory	4 memory controllers, distributed in the edges of the chip, (both for 16 and 64-core architectures), 160-cycle latency

Table 2.2: Messages generated by the coherence protocol.

Event	**Sequence of messages**
L1 miss	1º Request from L1 to the corresponding L2 bank 2º L2_Replies: Data reply from L2 to L1 3º L1_DATA_ACK: ACK from L1 to the L2 bank
L1 miss, another L1 owns the data exclusively	1º Request from L1 to the corresponding L2 bank 2º L2 forwards the request to L1 owner 3º L1_To_L1: L1 owner sends data to L1 requestor 4º L1_DATA_ACK: ACK from L1 requestor to the L2 bank
Invalidation (write or L2 replacement)	1º Invalidation from L2 to L1 sharers 2º L1_INV_ACK: ACK from L1s to the L2 bank
L1 replacement	1º Replacement data from L1 to the corresponding L2 bank 2º L2_WB_ACK: ACK from the L2 bank to L1
L2 miss	1º Request from L2 bank to the corresponding memory controller 2º MEMORY: Data from the memory controller to L2 bank
L2 replacement	1º Replacement data from L2 bank to the corresponding memory controller 2º MEMORY: ACK from the memory controller to L2 bank

Table 2.3: Main characteristics of the baseline network on chip.

General	Two virtual networks (requests and replies), 2 virtual channels (VCs) per virtual network
Routers	4-stage pipeline: routing and input buffering, VC allocation, switch allocation, and switch traversal Round-robin 2-phase VC/switch allocators 5-flit buffers per VC, enough to store an entire message
Links	16-byte flit size (link width), 1-cycle latency

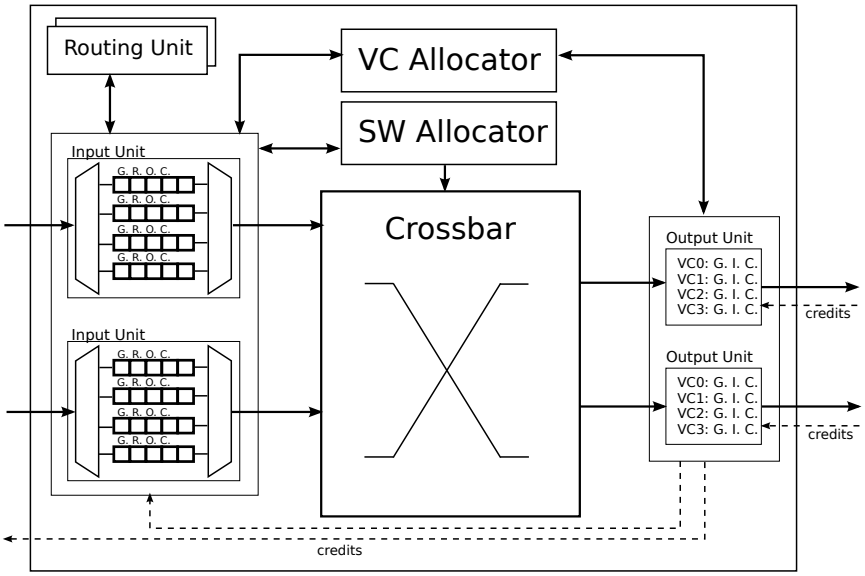


Figure 2.2: Architecture of the baseline router. VCs at the input units store global state (G), route (R), output VC (O) and credit count (C). At the output units, they store global state (G), input VC (I) and credit count (C).

virtual networks to separate traffic classes in order to avoid protocol deadlock. In practice, this means that we will need at least as many virtual channels as virtual networks. In our case, we include 2 virtual channels per virtual network (a total of 4 virtual channels) to improve performance by reducing head-of-line blocking.

2.3. Methodology

This section describes the simulation environment and workloads used in this work. This methodology is an important contribution of the thesis, placing great effort towards the accurate modelling of the whole chip and the use of representative applications.

2.3.1. Simulation Environment

We carefully model all the components of the chip and perform full system simulation of 16 (single and 4-threaded) and 64 (single-threaded) cores with Simics [92]. We include GEMS to model the memory subsystem [95], and an extended version of GARNET for the interconnection network [6]. To get the timing, area, and energy expended by the network we use DSENT [141], a state-of-the-art circuit modelling tool (with 32 nm technology and 2 GHz frequency).

2.3.2. Workloads

CMPs can execute parallel applications to reduce execution time, and multiprogrammed workloads (execution of independent programs on each core) to increase throughput. PARSEC is a benchmark suite composed of shared-memory parallel applications that focuses on emerging workloads and was designed

Table 2.4: Simulated workloads and execution methodology.

	Description	16 cores 1-thread	16 cores 4-threads	64 cores 1-thread
Parallel Workloads	From PARSEC: `blackscholes`, `bodytrack`, `canneal`, `dedup`, `ferret`, `fluidanimate`, `raytrace`, `swaptions`, `vips`, and `x264`. From SPLASH2: `barnes`, `cholesky`, `fft`, `lu_cb`, `lu_ncb`, `ocean_cp`, `ocean_ncp`, `radiosity`, `radix`, `raytrace`, `volrend`, `water_nsquared`, and `water_spatial`. Threads are automatically mapped to the cores by the operating system. Simulate the whole parallel region.	16 threads	64 threads	64 threads
Multiprog. Workloads	From SPEC CPU2006: `perlbench`, `bzip2`, `gcc`, `mcf`, `sjeng`, `libquantum`, `bwaves`, `milc`, `zeusmp`, `leslie3d`, `dealII`, `soplex`, `GemsFDTD`, `lbm`, `wrf`, and `sphinx3`. 20 different mixes with the applications randomly distributed among the cores. Applications are bound to the cores to avoid migration. Caches are warmed up for 200 million cycles and then, applications are executed for 500 million cycles.	16 apps.	16 apps, 4 times each (64 total)	16 apps, 4 times each (64 total)

to be representative of next-generation programs for chip-multiprocessors [19]. SPLASH2 is a mature benchmark suite that contains a variety of shared-memory, parallel, high performance computing, and graphics applications [152]. We use a selection of benchmarks from PARSEC and SPLASH2 with scaled inputs from PARSEC 3.0. We have used SPEC CPU2006, a benchmark suite composed of single threaded applications written in C, C++, and Fortran, to build multiprogrammed workloads in which each core runs a different application, so the only network traffic will come from cache misses and replacements [137]. We choose 16 applications with large working sets (according to [47]) to find potential bottlenecks in the interconnect.

Table 2.4 describes the workloads and their execution methodology for the different configurations under test. Table 2.5 shows the characterization of the workloads with respect to their behaviour in the memory subsystem. This helps us understand the amount of network traffic the applications generate. The most noticeable aspect is that the multiprogrammed workloads have a much lower L2 hit rate and need to access main memory more often.

Table 2.5: Characterization of the workloads with respect to their behaviour in the memory subsystem

	Parallel Applications			Multiprogrammed Workloads		
	16 cores 1-thread	16 cores 4-threads	64 cores 1-thread	16 cores 1-thread	16 cores 4-threads	64 cores 1-thread
LD/ST instructions	29.7%	26.3%	26.6%	28.3%	26.9%	26.3%
L1D hit rate	93.1%	92.9%	89.3%	95.4%	94.5%	93.0%
misses served by L2	91.8%	96.8%	95.5%	55.9%	62.1%	47.3%
misses served by main memory	8.2%	3.9%	4.5%	44.1%	37.9%	52.7%

Chapter 3
Analysis and Characterization of NoC Topologies

Summary

This chapter provides a comprehensive study of the interactions between the interconnection network and the memory hierarchy to enable a better co-design of both components. We explore the implications of the interconnect choice on overall performance by comparing the behaviour of three topologies (mesh, torus, and ring) and their concentrated versions. Simply choosing the concentrated mesh over the ring improves performance by over 40% in a 64-core chip. The key strength of this work is the holistic analysis of the network-on-chip and the memory hierarchy. Experiments are carried out with a full-system simulator that carefully models the processors (single and multithreaded), memory hierarchy, and interconnection network, and executes realistic parallel and multiprogrammed workloads. We corroborate conclusions from several previous works: network diameter is critical, the concentrated mesh offers the best area-energy-delay trade-off, and traffic is very light and unbalanced. We also provide interesting insights about application-specific features that are hidden when studying only average results. We include a fairness analysis for multiprogrammed applications, and refute the idea of the memory controller placement greatly affecting performance.

3.1. Introduction

We present an analysis of three topologies with varying degrees of complexity, performance, power, and area: mesh, torus, and ring. We model CMPs with 16 and 64 single-threaded cores, including a configuration with 16 4-threaded cores, and explore the effect of modifying the location and number of memory controllers. Our goal is to draw meaningful conclusions on the studied network configurations and analyse the details, pointing out the best choice from an integrated performance, area, and energy standpoint. We revisit the comparison of several topologies with our detailed simulation framework to update the results, validate or refute previous conclusions, and complete them with further analysis. This work has been published in [110].

3.2. Related work

Several publications have highlighted the impact of the network on performance, energy, and chip area. However, only a few papers focus on the comparison of interconnection network configurations. Balfour and Dally present an analysis of how different topologies affect performance, area, and energy efficiency [12]. However, they do not model the memory subsystem, only use synthetic traffic patterns, and do not consider simple topologies like the ring. Gilabert *et al.* focus on physical synthesis of several networks, but do not simulate real applications or systems larger than 16 cores [46]. Villanueva *et al.* highlight the importance of a comprehensive simulation framework and present results of the execution of real parallel applications and its close relationship with cache behaviour [148]. Sanchez *et al.* explore the implications of interconnection network design for CMPs [133]. We complement their results including a simple topology (ring), multiprogrammed workloads, traffic distribution analysis, the effect of memory controller placement, and the influence of the network topology on fairness.

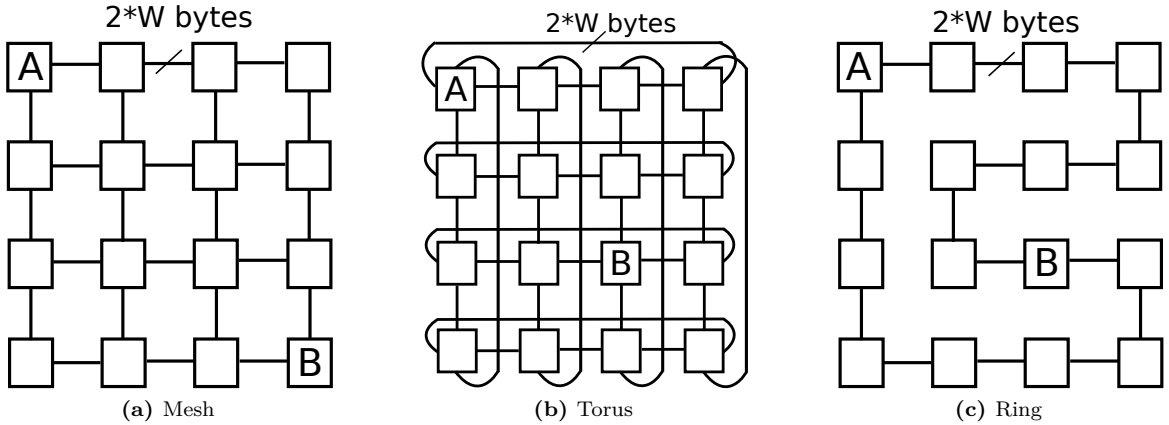
Many papers propose alternatives to conventional router architectures, topologies, and flow control methods on isolation. However, they do not consider the impact on the overall system and back up the results with network-only simulations of synthetic traffic and traces. Carara *et al.* revisit circuit-switching which, as opposed to packet-switching, allows to reduce buffer size, and guarantees throughput and latency [24]; Walter *et al.* try to avoid hotspots on systems on chip by implementing a distributed access regulation technique that fairly allocates resources for certain modules [149]; Mishra *et al.* propose an heterogeneous on-chip interconnect that allocates more resources for routers suffering higher traffic but they only get good results with a mesh topology [99]; Koibuchi *et al.* detect that adding random links to a ring topology results in big performance gains, although they only experiment with a network simulator [72]. All these studies either do not model the whole system, do not include a significant variety of real workloads, or do not experiment with different topologies. Also, most of them only include network-related metrics and fail to report on overall performance, or elaborate conclusions based on IPC (instructions per cycle), which has been reported to be unsuitable for parallel applications [156].

Another approach consists on designing the network considering the behaviour of the memory subsystem and the coherence protocol. Yoon *et al.* propose an architecture with parallel physical networks with narrower links and smaller routers that eliminates virtual channels [154]. Seiculescu *et al.* propose to use two dedicated networks: one for requests and one for replies [135]. Lodde *et al.* introduce a smaller network for invalidation messages, but only test their design with memory access traces [88]. Agarwal *et al.* propose embedding small in-network coherence filters inside on-chip routers to dynamically track sharing patterns and eliminate broadcast messages [7]. These studies try to improve the performance of the most commonly used networks, but do not venture with less conventional topologies. Also, they only experiment with a maximum of 16 cores. Krishna *et al.* propose a system to improve the frequent 1-to-many and many-to-1 communication patterns by forking and aggregating packets to avoid the increment in traffic as the number of nodes increases [75]. Bezerra *et al.* try to reduce traffic by statically mapping memory blocks to physical locations on the chip that are close to cores that access them [18]. The last two proposals are only evaluated with a typical mesh topology.

3.3. Topologies for Homogeneous CMPs: Qualitative Analysis

We compare today's most mainstream topologies: mesh, torus, and ring. Figure 3.1 shows a diagram of the three topologies for a 16-node chip. The *2D mesh* is a widespread choice for large-scale CMPs due to its regularity. Tiles are organized in a regular grid with links pointing to all 4 cardinal directions: north, south, east, and west. A *torus* is a mesh with wraparound links to reduce the average number of hops between tiles, at the cost of longer links ($\sqrt{2}$ times larger than a mesh[32]), larger area, and high power consumption. Longer links often involve higher wiring latency [151], but we kept the link latency constant for all topologies after verifying its feasibility with DSENT [141].

Driven by the observed low network occupancy and commercial NoCs [58], instead of moving towards higher-performance topologies, we opt for more efficient options to fit the power budget, such as *bidirectional rings*, which require a smaller area but have a larger diameter. Every node is connected to two other nodes using two links, one in each direction of the ring.



(a) Mesh **(b)** Torus **(c)** Ring

Figure 3.1: Diagrams of mesh, torus, and ring topologies for a 16-core CMP. Every line represents two links, one in each direction. W is the link bandwidth. A message going from A to B would be travel on one of the longest paths for each topology.

The torus and the ring have cycles in their topologies, which can lead to deadlocks. To avoid them, we implement a deadlock avoidance method by setting a dateline in each cycle where messages will be forced to use a specific virtual channel so that cycles are broken [32, 33].

Table 3.1 summarizes the main characteristics of the three topologies. Note that the comparison encompasses topologies with different bisection bandwidth, so first, we tune each topology to obtain a realistic design point, and then, we explore the trade-offs between complexity (which results in more bandwidth, power, and area) and performance for all configurations.

The number of input and output ports of the router is a direct indicator of the complexity; the higher the number of ports, the higher the area and expended energy. If we divided the network in two equal parts, the bandwidth we would have between the two parts is what we call the *bisection bandwidth*. A lower bisection bandwidth indicates that communications in the network will be slower. A *hop* in the network is a link the message traverses when going from source to destination. When counting the total number of hops we also include the local links going from the cache to the router, and from the router to the cache, so the minimum hop count is two (which corresponds to the communication between an L1 and an L2 in the same node through the router of that node: first hop from source cache to router and second hop from router to destination cache). The number of hops gives us an idea of the time it will take a message to traverse the network. In the table, we distinguish the maximum distance (also called *diameter*) and the average distance. Besides, the length of the link will have an impact on the power consumed by the network, which is modelled in DSENT.

Table 3.1: Qualitative comparison of the three topologies for a CMP system with N tiles (we assume that N will always be a perfect square). We include the basic and the concentrated versions of the topologies, with a concentration factor of c. The number of inputs/outputs does not consider tiles with a memory controller, where routers would have one more input and output, or the tiles in the edges of the mesh, where some ports would be left unused. For the concentrated topologies, indicated ports are for the global routers; local routers always have 6 ports. W is the link bandwidth and L is the link length. For the concentrated topologies, we indicate the length of the links that connect the global routers. Note that the local links have been considered in the hop count formulas. Therefore, to go from node 0 to node 1 we need 3 hops: one from cache 0 to router 0, one from router 0 to router 1, and one from router 1 to cache 1. In the average hop count, the +2 in the formulas corresponds to those local links. For the concentrated topologies, the average hop count is detailed in Table 3.2 due to its complexity.

Topology	Inputs/ outputs	Bisection BW	Max. hops (diameter)	Avg. hops (Avg distance)	Link length
2D mesh	6/6	$2W\sqrt{N}$	$2\sqrt{N}$	$\sim 2/3\sqrt{N}+2$	L
Torus	6/6	$8W\sqrt{N}$	$\sqrt{N}+2$	$\sim 1/2\sqrt{N}+2$	$L\sqrt{2}$
Ring	4/4	$4W$	$N/2+2$	$\sim N/4+2$	L
CMESH	6/6	$2W\sqrt{N/c}$	$2\sqrt{N/c}+2$	–	$2L$
CTORUS	6/6	$8W\sqrt{N/c}$	$\sqrt{N/c}+2+2$	–	$2L\sqrt{2}$
CRING	4/4	$4W$	$(N/c)/2+2+2$	–	$2L$

The simplicity of the ring topology allows us to test two improved designs. As opposed to the mesh and torus, which have 4 input and 4 output ports to the outside of the tile, the ring has only two of each. The number of ports has a direct effect on the amount of buffer space and the complexity of the switch allocator and crossbar. To make use of this idle space, we test a configuration in which we increase the link bandwidth keeping the router area slightly under that of the torus. This results in flits of 24 bytes, which will reduce the number of flits needed per message and, therefore, serialization latency (RING_FLIT24B). Following the same idea, we also include a ring configuration with reduced latency, where we merge the switch allocation and switch traversal stages, resulting in a 3-cycle router (RING_3CYCLE_R).

Connecting several tiles to the same router to build *concentrated* topologies is a popular choice to reduce the network diameter. This choice has been adopted by the new generation of the Intel Xeon Phi [59]. These designs reduce the amount of resources of the network but might introduce contention. We include concentrated versions of the topologies with a concentration factor of 4. To avoid increasing the router radix, we use *external* concentration with local routers, which allows us to maintain routers with a small area and high frequency with only a small performance degradation [77]. For the 16-core chips we implement a concentrated mesh (CMESH), as depicted in Figure 3.2. Memory controllers are connected directly to the global router. With only four global routers, the concentrated ring topology is equivalent to the CMESH; the concentrated torus would have additional links, but we omit the results because the higher bandwidth does not benefit performance and increases power and area. For 64 cores, we model the CMESH, CTORUS, and CRING. Tables 3.1 and 3.2 include the characteristics of the concentrated versions of the topologies.

3.4. Topologies for Homogeneous CMPs: Quantitative Analysis

This section presents the main contributions of our analysis for 16 and 64-core architectures. We include both system-oriented metrics (performance, area, energy, and fairness) and network metrics (hop count, network latency, and traffic distribution). We conclude with an analysis of the impact of memory controller placement.

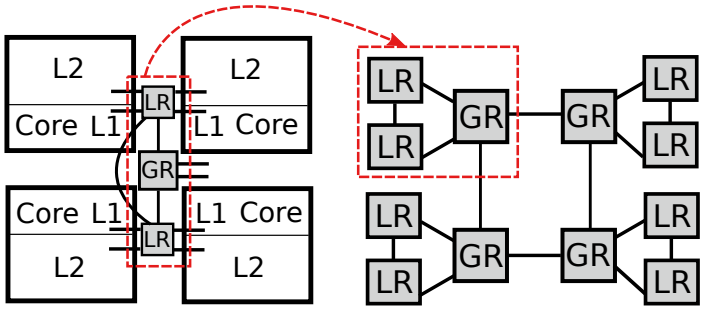


Figure 3.2: Connection of the nodes to the routers within a four-node cluster (left) and organization of all local and global routers (right) for a concentrated mesh in a 16-core chip. LR and GR stand for local router and global router, respectively. The lines that connect routers represent always two links, one in each direction.

Table 3.2: Average hop count for the concentrated mesh, torus and ring topologies. N is the number of tiles and c is the concentration factor. The formula is divided into the inter and intracluster communications, indicating the probability of each one and the hops in the global network and inside the clusters. Note that the hops to access and leave the network are now 4 (compared with 2 for the non-concentrated topologies), because messages need to traverse the local routers.

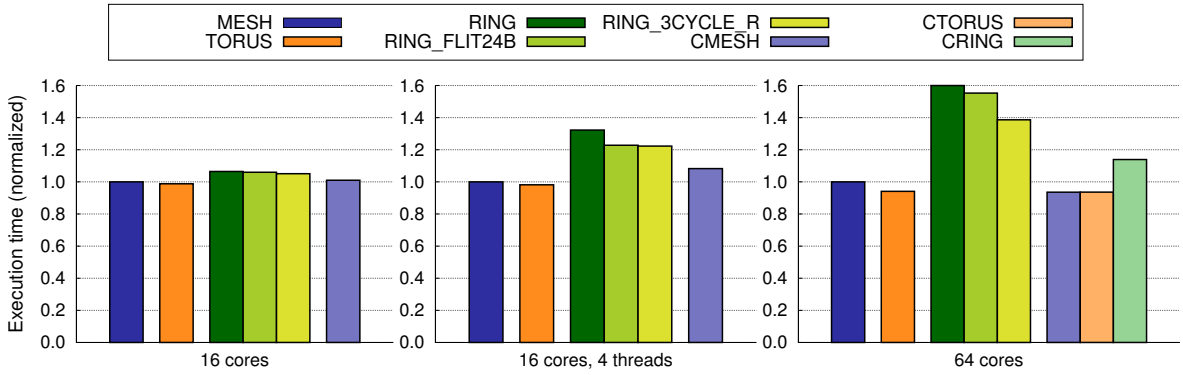
	Intercluster Communications			**Intracluster Communications**		
Topology	**Probability**	**Hops global network**	**Hops in cluster**	**Probability**	**Hops global network**	**Hops in cluster**
CMESH	$1-(c/n)$	$2/3\sqrt{N/c}$	4	c/n	0	2.5
CTORUS	$1-(c/n)$	$1/2\sqrt{N/c}$	4	c/n	0	2.5
CRING	$1-(c/n)$	$(N/c)/4$	4	c/n	0	2.5

3.4.1. Performance

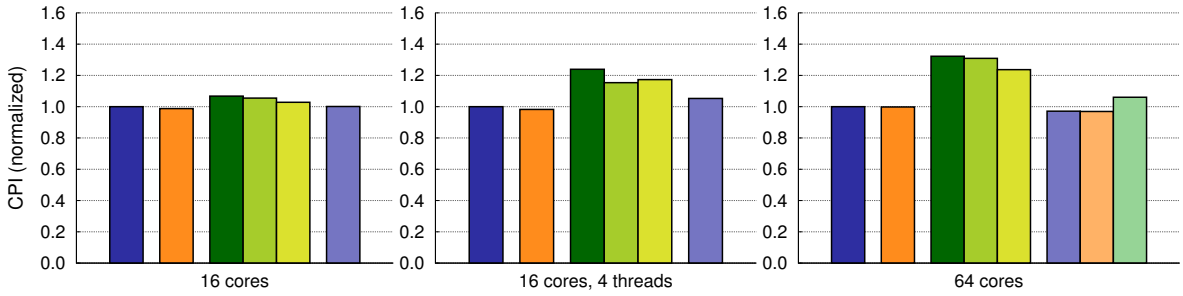
To compare the impact of the network configurations on performance, we analyse the number of processor cycles it takes for the parallel workloads to complete the parallel section; for the multiprogrammed workloads, we check how many instructions get executed in 500 million cycles. Figure 3.3 represents the average execution time for the parallel applications and the average CPI (cycles per instruction) for the multiprogrammed workloads, both normalized to the mesh topology. In 16-core single-threaded architectures differences between topologies are small, with the ring with 3-cycle routers and the CMESH being very similar to the mesh, and the torus performing only slightly better. Differences are more pronounced in the multithreaded configurations because the network needs to support a higher load, and topologies with fewer resources are more congested. In 64-core chips, the performance of the ring topologies drops significantly while the concentrated topologies stay very close to the mesh and torus. The conclusions are the same for both parallel and multiprogrammed workloads, and they are along the same line as the most recent industry developments: for the second generation of the Xeon Phi multicore processor, Intel has replaced the ring with a concentrated 2D mesh [58, 59].

Regarding the absolute CPI values for the multiprogrammed workloads, the CPI of each core in the mesh topology is 4.4 for 16 cores with 1 thread, 4.7 for 16 cores with 4 threads, and 6.1 for 64 cores. Even though the miss rate is small, the penalty of a cache miss greatly increases the CPI, with the highest portion of the miss latency coming from the network latency. Therefore, we can conclude that the impact of the NoC on performance is large.

In the following sections we analyse network-specific metrics and demonstrate how they influence the system performance. We show that the network is lightly loaded and that performance is a direct consequence of the number of hops it takes a message to go from source to destination.



(a) Parallel applications



(b) Multiprogrammed workloads

Figure 3.3: Average execution time for the parallel applications and CPI (cycles per instruction) for the multiprogrammed workloads, normalized to the mesh, for 16 single and multithreaded cores and 64 single-threaded cores. In every case, the lower the bars, the better.

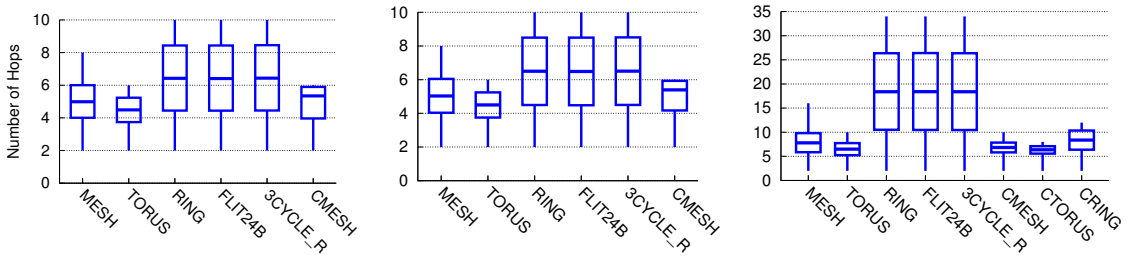
3.4.2. Average Hop Count

The diameter of the network is critical and concentrated topologies offer faster communications even though messages have to share a smaller amount of routers and links. In Figure 3.4 we present candlestick charts for the hop count of all the configurations, which show the minimum and maximum values, and the three quartiles. The differences among the workloads are very small. The first thing we notice is that the hop count directly reflects the performance results, showing that this metric determines the performance. Both the median and the variability of the hop count are much larger for the ring topologies, especially with 64 cores, which is a clear indicator of where we experience more pronounced performance drops. Hop count for the three ring topologies is the same in all cases because the variations do not affect the topology. However, performance is different because for the ring with 24-byte flits, data messages are only three flits long instead of five, which reduces the serialization latency; for the ring with 3-cycle routers, every hop takes a smaller number of cycles.

The high impact of the hop count and, more generally, the network latency, is partly due to the simple in-order cores of our system. More complex cores capable of running instructions out-of-order and non-blocking caches would be able to hide some of the network latency by executing other instructions in parallel with the L2 or memory access. However, the pressure on the caches would not increase enough to give relevance to network throughput, because it has been demonstrated that supporting only 2 in-flight misses is enough to eliminate most of the memory stall cycles [63].

3.4.3. Network Latency

Network congestion can delay messages and have a large impact on system performance. Figure 4.5 represents the *network latency* split in *base latency* (cycles it would take packets to traverse the network without contention), *blocking latency* (extra time spent in the network due to contention), and *queueing*



(a) 16 cores **(b)** 16 cores, 4 threads **(c)** 64 cores

Figure 3.4: Average hop count for 16 single and multithreaded cores and 64 single-threaded cores. There are no differences between parallel and multiprogrammed workloads, so results are averaged together in all configurations. We present candlesticks, where we can see the minimum, maximum, median, lower quartile (Q1), and upper quartile (Q3) values. Note that scales are different. Hops to traverse the local links from and to the nodes are included in the count.

latency (time a message is waiting in the network interface before it can get a free virtual channel to enter the network). The latency for parallel and multiprogrammed workloads is very similar, with the multiprogrammed workloads having slightly higher blocking latency in some cases. That is because these workloads access main memory more often (as we will demonstrate in section 3.4.4), which creates a bottleneck in the nodes of the network close to the memory controllers.

We can clearly see that the blocking latency is a small percentage of the total latency in the single-threaded configurations: 14% for 16 cores and 16% for 64 cores. It significantly increases with multithreaded cores, reaching an average of 31%, because the higher traffic load creates some congestion, especially in networks with fewer resources (ring and concentrated mesh).

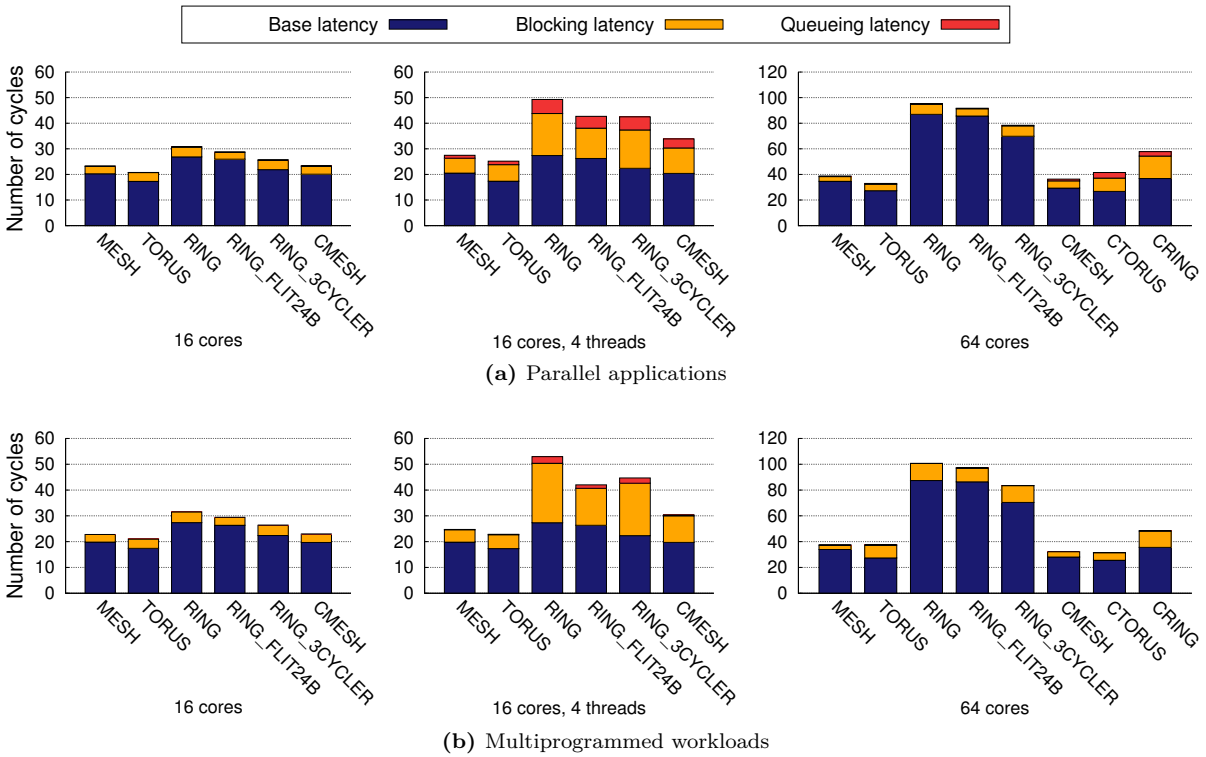
If we focus on the two optimized ring versions, we notice that one of them does not consistently have shorter latency than the other. The 3-cycle router version is normally better, with a shorter base latency. This is because all messages benefit from faster transmissions, while in the 24-byte flit version, only data messages, which need more than one flit, improve their latency. However, in the configurations with 4-threaded cores, the blocking latency is shorter for the ring with 24-byte flits. In those cases when there is more traffic in the network, a nice effect of having larger flits becomes relevant: messages with fewer flits can traverse the network in a more compact way, reducing the number of cycles in which they occupy several routers at a time, thus reducing the probability of conflicts with other messages. This improvement in the blocking latency results in shorter network latency for the 24-byte flit ring in the multithreaded configuration with multiprogrammed workloads.

As we already mentioned in the previous section, we can notice again that the network latency results correspond directly with the average hop count and the system performance (the shorter the latency, the better the performance), which demonstrates the huge impact the network has on the system. Nevertheless, the network is mostly lightly loaded, so although it may take long to traverse it, resources are idle most of the time. These results ratify the conclusions of Sanchez *et al.*, which point out that the number of hops is the most critical parameter of the network [133].

3.4.4. Traffic Distribution

To analyse traffic distribution, we measure the number of injected flits per node. We notice that traffic is unevenly distributed in the interconnect, meaning that some resources will be used more often than others. In this section, we present results for `canneal` as a representative example of parallel applications, and a multiprogrammed mix. For a given application and number of cores, the distribution remains constant when we change the network topology, so we illustrate the results only for the mesh, torus, ring, and CMESH. Conclusions still hold for all applications and number of cores, so we focus on a 64-core chip.

Figure 3.6 depicts a heat map of injected flits per cycle for each node for `canneal` and a multipro-



(a) Parallel applications

(b) Multiprogrammed workloads

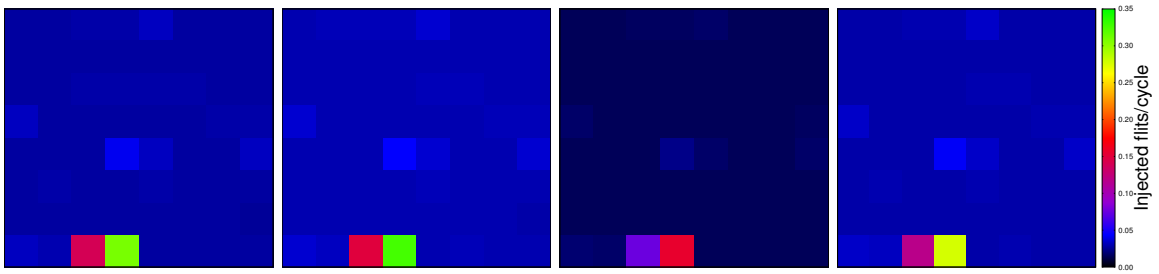
Figure 3.5: Average network latency in number of cycles broken down into base, blocking, and queueing latency for 16 single and multithreaded cores and 64 single-threaded cores. Note that scales are different.

grammed mix executed on 64 cores. All the traffic is generated by the memory subsystem, so every action has a reaction (request-reply, invalidation-ack). Hence, the heat maps also indicate which nodes are receiving messages more often. The number of flits per cycle is smaller for the ring because a very similar amount of traffic gets injected in a much larger period of time. Nevertheless, the distribution of traffic is the same regardless of the topology: certain nodes inject more flits than others. In the parallel workloads such as `canneal` (Figures 3.6a to 3.6d), this is because a couple of L2 banks are being accessed more frequently than others, which depends on the physical distribution of the data touched by each application. The rest of the simulated parallel applications exhibit similar patterns, with 2 out of the 64 nodes injecting more than 40% of the traffic in many applications.

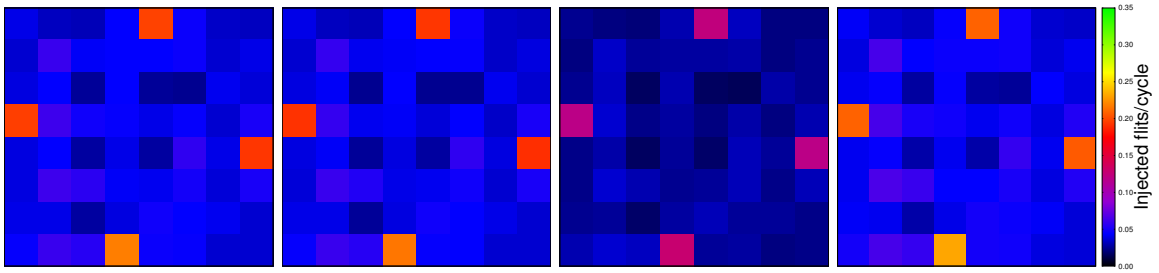
Figures 3.6e to 3.6h show results for one of the multiprogrammed mixes. In this case, we see four clear hotspots in the edges of the chip, where the memory controllers are located. The multiprogrammed workloads access main memory more often than parallel applications. Apart from that, the rest of ideas we introduced for parallel workloads are still valid. There are several parallel applications which have a larger working set and need to access main memory more often (`fft`, `ocean_cp`, `ocean_ncp`, and `radix`). For those applications, we also see more flits injected from the nodes with memory controllers.

Figure 3.7 depicts the number of flits per cycle that traverse the network links for `canneal` and multiprogrammed mix. We see that link utilization is higher around the nodes with higher injection rates. Also, it is higher in the ring topologies, since there are fewer links to transport the same amount of information. The torus wastes more resources because it shows the lowest link usage, even though it injects the highest number of flits per cycle. In the execution of the multiprogrammed workload in the mesh topology, we detect that links are used more often in the center of the chip, which is the characteristic behaviour for this topology with uniform traffic. The location of the memory controllers in the edges of the chip increases link usage in the center.

Evaluating all the results of this section, we notice that the network is lightly loaded, even around the most active nodes; furthermore, some parts of the network are idle most of the time. Consider-

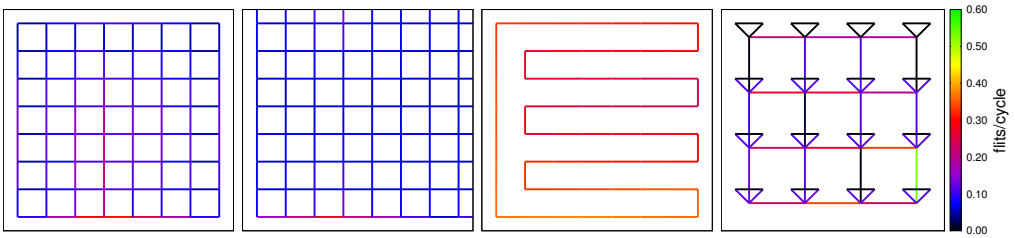


(a) `Canneal`, Mesh **(b)** `Canneal`, Torus **(c)** `Canneal`, Ring **(d)** `Canneal`, CMESH

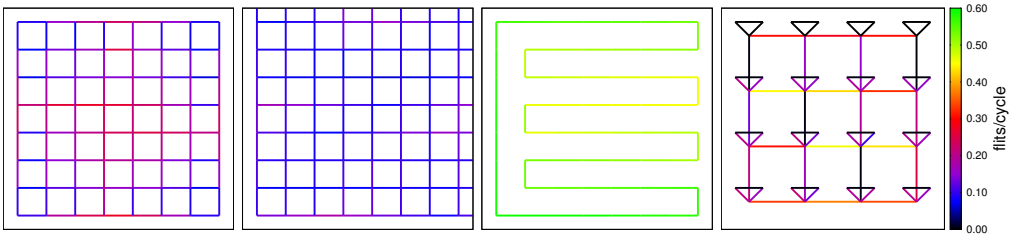


(e) Multi, Mesh **(f)** Multi, Torus **(g)** Multi, Ring **(h)** Multi, CMESH

Figure 3.6: Injected flits per cycle and node for the `canneal` parallel application (top) and a multiprogrammed mix (bottom) executed in 64 cores.



(a) `Canneal`, Mesh **(b)** `Canneal`, Torus **(c)** `Canneal`, Ring **(d)** `Canneal`, CMESH



(e) Multi, Mesh **(f)** Multi, Torus **(g)** Multi, Ring **(h)** Multi, CMESH

Figure 3.7: Link utilization in flits per cycle for the `canneal` parallel application (top) and the multiprogrammed mix (bottom) executed in 64 cores. Each line is the combination of two links, one in each direction. Injection and ejection links have been left out. In the torus, links that touch the edges of the chip represent the wraparound links. In the CMESH, lines that make up triangles are connections between the local and global routers of each cluster of cores. Note that scales are different.

ing all applications executed on single-threaded architectures, nodes in parallel and multiprogrammed workloads inject an average of 0.024 and 0.052 flits per cycle, respectively; for our 16 multi-threaded core configuration, they inject 0.11 and 0.23 flits per cycle on average. For comparison, Dally shows that the saturation throughput of an 8x8 mesh with four 1-flit virtual channels is around 0.5 and 0.6 [34], and that value would be even lighter with larger VCs. Since the network is lightly loaded, congestion delays are not a major contributor to network latency. Even on the multithreaded configurations where there is more traffic in the interconnect, we still see the same relative performance among the topologies, pointing out the paramount importance of the network diameter. This explains why the concentrated topologies reduce network distance without a significant increment on network contention.

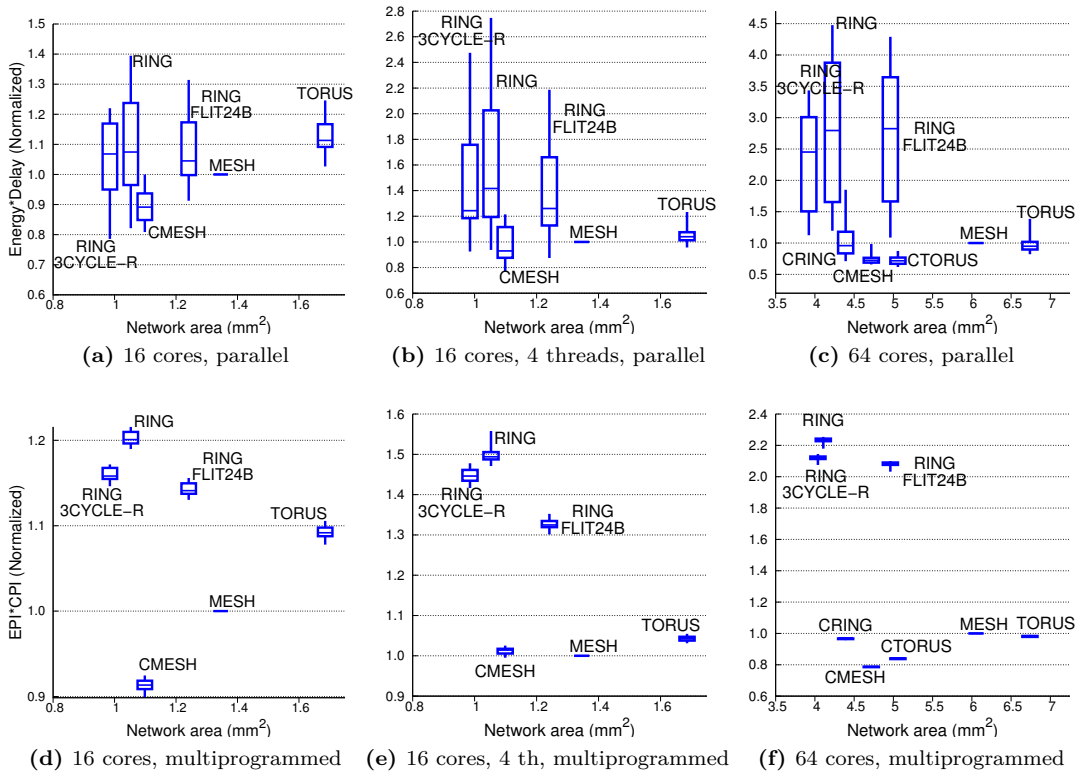
Our results show that real workloads exhibit non-uniform traffic patterns across the network, even though this cannot be perceived when considering only average statistics. Instead of the fairly uniform traffic distributions seen with synthetic networking workloads, we observe hotspots in some locations. This points out that synthetic traffic patterns should have hotspots in both flit injection and destination distribution in order to reflect the real traffic load imposed on the network by parallel and multiprogrammed workloads. They also show the potential of fine-grained network reconfiguration for real applications. For instance in the form of dynamic resource allocation or frequency/voltage scaling (DVFS), where some parts of network save power while others increase execution speed. Recent studies also confirm this potential: Lee *et al.* use DVFS for thermal management in 3D ICs, both in the cores and routers, but they do not consider parallel applications [84]; Haghbayan *et al.* also propose DVFS control to honour power and thermal constraints in a dark silicon context, but exclude the network from such control and consider a synthetic task generation model instead of real workloads [126]. Reconfiguration is beyond the scope of this work but our data adds experimental evidence to its great interest.

3.4.5. Area, Energy, and Delay

When making design choices for future architectures we need to consider performance, power, and area. For parallel applications, we calculate $\text{Energy}_{Network}$*$\text{Delay}_{ParallelSection}$ (ED); for multiprogrammed workloads, where we simulate a constant number of cycles, we use EPI*CPI (EPI=$\text{Energy}_{Network}$ per Instruction, CPI=Cycles per Instruction). Figure 3.8 depicts network area (as reported by DSENT) versus ED or EPI*CPI normalized to the mesh. To display the variance across the parallel applications and the multiprogrammed mixes, we represent the results with candlesticks. Ideally, we would like our configuration to be in the bottom left corner of the graphs.

For 16 single-threaded cores (plots 8a and 8d), the CMESH offers the lowest values for energy and delay, with a small area (only 8% bigger than the ring and 18% and 35% smaller than the mesh and torus, respectively). For 16 multi-threaded cores (plots 8b and 8e) and, especially for 64 cores (plots 8c and 8f), the ED and EPI*CPI increase substantially for the ring topologies with all workloads. The Delay contributor increases much more significantly with more cores due to the higher hop count. Therefore, networks with lower diameter perform better when integrating a larger number of cores. In this case, the CMESH still offers the best trade-offs. We also see that the variance across the multiprogrammed mixes is very small, pointing out that the way of distributing independent applications in the chip does not impact either performance or network energy. Our results show that over-dimensioning the network is not the best solution: a simple topology like the CRING is better than the torus from all standpoints. Even in the multithreaded architecture (plots 8b and 8e), where the network has a higher load, the CMESH still offers a better trade-off than the torus.

We also see that the deviation of the results varies among topologies and is bigger with 64 cores. It is proportional to the variation in network latency, which increases with the average distance of the network and hop latency. This is because the Delay component of the ED product suffers bigger increases in certain applications where the thread distribution generates disadvantageous traffic patterns for the ring topology.



(a) 16 cores, parallel **(b)** 16 cores, 4 threads, parallel **(c)** 64 cores, parallel

(d) 16 cores, multiprogrammed **(e)** 16 cores, 4 th, multiprogrammed **(f)** 64 cores, multiprogrammed

Figure 3.8: Area versus Energy*Delay for the parallel applications (top) and EPI*CPI for the multiprogrammed workloads (bottom) for 16 single and multithreaded cores and 64 single-threaded cores, normalized to the mesh. The RING and RING_3CYCLE_R have the same area, but candlesticks have been slightly shifted on the horizontal axis for better visualization, both have an area of $1.0mm^2$ for 16 cores and $4.1mm^2$ for 64 cores.

3.4.6. Fairness

In a multiprogrammed environment, fairness determines if resources are evenly distributed among independent applications. A system is fair if all the multiprogrammed applications experience an equal slowdown compared to their performance when executed alone. Our interest lies in assessing whether the topology influences fairness. To numerically quantify fairness, we rely on the following formula:

$$fairness = \frac{\min\limits_{i}\left(\frac{CPI_i^{MT}}{CPI_i^{ST}}\right)}{\max\limits_{i}\left(\frac{CPI_i^{MT}}{CPI_i^{ST}}\right)}$$

where CPI, ST, and MT refer to cycles per instruction, single thread, and multi-threaded execution, respectively [42]. The i index refers to the applications. The ideal value would be 1; the closer we are to it, the better fairness we have. To calculate the fairness, we take one of the mixes and simulate both the mix and each application running alone in chip, pinned to the same core in both cases.

Figure 3.9 shows the fairness for all the topologies on 16 single and multithreaded cores, and 64 single-threaded cores according to that formula. In order to evaluate if there are any outlier numbers that are negatively affecting the final results, we present in Figure 3.10 candlesticks with the ratio between the CPI of the applications executed along with the rest of applications (CPI-MT) and the CPI of the same application running alone in the chip (CPI-ST). In this case, fairer configurations will present short candlesticks, while unfair networks will have big and long candlesticks. The same conclusions can be extracted from both graphs. With lightly loaded networks, fairness is very similar across all topologies. It significantly decreases in two cases: for the multithreaded cores respect to the single-threaded configurations, and for the ring topologies respect to the others with 16 multithreaded cores. These reductions correspond to cases where the network supports a higher load. In those situations, there is

more congestion on the network, and that has a higher impact on the applications that need to use it more often, while others with more hits on their first level cache remain undisturbed. In the multithreaded cores, we clearly notice a correlation between performance and fairness: the mesh, torus, and CMESH show the highest fairness because more efficient networks can successfully support more simultaneous communications without interferences.

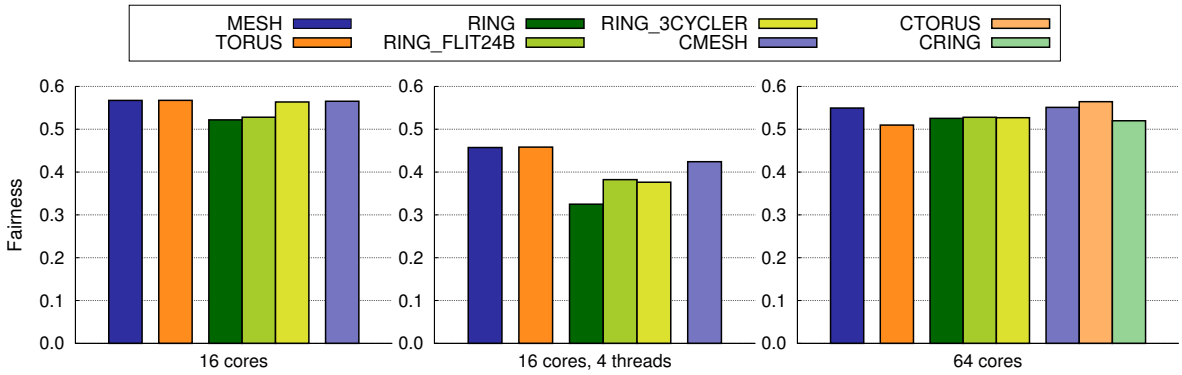
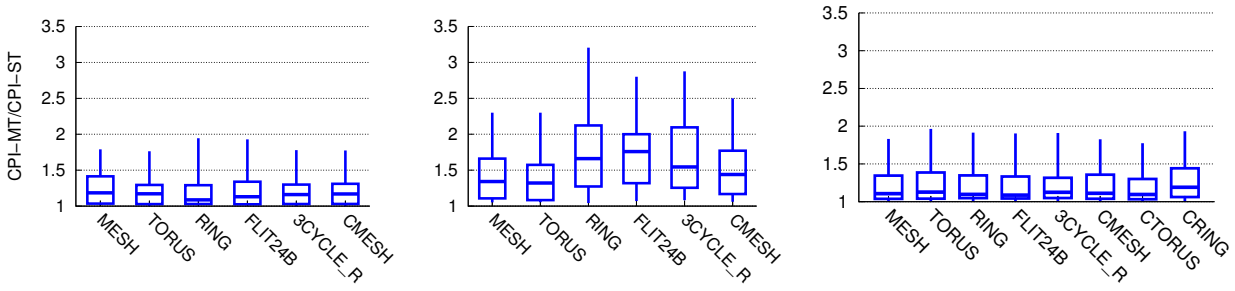


Figure 3.9: Fairness for the multiprogrammed workloads in chips with 16 single and multithreaded cores and 64 single-threaded cores. The higher the bars, the better.



(a) 16 cores **(b)** 16 cores, 4 threads **(c)** 64 cores

Figure 3.10: Fairness for the multiprogrammed workloads in chips with 16 single and multithreaded cores and 64 single-threaded cores, represented by the ratio of the CPI of the applications executed along with the rest of applications (CPI-MT) divided by the CPI of the same application running alone in the chip (CPI-ST). We present candlesticks, where we can see the minimum, maximum, median, lower quartile (Q1), and upper quartile (Q3).

3.4.7. Memory Controller Placement

In the previous sections, all configurations had four memory controllers located at the edges of the chip. It has been demonstrated that the location of the memory controllers impacts memory latency in a mesh topology [5]. We have compared several options varying the number and placement of memory controllers to look for the best configuration in terms of performance. For the sake of brevity, this section focuses on multiprogrammed workloads because they exhibit higher main memory access rate. Also, we limit the design space to the mesh and CMESH topologies, because they are the ones that offer the best performance, energy, and area trade-offs (as we showed in section 3.4.5), and to the 64-core chip, where distances are longer and MC placement has a larger impact. Figure 3.11 shows the nodes of the chip where the memory controllers are located for the mesh and the CMESH topologies. We test 9 configurations for the mesh, with 4, 8, and 16 memory controllers; for the CMESH we test 5 configurations with 4, 8, and 16 controllers. In the CMESH, the MCs are connected directly to the global routers (see Section 3.4), so we divide the chip in only 16 squares, which represent clusters of 4 cores each.

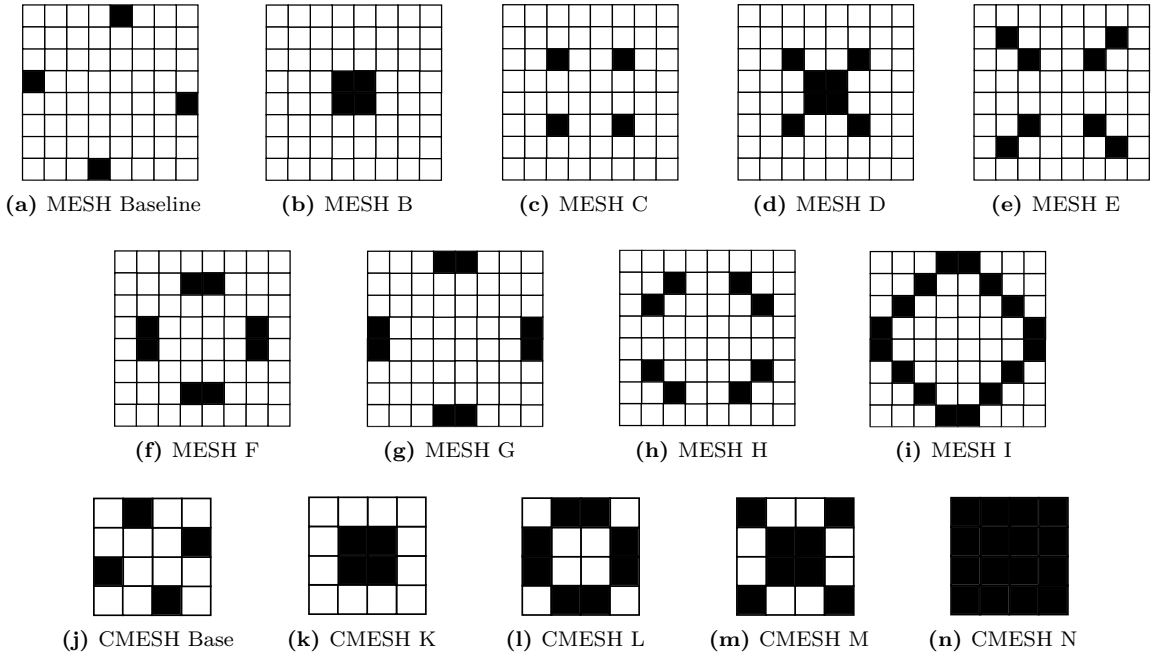


Figure 3.11: Memory controller configurations for the mesh -(a) to (i)- and CMESH -(j) to (n)- topologies with 64 cores. The shaded tiles include a memory controller. For the CMESH, MCs are connected to the global routers so we only represent the 4-core clusters of the chip.

We calculate the average performance of the multiprogrammed workloads with all the memory controller configurations, and see that variations in performance are so small (always smaller than 0.03%) that both the number and placement of memory controllers seem to be a secondary issue. This is because benchmarks do not miss in the L2 very often, even for the multiprogrammed workloads, which is where we detected the largest amount of traffic to memory. Increasing the number of memory controllers does not have a significant impact on performance either.

Abts *et al.* state that memory controller placement is critical and that a good placement reduces contention, lowers network latency, and provides predictable performance [5]. Although they simulate different workloads than we do and with smaller caches, we have compared the amount of traffic to memory they have to what we see in our simulations. We have determined that our multiprogrammed workloads access main memory more often than their applications, increasing the effect of memory controller placement on overall performance. It is true that some memory controller placements reduce network latency, but we go a step further and guarantee that the impact on system performance is negligible for such light traffic to memory.

Our main objective was to determine if the memory controller placement is relevant for this kind of general purpose CMPs with these applications, which represent reasonable workloads. However, it is also interesting to determine if the memory controllers could significantly affect performance on a scenario with more traffic to memory. In order to analyse this, we repeat the experiments with different system parameters: reduced L2 cache size to increase the traffic to memory (down to 128 KB per core from the original 1 MB per core), and faster memory access to increase the impact of the network latency reduction (50 cycles instead of the baseline 150). These new simulations increase the traffic to memory by 50%, but the effect of the memory controller placement is still small. In average, the difference in performance with respect to the baseline is 0.6%, which is much more than what we were seeing before, but still very small. Therefore, we conclude that the memory controller placement does not have a relevant impact on performance in our general purpose CMP, even with system parameters that enlarge the impact of the network on memory access time. In any case, our results do not rule out the importance of this issue in specific situations, such as streaming memory applications or systems with even smaller caches.

3.5. Concluding Remarks

Considering the interconnection network and the cache hierarchy simultaneously helps identify improvement opportunities in the design of CMPs. Both elements have a significant influence on system performance, area, and power consumption. We have modelled in detail the processors, memory hierarchy, and network using full-system simulation and executing both parallel and multiprogrammed workloads. We have performed a qualitative and quantitative analysis of three network topologies: mesh, torus, and ring, including two additional ring configurations (one with more bandwidth and one with 3-cycle routers) and concentrated networks for CMPs with 16 single and multithreaded cores and 64 single-threaded cores.

Our results show that performance is highly affected by the choice of the interconnect, especially in 64-core systems, where the ring performance drops by 72% with respect to the CMESH for parallel workloads. The ring topologies perform worse due to the increased hop count, which translates into higher network latency. In average, compared with the CMESH, the ring suffers from a 34% network latency increase in the single-threaded 16-core chip, 60% in the multi-threaded 16-core chip, and 136% in the 64-core chip. The CMESH topology offers the best performance with low energy consumption (17% less than the torus for 64 cores) and area (30% smaller than the torus for 64 cores) for all workloads considered and both 16 and 64-core chips, even with multithreaded cores, which generate a heavier traffic load.

We have reported that in real applications traffic is very light and not uniformly distributed, pointing out the potential of heterogeneity, either in the form of dynamic resource allocation or frequency/voltage scaling. For parallel applications, both the injection rate and the message destinations are more variable than those we see with synthetic traffic patterns, with only 2 nodes injecting an average of 33% of the traffic in the 64-core chips; for multiprogrammed workloads, traffic is random with hotspots at the memory controllers, which inject 25% of the traffic.

For multiprogrammed workloads, we have concluded that that contention in the network causes fairness to drop, especially for networks with lower performance. For example, the mesh has 26% lower fairnes with 16 single-threaded cores than with 16 multithreaded cores; focusing only on the multithreaded cores, the ring has 28% lower fairness than the mesh. We have also determined that the placement and the number of memory controllers has a negligible effect on system performance with realistic applications, because they have limited memory access.

Chapter 4
Reactive Circuits: Dynamic Construction of Circuits for Reactive Traffic in Homogeneous CMPs

Summary

Considering the characteristics of the memory subsystem while designing the NoC helps identify opportunities to build more efficient designs. The memory subsystem must ensure a coherent memory state, which enforces the use of request-reply message patterns. Leveraging these frequent request-reply patterns, we extend the functionality of requests and use them to dynamically build a circuit for their replies with the objective of reducing their network latency. Starting from this simple idea, which we denote Reactive Circuits, we evaluate several implementations of the mechanism: with and without sharing circuits between messages, performing timed reservations, and removing the implicit coherence messages. A careful implementation of this circuit reservation mechanism in a wormhole router achieves an average 20.8% reduction in network energy consumption, 5.8% smaller router area and a 4.8% system performance increase in a 64-core chip, compared with a conventional network.

4.1. Introduction

We focus on optimizing the network on-chip by customizing it for the traffic it has to support, keeping in mind that the most relevant metric we must take care of is latency as we demonstrated in Chapter 3. The traffic is generated by the coherence protocol to transport information among caches located in different nodes, and mostly consists of data requests and replies. This Chapter introduces Reactive Circuits, a NoC design that takes advantage of this reactive nature of traffic. This novel design leverages the predictable network traffic behaviour to dynamically build virtual circuits for replies. We evaluate it in a single-thread homogeneous chip multiprocessor with 16 and 64 cores connected by a mesh.

Analysing the coherence protocol in a standard wormhole 4-stage pipeline router, we note that the request-reply pattern dominates over the rest. Table 4.1 shows the relative amount of messages travelling through the network, classified into requests and reply types (detailed information about the messages generated by the coherence protocol can be found in Table 2.2). More than half of the messages are a reply to another message and, therefore, we know their source and destination before the message is injected into the network. With this information, routers can reserve in advance crossbar path and output virtual channel, removing those stages from the critical path of the router pipeline. To hide the circuit reservation latency, we use the requests to make the reservation at every router in the path. We also observed that the network is lightly loaded (nodes inject, in average, less than four flits every 100 cycles), suggesting it will be feasible to keep resources busy for long periods of time. Nevertheless, we also include a version that optimistically calculates when the circuit will be needed and reserves only that timeslot.

Reactive Circuits aims to improve state-of-the-art low-latency NoCs by making the following contributions:

- Implementing a circuit reservation technique with several versions, removing routing and arbitration latency from the router pipeline, and significantly reducing network latency.
- Reserving the circuit on-demand, but hiding the reservation latency with the data request and without any extra circuit setup message or setup network.
- Removing unnecessary buffering from the virtual channel reserved for circuit construction reducing router area and static power.
- Eliminating some acknowledgement messages that are used to guarantee coherence but are no longer needed when data travels through a reserved circuit.

We simulate the mechanism with 16 and 64-core chips using a full-system simulator with realistic workloads. Reactive Circuits reduce network latency, static power and router area. These results emphasize the importance of considering the system as a whole and studying how all the elements interact with each other [78, 133].

4.2. State-of-the-Art

Several works have proposed hybrid packet-circuit switching techniques to speed up certain messages. Some proposals suggest separate networks for packet and circuit switched messages: Palumbo *et al.* determine if messages will use the packet or the circuit switched channels depending on their size [119]; Duato *et al.* decide at compilation time whether a circuit between two nodes should be established based on expected communication patterns [37]; Abousamra *et al.* use the requests to reserve circuits for the replies based on estimates of circuit utilization times [3]; Abousamra *et al.* send a circuit reservation request as soon as a cache hit is detected, though this may not be enough to completely hide the circuit reservation latency [4]. Other authors implement a single network that supports both packet and circuit switching: Enright *et al.* build circuits on demand and undo them when they conflict with another circuit [61]; Abousamra *et al.* configure circuits periodically based on online communication statistics [2]; Kline *et al.* reserve circuits on demand so that flits can traverse multiple hops in a single cycle [71]; Mazloumi *et al.* reserve a circuit with the request and activate it with a probe message

Table 4.1: Percentage of messages that traverse the network (average for all benchmarks executed in a 64-core chip).

Requests	Replies		
47.0%	53.0%	**L2_Replies** Data from L2 to L1	22.6%
		L1_DATA_ACK. L2 acknowledges data reception	23.0%
		L2_WB_ACK. L1 acknowledges write-back reception	4.7%
		L1_INV_ACK. Invalidation acknowledgement	1.1%
		MEMORY. Data from main memory	0.9%
		L1_TO_L1. Data from L1 to L1	0.7%

when the reply is ready [96]; Liu *et al.* speed up circuit setup in TDM NoCs by sending parallel probes [87].

A different technique preallocates resources in advance to allow faster data transmission, either sending control flits through specialized networks or with tokens that inform neighbouring nodes about their buffer availability [123, 44, 83].

Most of those mechanisms establish circuits between nodes using dedicated networks or links [61, 71, 96, 87] or at least need to send specific setup messages [119, 37, 4], and many of them need to wait for the circuit setup delay [44, 87]. One of them introduces complexity at the network interfaces by forcing them to keep communication statistics [2]. The proposals most similar to our Reactive Circuits mechanism are [3, 96], using the request to reserve circuits for the reply, but they do not go as far as removing buffers to reduce router static power and area or eliminating unnecessary coherence messages. [3] also does an estimation of the time when the circuit will be needed, but does not use it to avoid circuit conflicts.

Another common approach to reduce network latency involves routers that speculate by using paths without prior reservation, which only work if there is no contention [102, 101, 122, 76]. These routers are more complex and may require reduced network frequency or result in energy and performance penalties when the implemented shortcuts cannot be used. There are also several proposals that radically modify the routers to eliminate all buffers and reduce per-hop latency to one or two cycles, but suffer big penalties with any level of congestion [69, 98].

Many of the publications we have mentioned do not perform full-system simulations with real traffic, and therefore, are unable to capture the effect of realistic traffic patterns on their proposals [119, 37, 96, 87, 123, 44, 83, 102, 101, 122, 76, 69, 98].

The Reactive Circuits proposed in this paper uses some of the same concepts already introduced by other publications, but combines them to implement an optimized mechanism that does not require extra networks, additional management messages, gathering statistics, or modifying the coherence protocol. We leverage the memory hierarchy behaviour to efficiently reserve network resources in advance with minimal changes to the routers, and completely hiding circuit reservation delay.

4.3. Setup, Operation, and Release of Reactive Circuits

This section describes the details of the Reactive Circuits mechanism for each of the implemented versions: reserving, using, and undoing fragmented and complete circuits, sharing circuits, eliminating coherence messages, and building timed circuits.

4.3.1. Reserving Reactive Circuits

As discussed before, when a request reaches its destination, we already know that a reply is going to be sent back to the source, based on the patterns introduced in Table 2.2. A previous approach consists of sending the head of the message in advance to reserve the resources along the way, building a circuit for the data in parallel with the L2 access [4]. However, in our case the L2 hit access is too fast (7 cycles) compared with the average time needed to set up the circuit (19 cycles in a 16-core chip and 59 in a 64-core system). We overcome this problem by reserving reactive circuit for the reply as its request travels towards the destination. Using dimension order routing (DOR), request and reply follow disjoint paths because both messages travel first in the horizontal direction, and then in the vertical one. In consequence, we start by modifying the DOR algorithm so that requests and replies use XY and YX routing, respectively, so the path to and from the destination match. This change does not generate deadlocks because different message types use different virtual networks.

Requests go through the four original stages of the router (see Table 2.3). In parallel with VC allocation, the reactive circuit is built for the reply. During that process, the necessary information to identify the circuit is stored in the router (requestor identifier and cache line address). Since we have two VCs for replies, we dedicate one to circuits and leave the other for replies that do not have a circuit. This way, Reactive Circuit routers will support packet and circuit switching simultaneously. Information of the circuit is also stored in the network interface where the circuit starts.

Out of the message types in Tables 4.1 and 2.2, reactive circuits are built for data sent from L2 to L1 (L2_Replies), replacement acknowledgements (L1_WB_ACK), and main memory replies (MEMORY), which account for 53.2% of all reply messages. Invalidation acknowledgements (L1_INV_ACK) and direct data transfers between L1 caches (L1_TO_L1) are a very small percentage of the reply messages (only 1.8%). L1_DATA_ACK messages are replies sent from an L1 to an L2 after a request-reply communication to confirm the reception of the DATA. These messages are essential to maintain coherence as they guarantee the L2 that the data has been received and prevent race conditions, and are used both in academia [125, 40] and industry, e.g. AMD Opteron [31]. L1_DATA_ACK messages do not follow the same path as the request and reply between L1 and L2: the request follows the XY path from the L1 to the L2 and the reply follows the YX path from L2 to L1 (so it goes through the same routers as the request), but the ACK is a reply that follows the YX path from L1 to L2. Therefore, it is not possible to use a previous message to build a circuit for them.

The following sections describe specific details of the implementation and introduce the different versions of the Reactive Circuits mechanism.

4.3.2. Fragmented versus Complete Circuits

When trying to build a circuit at a router, the necessary resources might not be available. In this situation, there are two design alternatives:

- Allow *fragmented circuits*, keeping the partial path reserved, and attempt to reserve the rest of the path after the next hop.
- Support only *complete circuits*, so that any lack of resources will force us to undo the previous reservations.

With *fragmented circuits*, we need to assure messages can always be stored in the router in case their circuit has not been completely built. As we already mentioned, we start by dedicating one VC for replies without a circuit, and the other one for replies with circuit. In the baseline NoC, VCs are not heavily used and are rarely blocked. However, keeping VCs reserved for a longer period of time has a negative effect: there may not be enough resources to exploit the full potential of the proposal. Therefore, with fragmented circuits we include an additional VC to increase the number of simultaneous circuits, ending up with a total of three VCs in the reply virtual network. This extra VC will mean an increment in router power and area.

Building only *complete circuits* allows us to implement many simplifications in the router. We guarantee that a message with a circuit has all the resources it needs from source to destination. Hence, it will never get blocked in the network. This has two beneficial effects: first, it allows us to remove the buffer storage of the VC dedicated for circuits reducing the router area; second, we can build as many circuits as we want for that VC in every input port because flits will just go through the router without stopping.

All the complete circuits in the same input port of a router must have the same source to avoid conflicts: two circuits with different input ports and the same output port cannot be built at the same time on a router. This is because if two flits arrived at the same time wanting to use those circuits, one would have to be dropped because both of them would not be able to leave through the same output port in the same cycle. We have experimentally explored the best number of simultaneous circuits built per input and set it to five. This number reduces the probability of failing to build a circuit due to lack of free storage for circuit information, but it is small enough to minimize area and power consumption, as we will demonstrate in the evaluation section. Figure 4.1 presents the modified router that implements the reservation of complete circuits.

To clarify the difference between the two alternatives, we show an example of how circuits are built in Figure 4.2. To simplify the example, we assume there is one single VC dedicated to circuits. In both cases, there is a blue circuit already built from L2A to L1A, and a new request tries to build a new circuit (green) from L2B to L1B. The request builds the circuit as it is traversing the network, so the circuit is built starting from its final router (L1B) and ending in its first router (L2B), in the opposite direction of the replies that will use it. In Figure 4.2a, there is one hop in the network where a conflict is detected (in the router marked as R2), but the fragmented circuit can be built in all the other routers. In Figure 4.2b we see that the situation is very different with complete circuits. When the request tries to build the circuit in router R1, it detects there is already another circuit using the needed input port (East port). That circuit that is already completely reserved has a different source than the one we are trying to build now, which means that at some point we will need two circuits with different inputs and the same output in a router (in router R2). Therefore, this circuit cannot be reserved in all the routers and, since this mechanism only supports complete circuits, the successful reservations in the downstream routers have to be undone.

4.3.3. Using the Circuits

When a reply arrives at a router, it checks if there is a circuit built for it. In that case, it can go straight through the crossbar leaving the router in just one cycle. When the tail flit of the message leaves the router, it frees the circuit resources by clearing the B bit. With fragmented circuits, a message that had a circuit might arrive at a router where there is no built circuit. When that happens, it will just be stored in the VC and go through the usual four stages of the router.

Even when there is a circuit built at a router, the ports and links involved can still be used by other messages. The crossbar prioritises messages with a circuit, but it grants access to the other virtual channels when the circuit is not used.

4.3.4. Undoing circuits before they are used

We must undo a circuit before it gets used under the following situations:

- The coherence protocol lets an L2 cache bank forward a request to an L1 that owns a cache line exclusively, who will supply the data directly. Therefore, the circuit built between the requestor L1 and the L2 bank will never be used and should be explicitly torn down.
- When we try to build a complete circuit and reach a router where resources are not available, we have to undo the section of the circuit we had successfully built so far (see Figure 4.2b).

In both those situations, we undo the circuit with a simple and efficient technique: we send the data of the circuit to be undone towards the circuit destination using credits. If a credit had to be sent at the

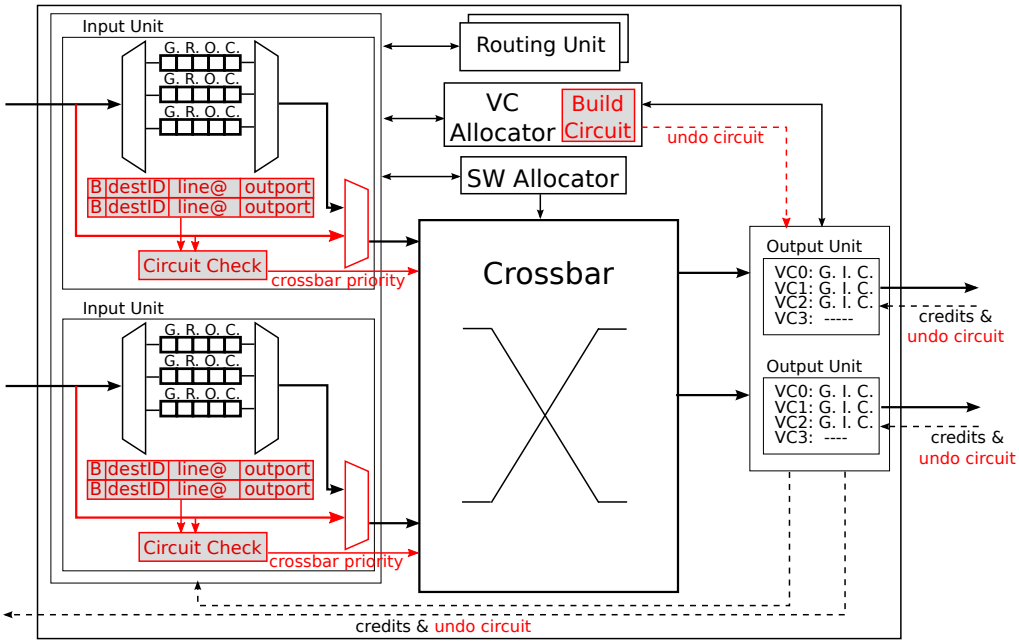
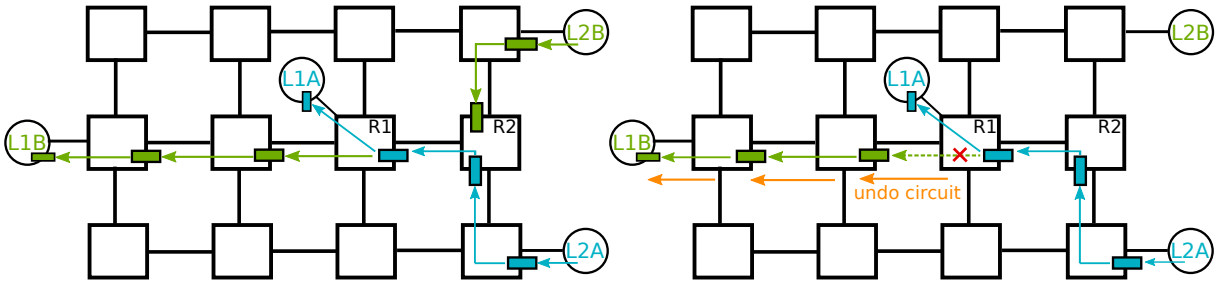


Figure 4.1: Architecture of the router that reserves complete Reactive Circuits. The modifications with respect to the baseline router are highlighted. They include "Circuit Check" logic at the Input Units and a "Build Circuit" module in the VC allocator. In this drawing, two simultaneous circuits can be built per input port. VCs at the Input Units store global state (G), route (R), output VC (O) and credit count (C). Circuit information includes built-circuit bit (B), destination identifier (destID), cache line address (block@) and output port (outport). Credits may carry undo-circuit information. The Output Units store global state (G), input VC (I) and credit count (C).



(a) Fragmented circuits **(b)** Complete circuits

Figure 4.2: Example of how circuits are built with fragmented and complete circuits, using always one single VC for circuits. In both cases, the blue circuit has already been built by a request going from L1A to L2A. Then another request has tried to build the green circuit from L2B to L1B.

same time to free a buffer, we piggyback the information; otherwise, we send a specific credit.

We also considered undoing circuits when an L2 miss occurs, because resources will be held for a long time while the request goes to main memory. However, simulation results show better performance if we keep them built.

4.3.5. Reusing Complete Circuits

In the previous sections, circuits were specifically built for a message and used only by that message. We go a step further to improve our mechanism and try to find other messages that can reuse the circuits. When a reply that does not have a built circuit is about to leave the network interface, it checks if there is any circuit starting at that NI that it could use to get closer to its destination. In that case, the message becomes a *scrounger* message that uses the circuit to reach an intermediate destination. At that point,

the network interface will forward the message by re-injecting into the network that it can arrive at its final destination.

Note that we can only apply this method with complete circuits because there are no buffer guarantees for two messages using the same fragmented circuit.

4.3.6. Eliminating Coherence Messages

Studying the coherence protocol while designing the NoC has allowed us to notice a rewarding effect of our reactive complete circuits. We already mentioned that we cannot build a circuit for the L1_DATA_ACK reply messages that are sent from L1 to L2 after the L2_Reply (see Section 4.3.1). Since the NoC does not guarantee message ordering, this L2_DATA_ACK avoids a situation where the L1 received an invalidation or a forwarded request for data it had not received yet. However, if the L2_Reply uses a complete circuit to get to the L1 requestor, we are sure the data, travelling at a speed of 2 cycles per hop and never blocking, will arrive before any other message sent from L2 to L1 afterwards. Therefore, we can acknowledge the data reception to the L2 without the need to wait for the L1_DATA_ACK message. With this simple observation, we can omit those messages to reduce contention in the network and energy consumption. On top of that, other requests waiting to access the same cache line will reduce their waiting time since the L2 cache line will not be blocked while the L2_Reply and L1_DATA_ACK messages are exchanged.

4.3.7. Timed Reservation of Complete Circuits

Having complete circuits is the most beneficial option to reduce router power and area. However, circuits cannot be built when there is a conflict, which means there cannot be two circuits with different input ports and same output port built simultaneously in a router. If two flits arrived at the router at the same time wanting to leave through the same port, one would be forced to wait and we would not be able to store it because we have removed the buffers for that virtual channel. However, given the light load of the network, it is very improbable for those two flits to arrive at the same time and create a real conflict.

Therefore, circuits cannot be built due to the possibility of a collision that may not actually happen. To avoid that, we implement timed circuit reservation: we optimistically calculate when the reply will go through the router, and reserve the circuit only for those cycles. This way, the channel is not busy from the moment it is reserved until it is used, it will only be busy for a short time interval. The time will be calculated using the number hops between the current router and the destination, the hop latency for the request (five cycles/hop) and for the reply (two cycles/hop), and the cache hit latency [3]. It will then be stored in two counters where we will annotate the cycles until the circuit reservation starts and finishes, and that will be decreased every cycle. Abousamra *et al.* also calculate the expected reply arrival time but use it only order the circuit reservations, stating that timed based reservations are impractical due to unforeseen delays. We address this issue by enhancing the basic idea with three variations of complete timed circuits, as explained below.

Now, circuits with different input port and same output port can be built at the same time, as long as they use non-conflicting time slots. When a reply is going to be sent, it can only use its circuit if it is within the optimistic timing estimation. Otherwise (for example in case of a cache miss), the circuit will be undone and the reply will need to go through all the stages of the router. Figure 4.3 describes how timed circuits are reserved and includes three variations designed to increase the flexibility:

1. Reserve the circuits with *slack*. Instead of reserving the exact number of cycles the reply will need, we give the option to reserve more cycles to be able to accommodate delays due to failed arbitrations of the request and extra cache delays.

2. Allow reserving the circuits with *delay*. If the time slot the circuit needs has already been occupied, we try to reserve the circuit for some cycles later. The reply may need to wait for its time slot before being sent, but it will reach its destination faster by using the circuit. Note that this version must

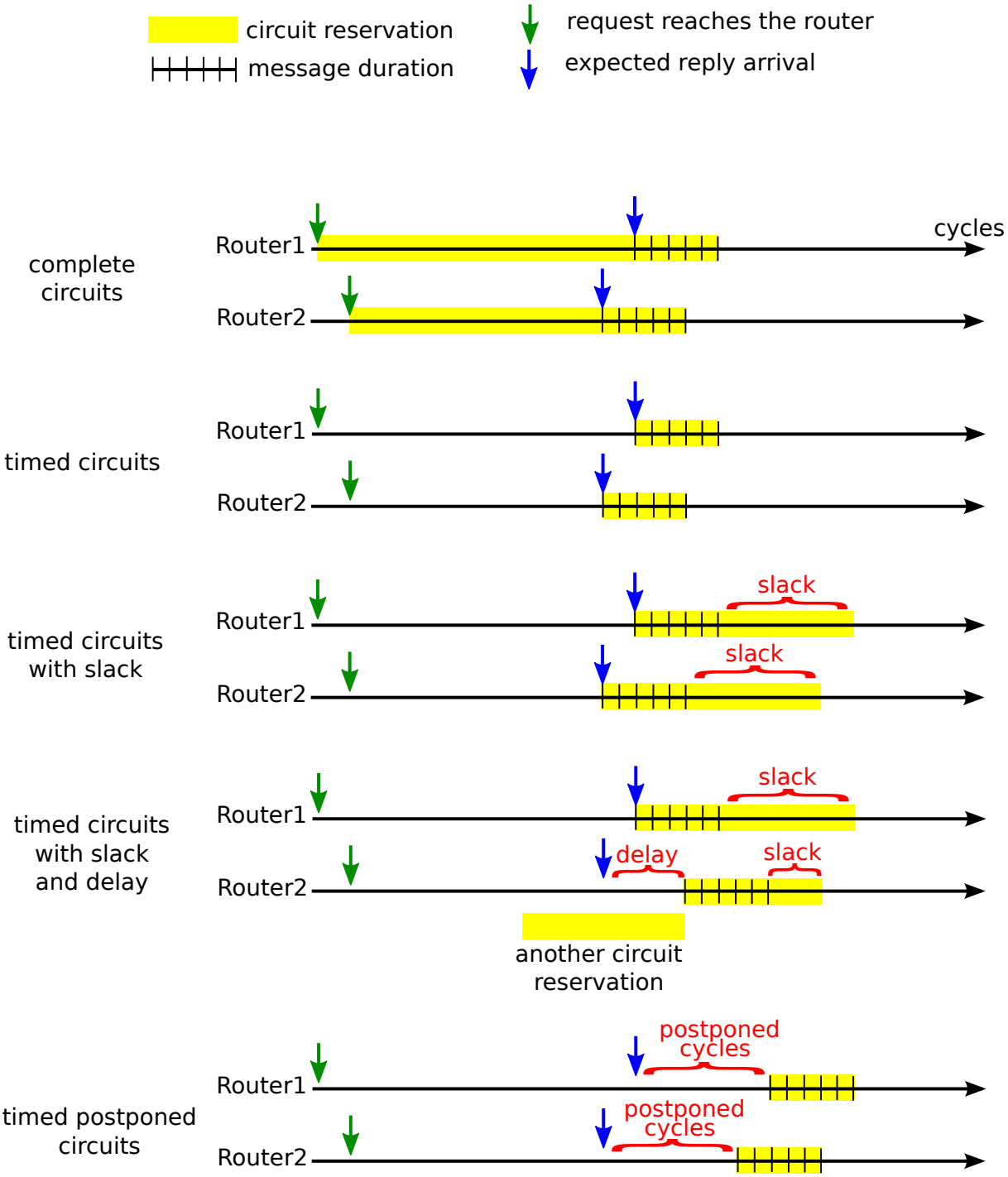


Figure 4.3: Diagram for reactive circuit reservation in the four variants of complete timed circuits. Basic complete circuits are also included for comparison. In each configuration, the construction of the circuit is shown for two consecutive routers.

be combined with the previous one: we need to reserve the timeslot with a slack so that we can introduce a delay and still be on time for the reservations already made in previous routers.

3. To have the flexibility of the slack without reserving the circuit for a longer period of time (which increases the probability of conflicts), we reserve *postponed* circuits. In this case, we reserve the circuit for the exact number of cycles it needs, but for a later time. This will increase the number of circuits that can be built and used, but all the replies will need to wait for the circuit, even if the request was not delayed and they were ready before.

In the three versions, the number of cycles of slack, delay or postponement is proportional to the path length, introduced as number of cycles per hop.

4.3.8. Ideal Circuit Reservation

We consider an ideal version of the mechanism that will successfully reserve and use all the circuits. This version is not a feasible design due to the increased area and power consumption, and the inclusion of logic that would not fit in a single cycle, but we include it as an upper bound for performance comparison. It consists of keeping the buffers and reserving all the circuits, without caring about conflicts or timing, and without a limit in number of circuits per input port. Then, all the replies will use their circuit to reach their destination. At every hop, the router needs to check if there are two conflicting flits using circuits, and in that case, prioritise one of them and keep the other in the buffer. That is done in a single cycle, as well as checking if the circuit has credits for the next hop before forwarding the flit. We would not be able to implement this in a real system, but all the replies will use circuits and suffer only small delays if there are collisions, which will give us the best performance Reactive Circuits can offer and will be useful for reference.

4.4. Evaluation

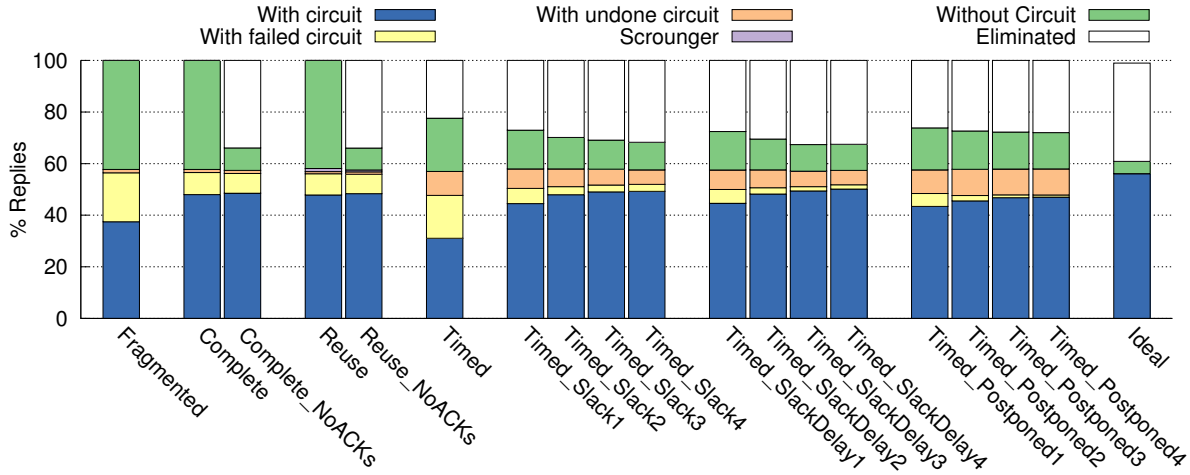
This section presents the main results for the Reactive Circuits techniques, including power, area, and performance.

4.4.1. Construction and use of Reactive Circuits

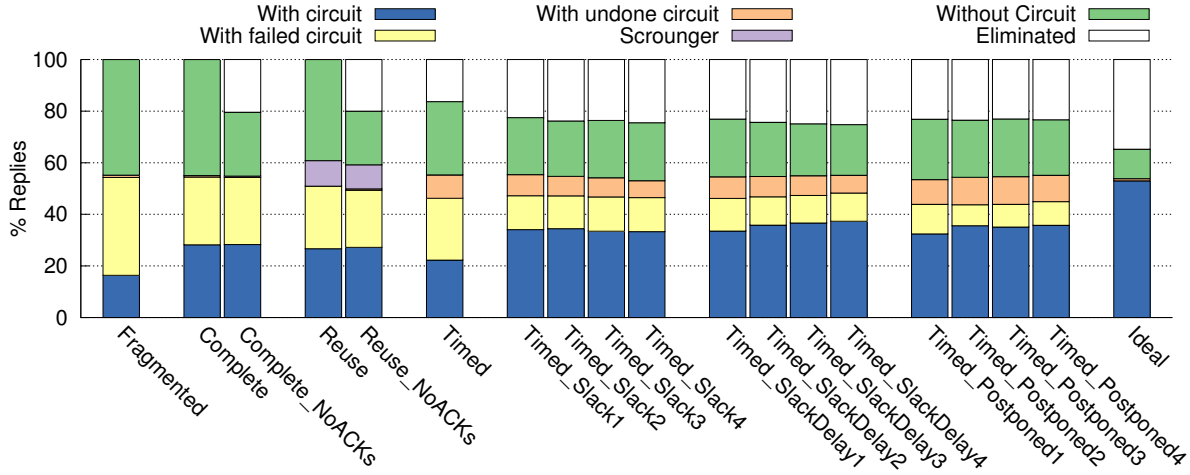
We analyse how effective each version of our mechanism is in building and using circuits. Figure 4.4 presents the percentage of replies that travel on a circuit, with a failed circuit (could not be completely built), with an undone circuit (it was completely built but had to be undone), that travel on a circuit built for another message (scrounger messages), that were not eligible for a circuit, and that were eliminated (removed L1_DATA_ACKs due to successful L2 to L1 circuits). It includes every circuit-building configuration tested in 16 (Figure 4.4a) and 64-core chips (Figure 4.4b), and we present the average of all the parallel applications and one multiprogrammed mix.

The first bar of the graph corresponds to fragmented circuits. In this case, the failed circuits are those that could not be completely built, but replies using them will still have sections of their path with a built circuit. As we already anticipated in Section 4.3.4, there are some cases when a built circuit will not be used due to the behaviour of the coherence protocol (when the L2 bank forwards the request to the L1 owner). However, this is a very small percentage of the total of replies. Apart from that, there are more than 40% of replies that cannot benefit from the mechanism because they are not associated with a request that can reserve the circuit.

The rest of the bars are different versions of complete reactive circuits. We detect that in this case we can reserve more successful circuits (blue section of the bars). This is because fragmented circuits have to guarantee a buffer for all the replies, which forces us to set a low maximum of circuits per port (two in this case). On the other hand, replies with complete circuits will never block, so they do not need a buffer. This allows us to reserve more simultaneous circuits per input port (five in our case), and almost



(a) 16 cores



(b) 64 cores

Figure 4.4: Percentage of replies that travel on a circuit, with a failed circuit (could not be completely built), with an undone circuit (it was built but had to be undone), that were scrounger messages, that were not eligible for a circuit, and that were eliminated, for every circuit-building configuration tested.

all circuit failures come exclusively from output port conflicts (when we would need two circuits from different input ports to the same output port). Removing coherence messages (*NoAck*) has a significant impact by eliminating 20-30% of the replies. On the other hand, reusing circuits has only some impact on the 64-core configuration, because there are more circuits built on the network, and, therefore, a higher probability for scrounger replies to find a suitable circuit.

We then present results for basic timed circuits and three additional versions, always removing the non-necessary coherence messages. The three versions correspond to the ones introduced in Section 4.3.7 (Slack_, SlackDelay_, and Postponed_, where "_" is the number of cycles per hop). They are all simulated with different values for the slack, which is introduced as number of cycles per hop in the path. This way, the slack automatically adapts to the path length. In the basic timed version, we notice that there are more failed circuits than in the simple complete circuits scheme, especially in the 16-core system. This is because the strict timing restrictions cause the circuit to fail as soon as the request suffers any delay (loses any VC or switch arbitration), the optimistic timing calculation performed for the reply does not stand any more after that. We clearly see in the figure how the number of successful circuits rapidly increases as we introduce slack, effectively solving the problem. However, especially with 64 cores, we realize that increasing the slack does not necessarily allow more circuits to be built. This is because there is a trade-off in the number of cycles of slack we reserve: with a small slack, circuits fail because the timing cannot be met after small delays; on the other hand, higher slacks give more room for

Table 4.2: Percentage of circuit reservations in all routers that correspond to the first, second, third, fourth, and fifth reservation in that input. The percentage of failed circuits is also included.

Avg. circuit reservations in routers	**1st circuit**	**2nd circuit**	**3rd circuit**	**4th circuit**	**5th circuit**	**failed**
	48%	24%	7%	6%	6%	9%

delays, but reserve circuits for longer periods of time, making it more likely to have conflicts in output ports.

Apart from that, we notice a negative effect in all the timed circuits: the amount of circuits that get completely built but have to be undone without being used significantly increases. In the versions of the mechanism without timing, this was only caused by a pattern in the coherence protocol that happened sporadically (the L2 bank forwarding the request to the L1 owner). However, with timed circuits a reply must leave the network interface exactly within the reserved timeslot; otherwise, the circuit must be undone and the reply has to follow traditional router pipeline. This unpleasant situation happens due to unpredictable delays in the caches, mostly because requests are blocked in busy cache lines waiting for acknowledgements.

The last bar with the ideal circuit construction has been included for comparison. In the 16-core chip, our mechanism achieves results very close to the ideal, while the 64-core chip cannot exploit the mechanism to its fullest potential. Comparing Figures 4.4a and 4.4b, we notice that it is more complicated to build circuits with a larger chip, making the scalability of the mechanism a concern. This is due to the longer paths messages need to follow and the increased amount of traffic, which generate more conflicts and cause circuits to fail. This means that less replies will be able to reduce their latency and that more replies will need to use the same non-circuit VC, thus increasing latency. With the basic version of complete circuits for 64 cores, only about 25% of replies use a circuit, the remaining 75% must use the other VC, thus increasing congestion. This situation is however improved by two optimizations: removing acknowledgements reduces the amount of replies using the non-circuit VC down to about 50%; timed circuits increase the amount of replies that can use a circuit to about 40%, and in turn, also increases the number of acknowledgements that can be removed. With all these optimizations, there are less than 40% of replies contending for the non-circuit VC. Assuming that in the baseline configuration both VCs would be used equally (50% of replies in each VC), with the most optimized reactive circuits version we are actually reducing the load of that VC and maintaining the benefits of Reactive Circuits. We expect the effect of those optimizations to be even more relevant with bigger chips.

In the complete circuits versions, we can reserve several circuits per input port. As we explained in Section 4.3.2, we experimentally choose the number of simultaneous circuits to be big enough to reduce failed circuits due to lack of storage but small enough to minimize area and power. As an example, Table 4.2 presents the number of simultaneous circuits built for the complete circuits version with eliminated coherence messages in a 64-core chip. The table includes the percentage of circuit reservations at routers that correspond to the first, second, third, fourth, and fifth reservation in the same input. We notice that it is much more common to reserve the first circuit at an input port that it is to reserve the second or third. Nevertheless, the storage for all the five circuits is used and it leaves a small percentage of failed circuits due to lack of storage. Note that the remaining percentage corresponding to the number of failed circuit reservations at routers is a very small percentage, much smaller than the number of failed circuits with respect to the total number of circuits that we saw in Figure 4.4. This is because a successful circuit contributes with many reservations (one per hop), while a failed circuit may also have successful circuit reservations in several routers before have one single failure.

4.4.2. Network Latency

Figure 4.5 shows how the circuit construction affects message latency depending on the type of message: requests, replies eligible for circuit construction (Circuit_Rep), and replies for which we cannot build a circuit (NoCircuit_Rep). We include the baseline and ideal configurations, and the most relevant versions of the Reactive Circuits mechanism. Since the latency of requests does not change in any of those versions, we show it only in the baseline experiment. In each bar we distinguish between network latency (cycles each message spends in the network) and queueing latency (cycles before the message can enter the

Table 4.3: Router area savings in the different versions of the circuit-building mechanism. Negative values correspond to configurations with larger area.

Version	**Area Savings**	
	16 cores	64 cores
Fragmented	-19.28%	-18.96%
Complete	6.21%	5.77%
Complete Timed	3.38%	1.09%

network). In the baseline configuration we see that the replies eligible for construction have higher latency than the requests, which is because most of them have five flits instead of one; replies not eligible for circuits are normally acknowledgements composed of a single flit.

When we build circuits for the replies, either fragmented or complete circuits, the network latency is significantly reduced. The highest savings are obtained with the basic complete circuits, reusing complete circuits, and timed circuits with slack and delay, always removing unnecessary acknowledgements. To make a fair comparison, we have considered the latency of the eliminated coherence messages to be zero. In the configurations where we remove those messages, we notice a dramatic drop in the latency of replies that are not eligible for circuits.

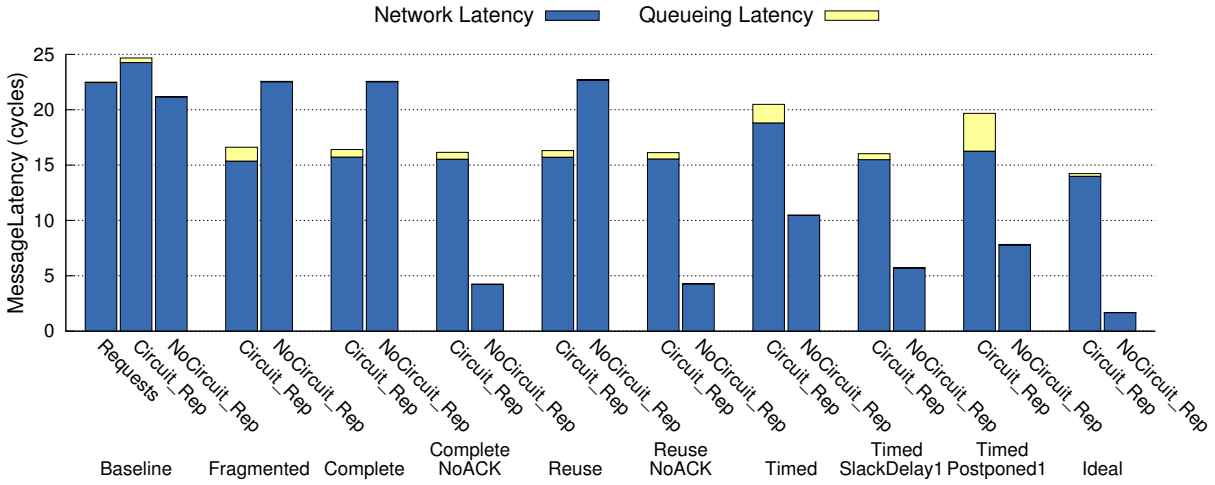
The timed circuits without any slack do not reduce network latency as much as the other options because, as we already showed in Section 4.4.1, not many circuits can be successfully built. We include two of the optimized versions of timed circuits: one with slack and delay, which significantly reduces the latency, and one with postponed circuits. The latter was implemented to increase the number of built circuits, but this was done by forcing a delay for every reply. Even though we can reserve many circuits, the forced delay has a negative impact on network latency. In fact, this option will not result in performance or energy improvements, so we will not include it in the following sections.

We notice that Reactive Circuits have a negative effect, especially in the 64-core chip: the queueing latency increases significantly, as well as the network latency for non-circuit messages. This is because virtual channels are now dedicated to each traffic type (circuit or non-circuit), so we eliminate their use as virtual lanes to reduce congestion, thus increasing the latency for non-circuit messages. Luckily, we can partially solve it by eliminating the unnecessary coherence messages, which lightens the load of the non-circuit VC.

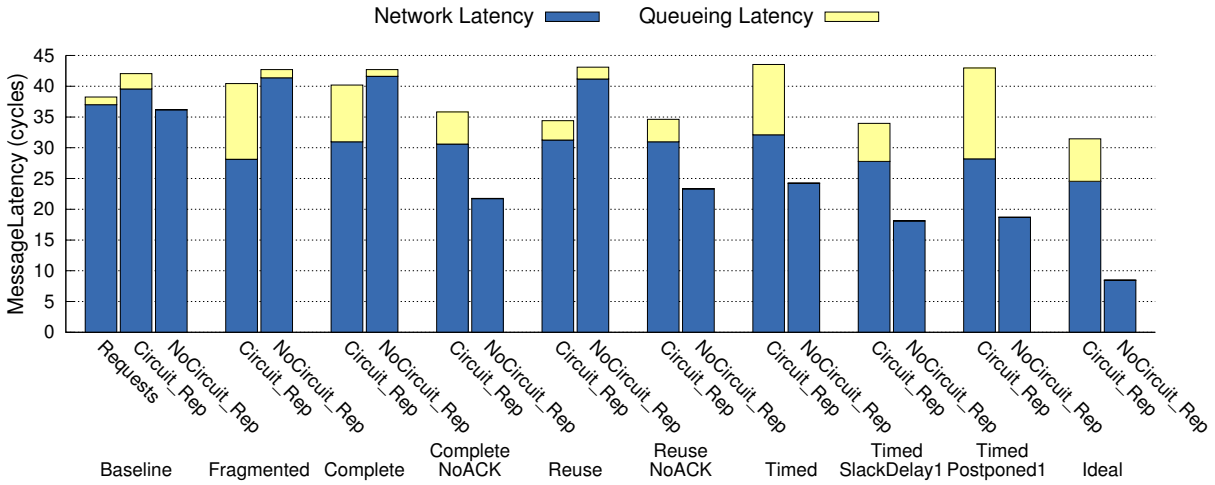
4.4.3. Router Area and Network Energy

Table 4.3 presents the savings in router area for each version of the mechanism compared with the baseline router with four VCs. With fragmented circuits, the area increases by almost 20% because we had to include an extra VC for circuits in order to increase the number of simultaneous circuits, as well as storage for the circuit information. In contrast, with complete circuits we also need to include storage for circuit information but we can eliminate the buffers in the VC dedicated for circuits, which makes the router area decrease by 6%. When enhancing complete circuits with timed reservations, we must also store the circuit timestamps, which cancels the benefit or removing the buffers almost completely. We always remove the buffering from one VC at every port in every router, therefore, these area savings will be maintained when scaling the chip to larger sizes.

These benefits in area, along with the speedup achieved as a result of the network latency reduction, translate into outstanding energy savings. Figure 4.6 depicts the normalized network energy for the most relevant configurations, including dynamic and static energy for both routers and links. The ideal version is not included because it involves unlimited storage for circuit information. With fragmented circuits, the energy increases the same way the area did. However, for the rest of versions, we substantially reduce the energy. The versions without unnecessary coherence messages involve further improvements



(a) 16 cores



(b) 64 cores

Figure 4.5: Message latency for different types of messages (requests, replies eligible for circuit construction (Circuit_Rep), and replies for which we cannot build a circuit (NoCircuit_Rep)) and Reactive Circuit versions, averaging the result from the parallel programs and one multiprogrammed mix.

due to the reduction in execution time and network utilization. The complete circuits removing the acknowledgements achieve the highest savings, with energy reductions of 15.2% and 20.8% in 16 and 64-core chips, respectively. The effect of the mechanism on the 64-core chip is more relevant because the network has a higher impact on larger systems.

4.4.4. System Performance

Figure 4.7 presents the average speedup of all parallel applications and one multiprogrammed mix for the most relevant versions of the mechanism. We notice that the speedups are not very large, mainly because the network is lightly loaded, which limits the effect of network latency on overall performance. Other similar proposals do not mention performance in their results, probably because the nice improvement in network latency translates into small performance improvements, like in our case. The speedup achieved by our mechanism is very close to the ideal one. Differences among versions are slightly more pronounced in the 64-core chip, where the network has a larger impact. The versions where we eliminate unnecessary coherence messages consistently achieve better results than their counterparts with all coherence messages. The version with the best performance results is the timed circuits with slack and delay, with performance improvements of 4.4% and 6.0% for 16 and 64 cores, respectively. Non-timed

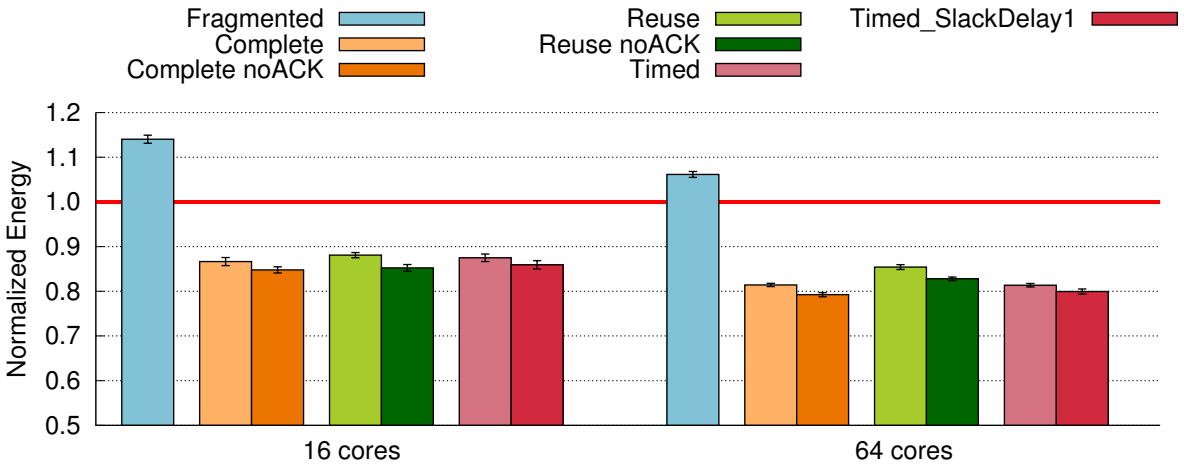


Figure 4.6: Network energy for the different versions of the Reactive Circuits mechanism normalized to the baseline without circuits. We present the average of all parallel applications and one multiprogrammed workload and include the standard error for every configuration.

complete circuits had larger energy savings than timed circuits even though their speedup is slightly lower (3.8% and 4.8% for 16 and 64 cores) because they do not need to store circuit timestamps.

Figure 4.7 also includes the standard error for every configuration, which is very small. The margin of error of our results with a confidence level of 95% is always less than 2% for 64 cores and less than 5% for 16 cores [60]. These results point out that, even though performance gains are small, they are consistent across all the simulated applications and statistically significant.

For complete timed circuits with slack and delay in a 64-core chip we present the speedup for each application in Figure 4.8. We can see that 50% of the simulated applications experience performance gains over 4.5%. There are several applications where the Reactive Circuits mechanism is especially beneficial and experience performance improvements above 10%, while only two applications out of the twenty two experience a very small slowdown (less than 2%).

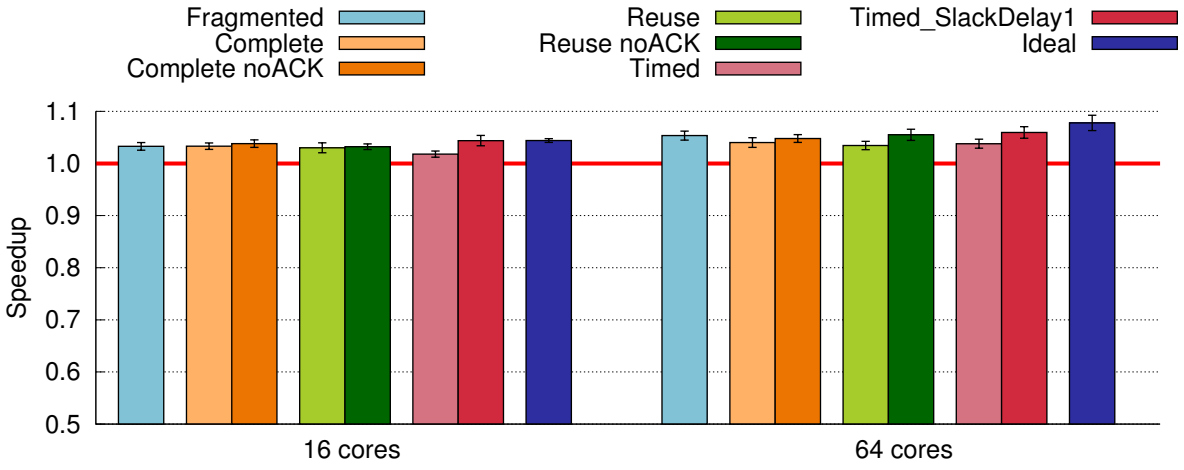


Figure 4.7: Speedup for the different versions of the circuit-building mechanism with respect to the baseline without circuits. We present the average of all parallel applications and one multiprogrammed workload and include the standard error for every configuration.

Under very adverse conditions, with heavy traffic loads, conflicts would be frequent and prevent complete circuits from being built, lowering system performance. However, timed circuits reduce the time circuits keep virtual channels occupied, thus rising the threshold over which the network would be too congested to build circuits and reduce latency.

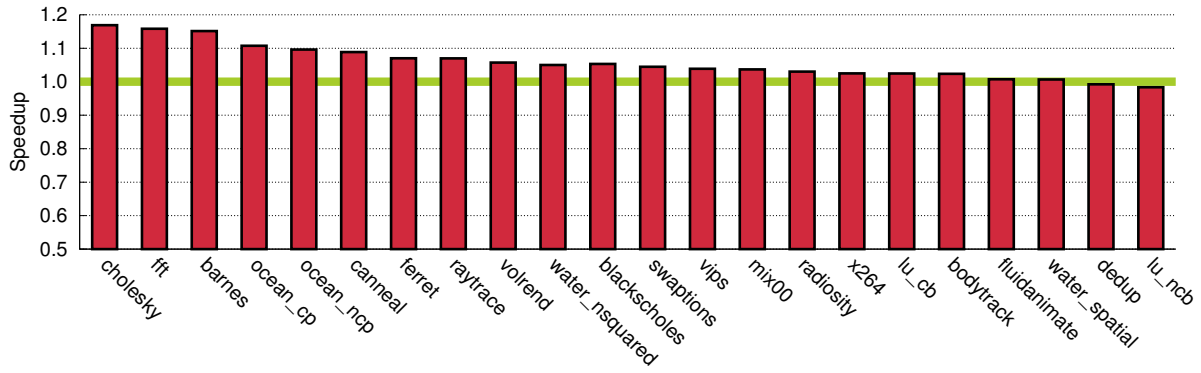


Figure 4.8: Speedup with respect to the baseline for every parallel application and one multiprogrammed mix for timed reactive circuits with slack and delay of 1 cycle per hop.

With the studied chip sizes (16 and 64 cores), all the versions of the mechanism achieve similar speedups. As the chip size increases, paths will be longer and there will be more messages using the network simultaneously. This will generate more conflicts and it will be more complicated to build complete circuits. For these reasons, timed circuits will be very useful to guarantee the scalability of the mechanism. They will only keep resources busy for short periods of time, thus reducing conflicts compared with non-timed circuits. Apart from that, it is considered that future systems with hundreds of cores will not be used monolithically to run one single application. Workloads do not offer enough parallelism to run efficiently on such a large number of cores. Therefore, the usage model of near-future networks-on-chip will likely involve partitioning and partition isolation, as it has already been implemented by Tilera with their Multicore Hardwall mechanism [150]. In a partitioned system, Reactive Circuits could be used independently inside each partition, thus eliminating concerns about the need to scale to a larger number of cores.

4.5. Concluding Remarks

CMPs are composed of multiple nodes connected via an interconnection network, which contributes with a substantial share to chip area, energy consumption, and system performance. The use of the interconnect is determined by the memory subsystem. By studying the communication patterns of the coherence protocol, we have come up with a smart network design that reduces both energy and area in the interconnect, and improves system performance.

Our work was inspired by the observation that most of the traffic follows a request-reply pattern, which helps anticipate the path most replies will follow. We have used that information to propose a mechanism called Reactive Circuits based on reserving network resources and dynamically building the circuit for the reply while the request travels through the network. Consequently, reply messages with a set-up circuit can go through the router in a single cycle, compared with the four cycles needed in the baseline router. Guaranteeing complete circuits for data messages has also enabled us to predict when they will reach their destination, and elegantly eliminate the need for their acknowledgement. To evaluate the proposal, we have performed full-system simulation with realistic parallel and multiprogrammed workloads. For a 64-core chip, where the NoC has more impact, our proposal with complete reactive circuits achieves an average energy reduction of 20.8% at the NoC, routers have 5.8% smaller area, and system performance improves by 4.8%.

Part III

Optical Network-on-Chip Design

This part of the thesis includes our work on optical networks-on-chip. It motivates the use of this emerging technology for on-chip communications and presents two design proposals that will contribute to a faster adoption of optical networks-on-chip: an algorithm to generate power-efficient optical rings with any number of nodes and automatic calculation of its power, and a complete network interface architecture for optical NoCs. We then incorporate the optical network into two realistic platforms (a chip multiprocessor and a multicore accelerator) and accurately compare performance and power with their electronic counterparts.

Chapter 5
Introduction to Optical Networks-on-Chip

Summary

This chapter presents the benefits of the emerging silicon photonics technology and how it can be used to implement optical on-chip networks (ONoCs). It introduces the two main optical network paradigms: space-routed ONoCs and wavelength-routed ONoCs, and justifies our decision to focus on the wavelength-routed optical ring topology for our research.

5.1. Motivation for Optical Networks-on-Chip

As the core count in homogeneous chip multiprocessors scales up, communication between distant cores requires multiple routing hops and overlapping messages experience significant contention and latency. Optical links can potentially operate at a much higher switching speed than electrical wires, and eliminate communication contention through the use of wavelength-division multiplexing (using several wavelengths in parallel on the same optical waveguide).

There is today consensus on the fact that optical interconnects can relieve bandwidth limitations at integrated circuit boundaries, and they are mainstream in data centres to overcome the bandwidth and communication power concerns. There are lots of ongoing development activities to exploit photonics for chip-to-chip communication [73], especially in the context of the network-in-package paradigm [8].

There is also a rich design space for on-chip optical communications [70, 14, 121, 146, 30, 80], including re-architecting of the DRAM memory sub-system [15], the revision of the processor-memory interface [142], or the development of new coherence protocols custom-tailored for the optical transport medium [80]. This significant amount of work has contributed to the foundation of cross-layer design methodologies for new optical networks [15], and projects superior bandwidth, latency and energy with respect to electrical wires beyond a critical length [67]. These benefits could be extended to on-chip communication architectures, either as standalone optical networks [74], or as hybrid interconnect fabrics [25]. However, it is still not clear whether it will be cost-effective to utilize this emerging interconnect technology for on-chip communication.

5.2. Space-Routed vs. Wavelength-Routed ONoCs

The inherent characteristics of optics (whose basic concepts are introduced in Section 1.1.2) have forced researchers to come up with new network designs radically different from the electronic ones, which can be classified in two groups: space-routed and wavelength-routed. Space-routed optical networks (SPONoCs) must set up a path in the optical network before they can start transmitting data, which introduces an unpredictable delay subject to conflicting requests on resources to be allocated, making them more suitable for scenarios featuring long-lasting connections [136]. They require two networks: an electronic NoC to perform the path-setup between cores and an optical network to transmit data packets once paths are reserved. SPONoCs use the wavelength-division multiplexing degree of freedom to increase the bit parallelism of each communication flow.

In contrast, wavelength-routed optical NoCs (WRONoCs) rely on the principle of wavelength-selective routing. As it is conceptually shown in Figure 5.1, every initiator can communicate with every target at the same time by using different wavelengths. For instance, initiator I1 uses wavelengths 1, 2, 3, and 4 to reach targets 1, 2, 3, and 4, respectively. Wavelengths are assigned to each communication to ensure that they will never interfere with each other on the network optical paths. This way, all initiators can simultaneously communicate with the same target by using different wavelengths. WRONoCs do not use wavelength-division multiplexing to increase bit parallelism, but to support contention-free all-to-all communication with a modulation speed of 10 Gbps/wavelength [103, 143]. This is a popular approach adopted in many optical network designs [147, 62, 104, 100, 15, 20] and attractive for specific application domains, where performance predictability and ultra-low latency communications are a must (e.g. data centre applications [66]).

We are aware that a number of intermediate solutions are feasible [15], however we decided to start the exploration of the huge design space of optical networks from WRONoCs since other solutions require some form of arbitration. Among the possible WRONoC topologies, we selected an optical ring inspired by [82]. Simplicity and low implementation cost make the optical ring topology one of the most appealing interconnection networks proposed in the open literature. In addition, and especially in a 3D-stacked scenario, the ring topology efficiently meets the place&route constraints unlike other solutions such as multi-stage and filter-based networks [23, 143]. As an example, Figure 5.2 depicts an optical ring with

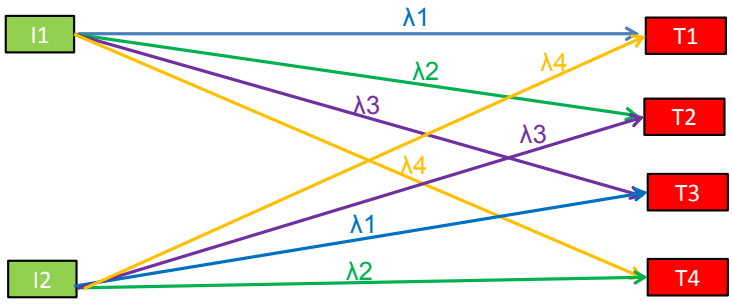
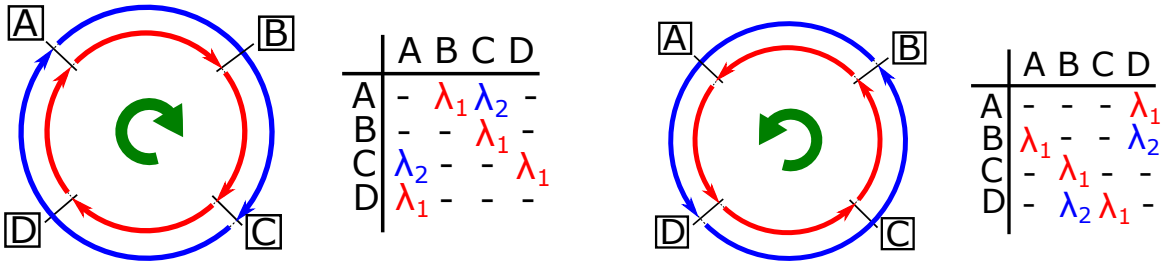


Figure 5.1: Wavelength-selective routing concept.

two waveguides and two wavelengths communicating four nodes. Note that the same wavelength can be used to implement several communications on the same waveguide if there are not any conflicts on the ring sections. Alternating clockwise and counterclockwise waveguides minimizes the number of required wavelengths by favouring the use of minimal paths.



(a) Clockwise waveguide

(b) Counterclockwise waveguide

Figure 5.2: Optical ring communicating four nodes with two waveguides and two wavelengths.

The light that travels inside the waveguides is generated by a laser source that can be located off-chip [29] or on-chip [28]. Since the technology for on-chip laser sources is still very immature, in this work we assume off-chip laser sources that generate light for all the required wavelengths. This light stream is then taken from the source to all the nodes in the chip using a laser distribution network.

Chapter 6
Designing Power-Efficient and Custom-Tailored Wavelength-Routed Optical Rings

Summary

Out of all the optical network-on-chip topologies, the ring has been proved to be far superior to its competitors: the contention-free all-to-all communications offer the lowest latency possible, while its clean physical design with few crossings and ring resonators provides unmatchable power results. The ring implements simultaneous communications by using a communication matrix that sets a distinctive waveguide-wavelength pair for each of them. That communication matrix has a high impact on energy consumption, but so far there have been very few efforts towards optimizing and automating its design. As far as we know, we propose the best optical ring design algorithm that produces rings with the lowest number of wavelengths and waveguides in the literature. The algorithm is completed with a layout-aware and fully automated power calculation framework to help the user choose the most power-efficient design point.

6.1. Introduction and State-of-the-Art

Among all the wavelength-routed topologies, the optical ring stands out as the preferred alternative, offering low latency communications along with reduced energy consumption [127]. Grani *et al.* present a comprehensive design-space exploration for optical rings and demonstrate their outstanding power consumption and performance results over electronic topologies [48]. A distinctive feature of optical rings is their flexible implementation, which opens up opportunities for their optimization and customization for specific interconnect requirements.

In practice, by varying the number of wavelengths and waveguides with the same connectivity requirements, the ring design can be tuned to produce infinite variations with very different power consumption values. To the best of our knowledge, the literature on the design space exploration of optical rings and on its automation is extremely poor. Grani and Bartolini briefly explore the trade-off between the number of waveguides and wavelengths and conclude that it is a topic worth studying [48]. LeBeux *et al.* propose the first algorithm to generate ring communication matrices [82]. Their main goal is to come up with a design methodology capable of materializing ring designs that meet the connectivity requirements of the system at hand. However, concerns arise from [82] about the efficiency of such design points, especially when considering that a power modeling framework is not reported, hence missing a fundamental quality metric to asses designs. This paper shares the same goal of automatically instantiating optical rings with the minimum allocation of resources, however, power optimization is the primary concern during the synthesis process. In practice, our tool searches for the right balance between number of waveguides and number of wavelengths to come up with a power-efficient interconnect solution.It automates optical ring generation driven by power efficiency, and builds upon a power modelling framework with layout accuracy. As an additional benefit, the tools is also capable of customizing optical rings for the communication requirements of the system at hand, thus avoiding unnecessary overdesign.

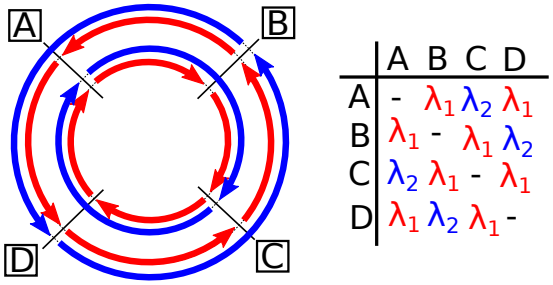
Automatically calculating the power consumption of the optical network becomes essential in order to prune the design space towards the most promising solutions. Bergman *et al.* analyse the insertion loss in an optical folded-torus taking into account the physical layout. Chan *et al.* explore insertion loss, crosstalk, and power consumption for the TorusNX and Square Root topologies [25]. PhoenixSim is a computer system simulator than includes optical network power calculations [27]. These proposals study power consumption for several existing topologies, but do not use that information to improve the topology design or choose the best among several topology variations. In contrast to previous work, we implement an automatic power consumption calculator for the ring that takes into account physical design constraints, such as core placement and optical power losses in the power distribution network. In particular, we use the layout-aware power models not only for power analysis of generated ring designs, but also to drive the synthesis process toward the most power-efficient ring configurations.

6.2. Motivation

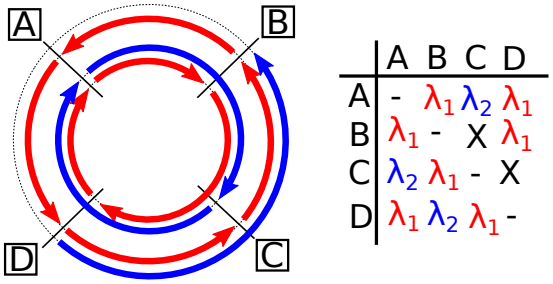
The optical ring is a wavelength-routed optical NoC, and relies on the principle of wavelength-selective routing, as it was introduced in Section 5.2. This means that every initiator can communicate with every target at the same time by using different wavelengths. The communication matrices (waveguide-wavelength pair to implement each communication) have to be designed to ensure that wavelengths will never interfere with each other on the network optical paths. As long as we stay within the restrictions imposed by the technology and the place-and-route constraints, the number of waveguides and wavelengths of the design can be tuned in order to implement all the required communications with minimal power. Including more waveguides on the ring increases the number of crossings (as we will show in Section 6.4), which leads to an interesting trade-off: multiplexing a lot of wavelengths on a few waveguides means the insertion loss of the wavelengths will be lower, but we will need to add up the power for a lot of them; on the other hand, increasing the number of waveguides will allow us to reduce the amount of needed wavelengths, but each one of them will have higher insertion loss.

Figure 6.1 depicts two rings that try to implement all-to-all communications for four nodes. Each

ring has a clockwise waveguide and counterclockwise waveguide, and the only restriction is that *the same wavelength cannot be used in the same section of a waveguide twice*. With a good wavelength assignment, Figure 6.1a is able to realize all the communications with two waveguides and two wavelengths. On the other hand, a bad wavelength assignment has been performed in the design of Figure 6.1b and, although there are still free sections in the outer waveguide, it would be necessary to include an extra waveguide or wavelength in order to have the same all-to-all communications, thus increasing power consumption. This example illustrates the importance of a good assignment of waveguides and wavelengths, which is what our algorithm will generate, minimizing the number of wavelengths and waveguides needed to communicate all the nodes and reducing power consumption.



(a) Good wavelength assignment



(b) Bad wavelength assignment

Figure 6.1: Optical ring communicating four nodes with two waveguides: an inner clockwise waveguide and an outer counterclockwise waveguide. Each waveguide is represented with two concentric circumferences, for each one of the two used wavelengths (red and blue). In the good wavelength assignment, all-to-all communications for the four nodes have been implemented with the two wavelengths using all the sections of the two waveguides. In the bad wavelength assignment, it has not been possible to implement all the required communications (from B to C and from C to D are missing).

6.3. Generating the Optical Ring Communication Matrices

We propose an optimized algorithm to communicate any number of elements through an optical ring. We follow the basic design ideas introduced by LeBeux *et al.* in [82]: a wavelength can be used to implement several communications on the same waveguide, and we alternate clockwise and counterclockwise waveguides. Our objective is to minimize the number of waveguides and wavelengths needed to implement all the communications without contention.

Our algorithm to generate the optical ring matrices and calculate the network power can be applied to communicate any number of elements in any tiled system architecture. Therefore, we do not impose any restrictions on the architecture, and the elements we communicate can be simple cores, large clusters, or other components. Besides, we can introduce as an input the communications that have to be implemented (connectivity matrix), enabling the use of the algorithm for more complex architectures such as the 3D design proposed in [82].

The mechanism we propose to build the ring communication matrices is detailed in Algorithm 1. As an input, the algorithm needs the number of waveguides of the ring, a maximum number of wavelengths, and the connectivity matrix. The latter allows us to indicate which specific nodes we want the ring to connect and, therefore, to use algorithm to build rings for any platform. The output consists of two communication matrices: one for the waveguides and one for the wavelengths for each communication. For a given number of waveguides, which may be determined by the place-and-route constraints, the algorithm generates the ring design with the minimal number of wavelengths.

For each communication that needs to be implemented in the ring (loop in line 5 of Algorithm 1), the algorithm first tries to set the connection on the minimal path between the two nodes reusing a wavelength already present in the design (lines 6 to 16). If that is not possible because some of the required ring sections are not free in any waveguide with any of the existing wavelengths, a new wavelength will be added to set the communication (lines 18 to 25). If the maximum number of wavelengths

Algorithm 1 Generation of optical ring communication matrices.

```
1: Input Data: num_waveguides, max_num_wavelengths, connectivity_matrix
2: Output Data: waveguide_matrix, wavelength_matrix
3: ring ← generate_ring(num_waveguides, 0 wavelengths)
4: used_wavelengths ← 0
5: for communications from connectivity_matrix COM do
6:     ▷ First try to reuse a wavelength to set the communication on the short path.
7:     success ← false
8:     for used_wavelengths wl do
9:         wg ← ring.get_free_waveguide_short_path(COM, wl)
10:        if wg exists then
11:            store_communication_short_path(COM, waveguide_matrix, wavelength_matrix, wl, wg)
12:            ring.store_use_short_path(COM, wl, wg)
13:            success ← true
14:            break
15:        end if
16:    end for
17:
18:    ▷ If it did not work, try adding a new wavelength
19:    if NOT success && used_wavelengths < max_num_wavelengths then
20:        ring.add_wavelength()
21:        used_wavelengths + +
22:        store_communication_short_path(COM, waveguide_matrix, wavelength_matrix,
             new wl, first wg)
23:        ring.store_use_short_path(COM, new wl, first wg)
24:        success ← true
25:    end if
26:
27:    ▷ If we could not add more wavelengths, try setting the communication on the long path
28:    for used_wavelengths wl do
29:        wg ← ring.get_free_waveguide_long_path(COM, wl)
30:        if wg exists then
31:            store_communication_long_path(COM, waveguide_matrix, wavelength_matrix, wl, wg)
32:            ring.store_use_long_path(COM, wl, wg)
33:            success ← true
34:            break
35:        end if
36:    end for
37:
38:    if NOT success then
39:        ERROR : Unable to generate ring
40:        break
41:    end if
42: end for
```

had already been reached, then the algorithm will try to set the communication on the non-minimal path, going around the ring in the other direction (lines 27 to 36). If it is not possible to do that either, the algorithm will finish its execution unable to generate the ring design with the given input (lines 38 to 41).

The first difference of the algorithm in [82] with respect to ours is that they fix the number of wavelengths to use and utilize all of them in the same waveguide until it is not possible to set any of the remaining communications, at which point they add a new waveguide. The benefit of this idea is that the number of wavelengths that can be multiplexed in the same waveguide is given by the technology, so it is a very reasonable input. The drawback is that using up all the sections of the first waveguide before adding a new one forces the algorithm to use non-minimal paths for several communications that could have found a shorter path on a second waveguide. Having longer paths has a negative impact on power because it increases the number of crossings and the propagation loss, as we will explain in Section 6.4. To avoid this problem, we fix the number of waveguides of our ring and reuse the same wavelength on all of them as much as we can before adding a new one, always trying to set communications on the shortest path first.

The inputs of the algorithm allow us to run it several times for a given number of waveguides to get different communication matrices by gradually reducing the maximum number of wavelengths. We can generate a first design without a restriction on the number of wavelengths, which will have only minimal paths. After that, we can try reducing the maximum number of wavelengths to generate rings with fewer wavelengths but more non-minimal paths, which yields a very interesting power trade-off.

A final and very important detail not mentioned in [82] is the order in which communications are set. We explore two different options:

- Setting long-path communications first. The objective of this choice is to have the shorter communications fill the gaps left on the ring by the longer communications.
- Setting short-path communications first.

The interesting results obtained from these options will be discussed in the evaluation section.

6.4. Calculating the Power

The second step of the ring design consists of calculating the power taking into account layout constraints and physical-level parameters. As a novelty, we include the power of the laser distribution network, which brings the power from the laser source to all of the nodes.

The laser source generates the optical power for the wavelengths, which then has to reach all the nodes and the waveguides in each node. To send the same wavelength to several paths at the same time we need to use splitters. To illustrate the complexity of the laser distribution scheme, we present an example in Figure 6.2 for a system with three nodes and three wavelengths, assuming the ring has one single waveguide. For each node, the power needed for each wavelength has been calculated from the insertion loss of the path that uses that wavelength and starts at that node. When designing how to distribute the laser to all the nodes, we need to set the appropriate splitting ratio so that the the required optical power reaches every node while minimizing the waste. Ideally, we would need selective splitters to apply the required splitting ratio to each individual wavelength in order to bring the exact power needed to every node. However, more pragmatic solutions do exist, which pose less stringent requirements on splitter technology:

- Using a separate distribution waveguide for each wavelength would allow the splitters to have the ideal splitting ratio at every node for every wavelength, but it would involve a very large number of crossings at every node to inject the laser into all waveguides, as we will show in Figure 6.3.
- Using the same power for all the wavelengths at each optical network interface (ONI), corresponding to the worst-case power across all wavelengths in the ONI, would allow us to use the same ratio

Table 6.1: Physical level parameters.

Chip size	16x16 mm	Modulator loss	1 dB
Crossing loss	0.15 dB	Coupler loss	1 dB
Propagation loss	0.15 dB/mm	Filter drop loss	1 dB
Bending loss	0.005 dB	Photodetector loss	1 dB
Splitter loss	0.2 dB	Coupler efficiency	90%
Receiver sensitivity	-20 dBm	Laser efficiency	8%

for all the splitters. In the example from Figure 6.2, we would need to bring 15 mW to all the wavelengths in ONI0, 20 mW for all the wavelengths in ONI1, and 30 mW for ONI2. The drawback of this method is the power waste. For example, wavelength 1 in ONI 0 would be getting 15 mW when it actually needs only 5 mW.

- Using a fixed splitting ratio for all the splitters in the laser distribution network. With a 50% splitting ratio, the same power will have to be sent to both branches for each wavelength. For example, the splitter directly above ONI1 needs to send 7 mW of power to ONI1 and 2 mW to ONI2 for wavelength 1. Since the ratio is 50-50, it will send 7 mW to both branches. For wavelength 2 it will send 12 mW, and for wavelength 3, 30 mW. Similarly to the previous option, we would also be wasting power.

Without lack of generality, we decide to choose the third option. From each node we will likely need to set paths of different lengths with very different power needs. Therefore, the second option that proposes to use the same power for all the communications starting at each node would be more wasteful. Besides, I we will show in Figure 6.3, the laser distribution network is implemented as a perfect binary tree, which will reduce the different of power that needs to be sent to each branch at every splitter.

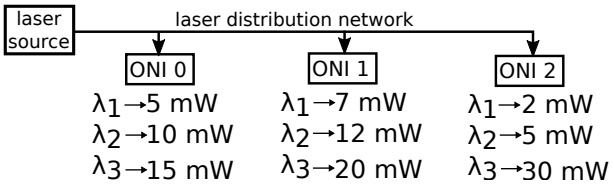


Figure 6.2: Example of the optical power needed at every ONI for each wavelength.

Our algorithm takes into account details that have a direct impact on power, not only by favouring the use of minimal paths, but also by reducing the waste of power in the chosen laser distribution network. It sets up communications in path-length order, and uses the same wavelength as much as possible before including a new one. As a result the same wavelength will be used to implement paths of similar lengths, thus minimizing the differences across nodes for every wavelength and reducing wasted power in the laser distribution network. These details of the algorithm that have a direct impact on power were not considered in [82].

The laser distribution network to reach all the nodes is implemented as a perfect binary tree (or as close as possible with the given number of nodes) to reduce the number of splitters, as shown in the left hand side of Figure 6.3, but it is an input that can be easily modified. Inside every ONI, the laser needs to reach all the waveguides, which generates crossings typically overlooked when designing optical ring topologies (right hand side of Figure 6.3). When calculating the power we consider physical level parameters that can also be introduced as an input. For this paper, we use the values shown in Table 6.1.

In order to focus purely on the power required by the generated communication matrices without the effect of the chosen laser distribution network, in some cases we will also calculate the power needed with an ideal power distribution network. We will simply add up the required power for all the paths in the ring assuming the splitters do not impose any restrictions and the insertion loss of the laser distribution network is zero.

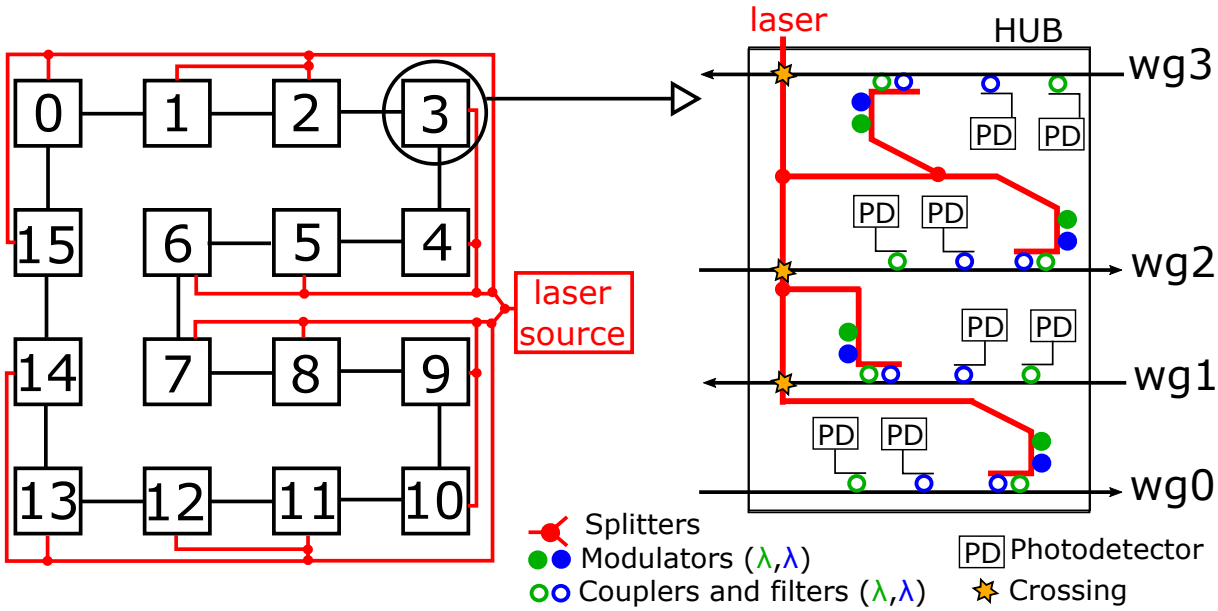


Figure 6.3: Laser power distribution tree in a 16-node ring and detail of the distribution of the laser to all the waveguides inside an ONI.

6.5. Evaluation

This section compares our algorithm with the only other existing algorithm for ring design [82], shows the number of wavelengths and waveguides needed to communicate chips of different sizes, and analyses the power consumption.

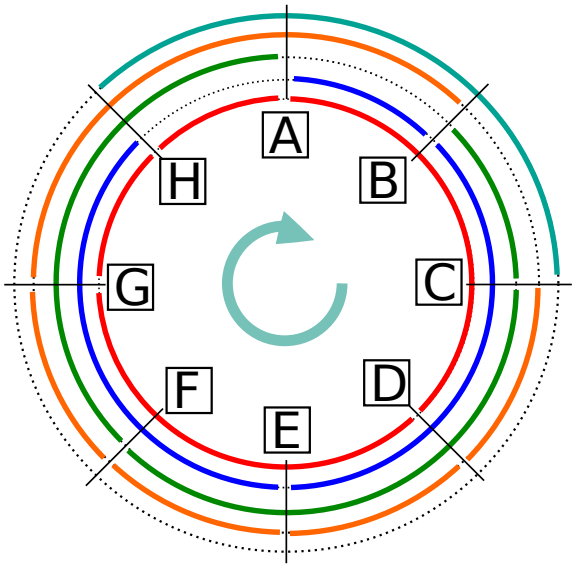
6.5.1. Detailed Example

We start by applying our algorithm to a detailed example. To facilitate comparison, we have reproduced the 3D architecture with two layers and four ONIs per layer presented in [82], and built an optical ring using two waveguides like they do. Figure 6.4 represents the communications implemented by each wavelength on the two available waveguides (note that the proposed two-layer system does not need all-to-all communications). Our algorithm manages to build all the communications with five wavelengths instead of the six needed in [82]. Also, the need of the algorithm from [82] to completely fill the first waveguide before adding the second forces them to use non-minimal paths on the first waveguide for communications that could have been implemented with minimal paths on the second waveguide.

We have also compared the power needed for the rings obtained with both algorithms, applying the power calculation described in Section 6.4 to both of them. Our optical ring requires 91.2 mW, while the ring from [82], with more wavelengths and longer paths, needs 117.3 mW, a 27% more. With the ideal laser distribution network, our design needs 15.2 mW, while the one in [82] needs 15.4 mW. The difference in this case is smaller because the penalization of the longer paths is not as relevant. Besides, as a side-effect of their algorithm, they use the outer waveguide (with crossings at the hubs) less frequently, thus reducing insertion loss. This unexpected benefit would not have a big impact in larger systems with more waveguides.

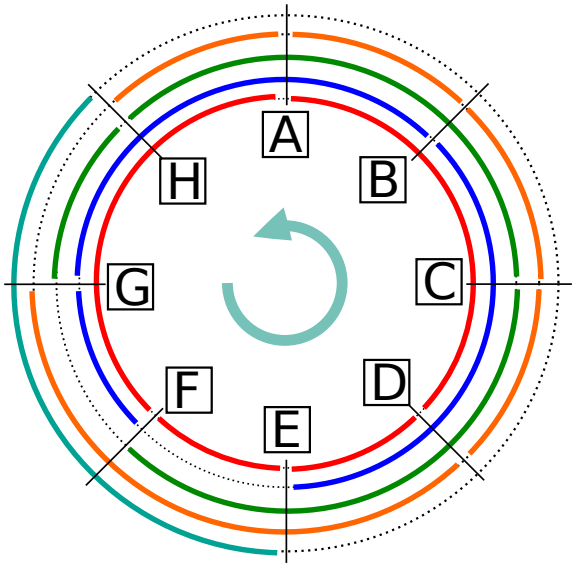
6.5.2. Exploration of the Number of Wavelengths and Waveguides

We run our algorithm for rings of different sizes and get the minimum number of wavelengths needed to implement all-to-all communications with different number of waveguides. Figure 6.5 shows our results, along with the available data from the algorithm designed by LeBeux *et al.* [82]. As expected, when increasing the number of nodes to communicate we need more waveguides and/or wavelengths to implement all the communications. Also, increasing the number of waveguides allows us to reduce the



	A	B	C	D	E	F	G	H
A	-	1	-	0	-	-	-	-
B	-	-	2	-	1	-	-	-
C	-	-	-	3	-	2	-	-
D	-	-	-	-	3	-	0	-
E	-	-	-	-	-	3	-	1
F	2	-	-	-	-	-	3	-
G	-	3	-	-	-	-	-	0
H	0	-	4	-	-	-	-	-

(a) Waveguide 0, clockwise



	A	B	C	D	E	F	G	H
A	-	-	-	-	-	0	-	3
B	3	-	-	-	-	-	1	-
C	-	3	-	-	-	-	-	2
D	0	-	3	-	-	-	-	-
E	-	1	-	0	-	-	-	-
F	-	-	2	-	0	-	-	-
G	-	-	-	3	-	1	-	-
H	-	-	-	-	4	-	2	-

(b) Waveguide 1, counterclockwise

Figure 6.4: Wavelength assignment in two waveguides to connect 8 nodes distributed in two layers, reproducing the detailed example presented in [82].

number of wavelengths because each one of them can be used for more communications.

We note that, given the same number of waveguides, our algorithm is able to build the ring with fewer wavelengths than [82] in every case, and the differences become more prominent with larger systems.

6.5.3. Power Consumption Analysis

We analyse the number of waveguides and wavelengths of each ring configuration along with the power it will consume and detect some interesting trade-offs. Figure 6.6 shows the power for the different rings our algorithm generates to implement all-to-all communications for 16 nodes, both with the realistic and the ideal power distribution networks described in Section 6.5.1. It includes the results obtained with the two options regarding the order in which communications are set: setting longer-path communications first (blue) and setting shorter-path communications first (red). We leave out of the graph the configurations

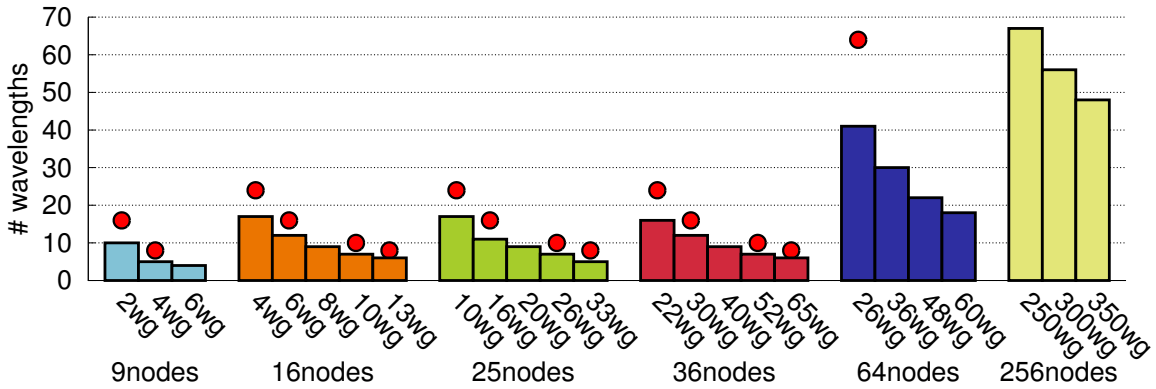


Figure 6.5: Number of wavelengths needed to implement all-to-all communications with optical rings with different number of waveguides (wg) in systems with increasing number of nodes. The red dots represent the number of wavelengths required to implement the same communications with the same number of waveguides by the only other existing ring design algorithm [82]. Please note that these numbers have been extracted from a graph and there may be small imprecisions.

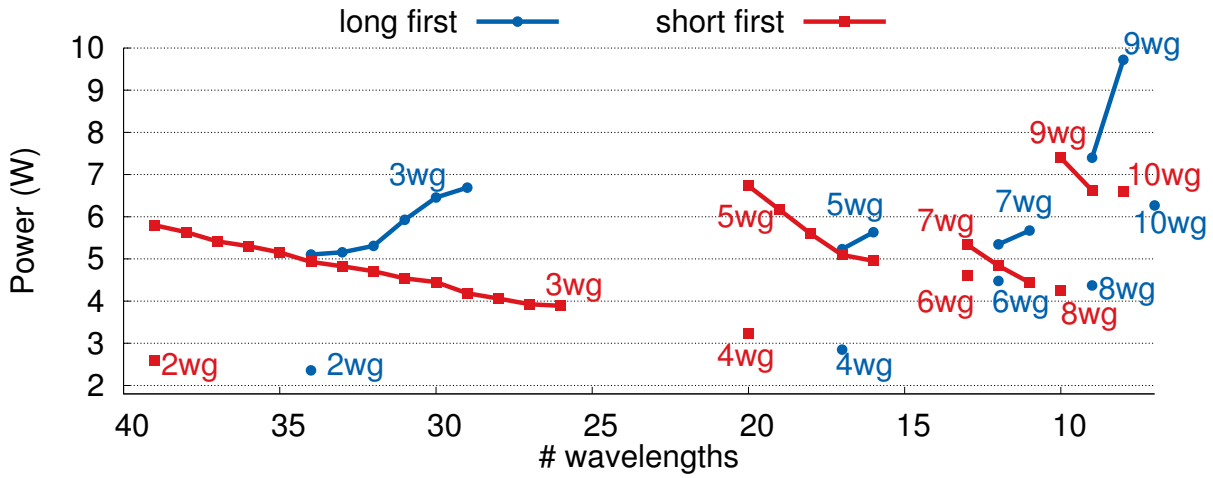
with one single waveguide because they have extremely high power consumption and require a lot of wavelengths, since they all need to share the same waveguide and all communications have to follow the same clockwise direction.

For a given number of waveguides, comparing the long and short-path first versions we notice that the first configuration the algorithm can build, where there are only minimal paths, is always better with long paths first: it has fewer wavelengths and consumes less power. This is because, as we already anticipated in Section 6.3, if we leave the short paths for the end they will be able to fill the gaps in the ring left by the longer paths.

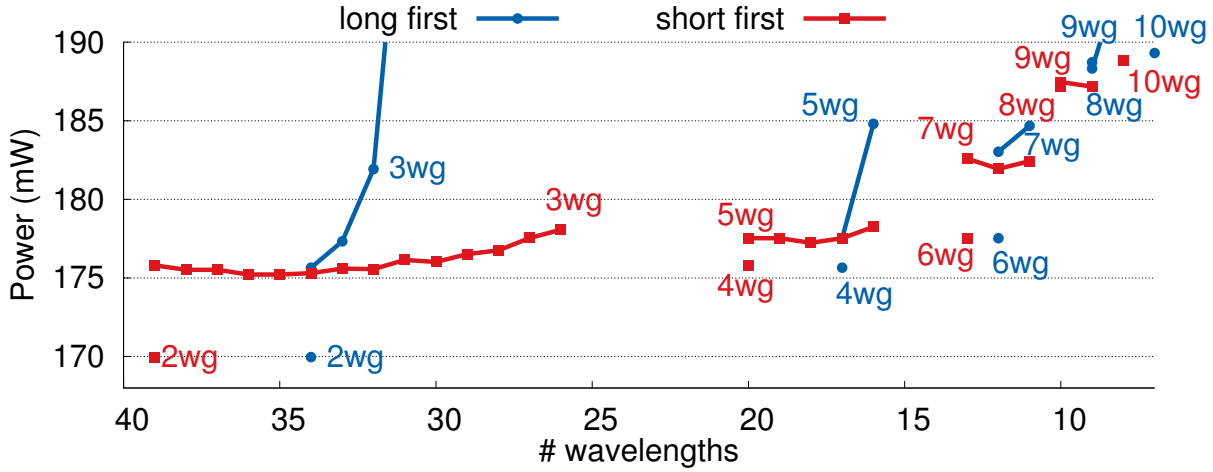
Focusing on the configurations built with the same number of waveguides with long paths first, we notice that, as the algorithm manages to reduce the number of wavelengths, the power surprisingly increases. This is because the reduction in the number of wavelengths is achieved by introducing non-minimal paths, which increases insertion loss (due to propagation distance and number of crossings). However, when building short-paths first with the realistic power distribution network, the power actually decreases as we build configurations with fewer wavelengths. The non-minimal paths to maximize wavelength reuse are set when a lot of the ring sections are already occupied, corresponding to the last communications implemented on the ring. With short paths first, those last communications are the ones with longest paths. Therefore, the difference in length between the minimal and the non-minimal path is small and will not penalize insertion loss as much. For example, in a 16-node ring a minimal path of 7 hops corresponds to a non-minimal path of 9 hops, which is a small difference. In contrast, a minimal path of 1 hop corresponds to a non-minimal path of 15 hops. This power reduction is true with the realistic power distribution network, where the 50% splitting ratio forces the power distribution network to send the worst-case power to both branches at every splitter. For the ideal network, splitters are not included, so power increases with the number of non-minimal paths also in the short-paths first version.

We also notice that when the number of waveguides is even results are much better than when it is odd. That is because there are the same number of clockwise and counterclockwise waveguides, which helps build a more balanced ring with minimal paths only. We also notice that with an even number of waveguides a design that minimizes the number of wavelengths by implementing non-minimal paths can never be found. In contrast, this is common for configurations with an odd number of waveguides because the extra waveguide than unbalances the design provides additional room on that direction for communications to be built on the non-minimal path. With higher number of waveguides it is rarer to find additional configurations with reduced number of wavelengths because the number of wavelengths is already small, and they are used across a higher number of waveguides.

It is also worth mentioning that these trends do not continue when generating rings for larger platforms. With more nodes, it is normally not possible to generate ring configurations with non-minimal paths



(a) Realistic power distribution network



(b) Ideal power distribution network

Figure 6.6: Power to implement all-to-all communications on a 16-node chip with varying number of wavelengths and waveguides. It also includes the two versions of the algorithm: setting long-path communications first and setting short-path communications first.

and fewer wavelengths than the first configuration obtained by the algorithm. This means that the best configuration is always obtained by setting the communications of the longest paths first.

Finally, the best ring configuration power-wise is the one with only two waveguides, both with the realistic and ideal power distribution networks. As we explained in Section 6.4, the laser power injected in the hub generates crossings in all the waveguides except the innermost one. Including more waveguides means that there will be more communications in the waveguides with crossings, therefore increasing the insertion loss of those paths. Also, in the realistic power distribution network, it involves including more splitters to bring the laser to all the waveguides.

6.5.4. Customizable Ring Designs

When communicating complex systems, we may need to build a ring for more complicated connectivity matrices instead of the basic all-to-all communications. In this experiment, we start with a fully connected 16x16 system and randomly remove some of the required connections to test if our algorithm is still able to come up with efficient designs. Figure 6.7 shows the power of the ring designs for 16 nodes with two waveguides using the realistic power distribution network, starting from full-connectivity and gradually

Table 6.2: Execution time of the algorithm

Configuration	Generate Ring (ms)	Calculate Power (ms)
9nodes, 2wg	0.14	0.06
16nodes, 4wg	1.3	0.3
25nodes, 10wg	8.7	0.5
36nodes, 22wg	25.9	0.79
64nodes, 26wg	387.2	4.0
256nodes, 250wg	660.5	135.5

decreasing the number of required paths. We can clearly see that the algorithm is very successful in optimizing power when reducing the number of needed connections, which makes it a suitable option for systems with specific connectivity requirements that need a customized design.

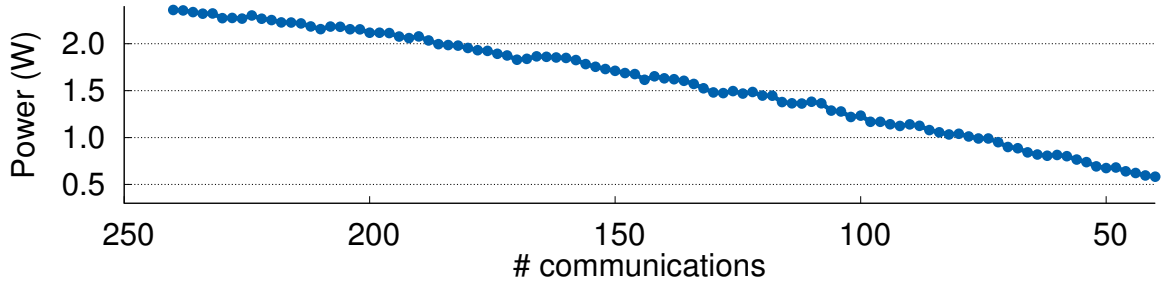


Figure 6.7: Power for ring designs to connect 16 nodes with two waveguides using the realistic power distribution network, starting from full-connectivity and gradually decreasing the number of required paths. Paths are randomly removed and the average power of 20 runs has been calculated for each point of the graph. Note that removing one path removes the communication between the two nodes in both directions.

6.5.5. Computation Time

To analyse the execution time of the algorithm we run it on an Intel Xeon E5606 that runs at 2.13 GHz and has 8 GB of RAM. Table 6.2 presents execution time for our algorithm to generate rings and calculate power for systems of different sizes. We observed that for a given number of nodes, execution time does not significantly change for different number of waveguides, thus we present results only for one case per chip size. We can see that the execution time increases with chip-size, but it is small even with the largest sizes.

6.6. Concluding Remarks

We present a tool to generate ring communication matrices with minimum number of waveguides and wavelengths, while optimizing for power efficiency. We automatically calculate power with physical constraint awareness including the contribution of the laser distribution network. Our algorithm is able to generate ring designs with fewer waveguides and/or wavelengths than any other existing proposal for any number of nodes. We demonstrate the importance of the order in which communications are set and show that an even number of waveguides allows for more balanced designs with reduced power. With a given number of waveguides, reducing the number of wavelengths does not necessarily mean saving power, because it enforces the use of non-minimal paths, which increases insertion loss.

Both with ideal and realistic laser distribution networks, we find out that the best design point is the ring with only two waveguides, pointing out that adding extra wavelengths is more cost-effective

than adding extra waveguides. There is a large margin between results with the ideal and realistic power distribution networks, indicating that power not only depends on an efficient communication matrix, but also on a good laser distribution network design, which should be the focus of further optimizations.

Chapter 7
A Complete Electronic Network Interface Architecture for Wavelength-Routed Optical NoCs

Summary

Although many valuable research works have investigated the properties of optical NoCs, the vast majority of them lack an accurate exploration of the network interface architecture (NI) required to support optical communications on the silicon chip. The complexity of this architecture is especially critical for a specific kind of ONoCs: the wavelength-routed ones. These are capable of delivering contention-free all-to-all connectivity without the need for path reservation, unlike space-routed ONoCs. From a logical viewpoint, they can be considered as full nonblocking crossbars, thus the control complexity is implemented at the NIs. To our knowledge, we propose the first complete network interface architecture for wavelength-routed optical NoCs, by coping with the intricacy of networking issues such as flow control, buffering strategy, deadlock avoidance, serialization, and above all, their codesign in a complete architecture. The evaluation methodology spans from area and energy analysis via actual synthesis runs in 40nm technology to RTL-equivalent SystemC modelling of the network architecture, and aims at verifying whether the projected benefits of ONoCs versus their electrical counterparts are still preserved when the complexity of their network interface is considered in the analysis.

7.1. Introduction

Projected quality metrics for optical networks-on-chip are overly optimistic for a number of reasons extensively discussed in [15], including optimistic technology assumptions, use of logical topology designs instead of physical ones, and overlooking static power. A big approximation of many projected results is the lack of a complete and accurate network interface architecture for driving on-chip optical communication, which may account for a large fraction of the overall network complexity.

This is especially true for a particular category of ONoCs: the Wavelength-Routed ones (WRONoCs). These networks deliver contention-free global connectivity without need for arbitration or packet routing by replicating the amount of wavelengths used, and by associating each wavelength with a different and non-conflicting optical routing path. WRONoCs can be conceptualized as non-blocking full crossbars. Therefore, all the complexity of the control architecture is located at the boundary of the interconnect fabric. To our knowledge, no complete NI architecture has been reported so far in the open literature, with the exception of NIs for space-routed ONoCs. However, these are conceptually simpler due to the intuitive conversion of electrical bit parallelism into optical wavelength parallelism [51]. In contrast, WRONoCs rely on serialization or on a limited bit parallelism, which questions the achievement of performance goals. Even neglecting this difference, the NI design for an optical medium is a non-trivial task due to the interdependent issues that come to the forefront: end-to-end flow control, buffer sizing, clock re-synchronization, and serialization ratio.

This Chapter takes on the challenge of designing and characterizing the complete NI architecture for emerging WRONoCs, in an attempt to validate whether (and to what extent) the projected benefits of optical NoCs over their electrical counterpart are still preserved with the NI in the picture. The distinctive feature of this work is the completeness of the architecture, including both initiator and target side. Especially, the digital architecture to enable optical NoC operation has been designed out of state-of-the-art basic building blocks (e.g., mesochronous synchronizers and dual-clock FIFOs), thus reflecting realistic quality metrics. Finally, for the optical and opto-electronic components, we used a consistent set of static and dynamic power values from the same literature source [16, 15].

Our evaluation methodology consists of 2 steps: first, we synthesize and characterize latency, and power for all of the architecture components on a low power industrial 40 nm technology; second, we set up a complete SystemC-based simulation infrastructure (for both the optical and electronic parts) with RTL-equivalent accuracy, thus enabling to capture fine grained performance effects associated with the microarchitecture.

7.2. Related Work

Early ONoC evaluation studies rely on coarse, higher-level models and/or unrealistic traffic patterns [121, 146, 62, 120], while more recent ones come up with complete end-to-end evaluations using real application workloads [79] and/or more accurate optical network models [26]. Looking in retrospect, early results have been only partially confirmed, nonetheless showing the potential of ONoCs for on-chip communication. For instance, with an aggressive electrical baseline technology, it became more difficult to make a strong case for purely on-chip nanophotonic networks [79]. However, even in this case, it was still possible to show significant potential in using seamless intra-chip/inter-chip nanophotonic links. Moreover, other works (such as [15]) related network energy to total system energy, thus making the point for fast interconnect fabrics capable of cutting down the static energy of non-network components, although they are themselves not energy-efficient.

The refinement of comparative analysis frameworks is far from stabilizing. In fact, other missing aspects are progressively coming to the forefront as the ONoC research concept strives to become an industry-relevant technology. For example, the NI architecture has so far been overlooked in most evaluation frameworks, or in the best case, only considered in the early stage of design [103, 15, 20, 51].

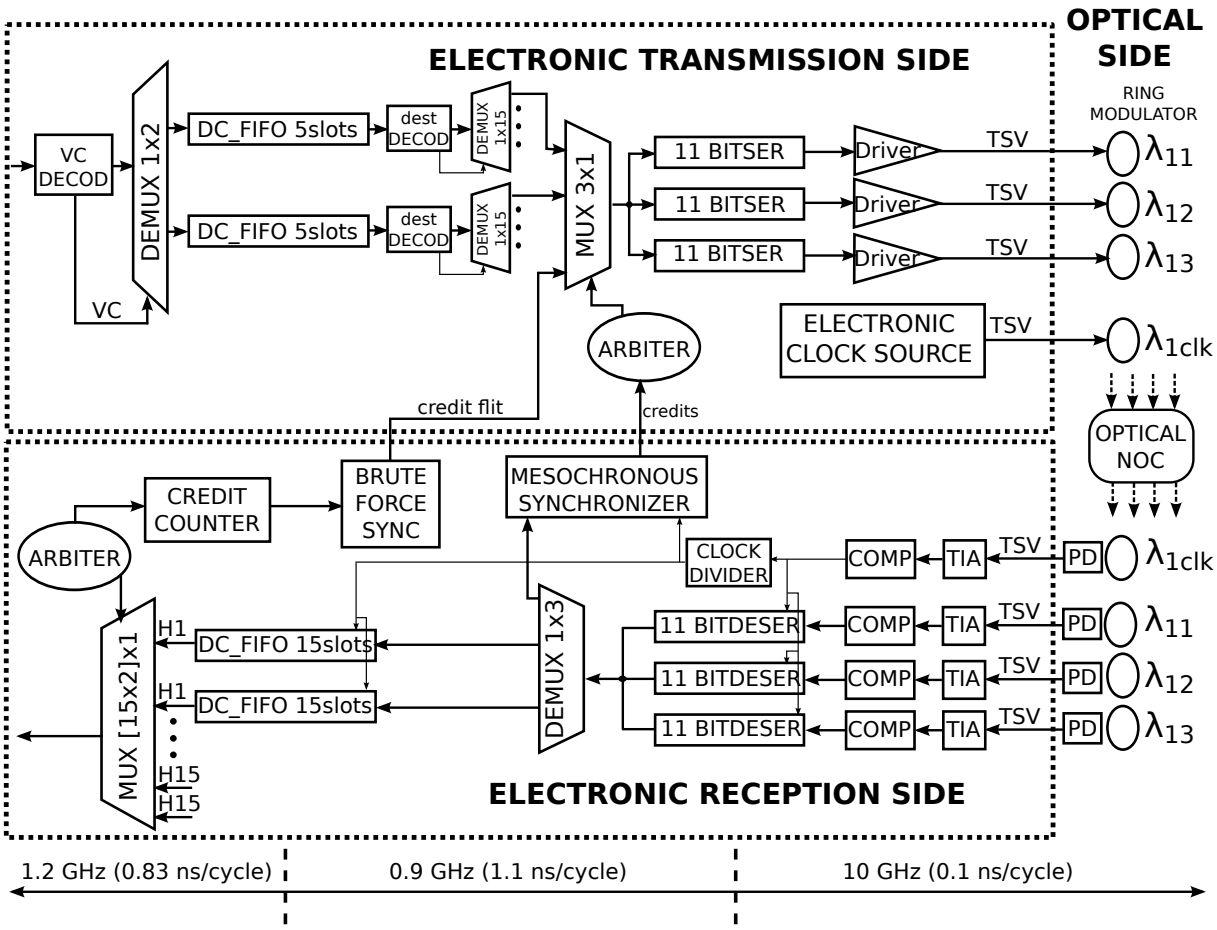


Figure 7.1: Optical network interface architecture for wavelength-routed optical NoCs, with 2 virtual channels and 3-bit parallelism

The distinctive features of our approach for the two systems included in our study are: architecture completeness, comparison with electrical interface counterparts, physical synthesis of digital components, RTL-equivalent SystemC modelling for microarchitectural performance characterization, and analysis of the impact of the NI and NoC parameters.

7.3. Network Interface Architecture

This section presents, to the best of our knowledge, the first complete network interface architecture for wavelength-routed optical networks, as depicted in Figure 7.1 [115, 111, 116, 112, 128]. As a consequence, the objective is not to present the best possible design point, but rather to start considering the basic components, and deriving guidelines about which ones deserve the most intensive optimization effort. Clearly, ONoCs move most of their control logic to the NIs, which should therefore not be oversimplified with abstract models. Even though we are using an optical ring topology, this NI architecture can work with any wavelength-routed optical network.

During the design of the NI, we consider a high-impact system requirement: message-dependent deadlock avoidance. Message-dependent deadlock arises from the interactions and dependencies created at network endpoints between different message types (as depicted in Figure 7.2) [32, 50]. In a complete system, the combination of these effects may lead to cyclic dependencies and block resources at both network endpoints and inside the network indefinitely. When we apply these considerations to WRONoCs, the problem gets simplified by the fact that there is no buffering inside the network, which means messages don't stop along the path, and, therefore, can't get blocked. However, we must break the dependency

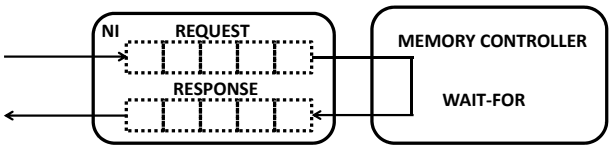


Figure 7.2: Dependence between a request and response at the NI.

cycles at the boundaries of the NoC by allocating a different buffer for each kind of message in the NI. This has direct implications on the buffering architecture of our target NI (that is, on the number of virtual channels), depending on the communication protocols the WRONoC needs to support. The design presented in Figure 7.1 includes 2 virtual channels as an example.

This should be combined with the requirements of wavelength routing: each initiator needs an output for each possible target, and each target needs an input for each possible initiator. As a result, in an initial version of the NI, each initiator came with 2 FIFOs for each potential target, and each target, with 2 FIFOs for each potential initiator. In a more energy-efficient version of the NI (see Figure 7.1), the transmission side reuses the same 2 FIFOs for all destinations, and flits are dispatched to different paths afterwards (all the logic components after the 1x15 demultiplexers are replicated for each destination). This energy optimization will not cause relevant latency degradations, since the nodes produce packets sequentially (not in parallel for every destination) and the network is lightly loaded. All the FIFOs at both the transmission and the reception side must be dual-clock FIFOs (DC FIFOs) to move data between the processor frequency domain (we assume 1.2GHz) and the one used inside the NI. As hereafter explained, the latter depends on the bit parallelism. We used the DC FIFO architecture presented in [140].

To size the DC FIFOs, we considered the size of the packets that would use each of the VCs: control packets need 2 flits, while data packets need 21 flits assuming flits are always 32 bits long. The FIFO depth will be assessed in the experimental results. All the VCs in the transmission side have 5 slots, which is the minimum size the DC FIFO needs to achieve perfect throughput. For the reception side, we sized the VCs based on the round-trip latency in order to allow uninterrupted communications, ending up with 15-slot DC FIFOs.

After flits are sent to the appropriate path depending on their destination, they must be translated into a 10 GHz bit stream in order to be transmitted through the optical NoC. This serialization process is parallelized to some extent to increase bandwidth and reduce latency. 3-bit parallelism means that 3 serializers of 11 bits each work in parallel to serialize the 32 bits of a flit, resulting on a bandwidth of 30 Gbps. The bit-parallelism determines the frequency inside the optical NI: 1.1 ns (0.1*number of bits) are needed to serialize a flit with 3-bit parallelism, but only 0.8 ns are needed with 4-bit parallelism. In turn, this also impacts the size of the reception DC FIFO based on round-trip latency, which increases from 15 to 17 slots when moving from 3 to 4-bit parallelism.

Another key issue to be considered in NIs is the resynchronization of received optical pulses with the clock signal of the electronic receiver. In this paper we assume source-synchronous communication, which implies that each point-to-point communication requires a strobe signal to be transmitted along with the data on a separate wavelength. With current technology, this seems to be the most realistic solution, even considering the promising research effort that is currently being devoted to transmitting clock signals across an optical medium [85]. The received source-synchronous clock at the reception side of the NI is then used to drive the de-serializers and, after a clock divider, the front-end of the DC FIFOs. We assume that a form of clock gating is implemented, so when no data is transmitted, the optical clock signal is gated.

Another typically overlooked issue is the backpressure mechanism. We opt for credit-based flow control because credit tokens can reuse the existing communication paths. Besides, the low dynamic power of ONoCs can easily tolerate the signalling overhead of this flow control strategy. Credits are generated at the reception side of the NI when a flit leaves the DC FIFO (at the processor frequency) and forwarded to the transmission side so that they can be sent back to the source (at the NI frequency). To synchronize between different frequency domains, we used brute force and mesochronous synchronizers.

7.4. Baseline Electronic NoC

We include in our tests an aggressive electronic NoC with a typical configuration that can be used for reference. It is the consolidated ×pipesLite architecture [139], which represents an ultra-low complexity design point. Each 32-bit switch includes several VCs with 5 slots each to avoid message-dependent deadlock, implemented by replicating the VC-less switch [45]. It takes one cycle to traverse the switch and one cycle to traverse each link.

The network interface consists of two parts [79]. The first one is a packetizer, which acts as protocol converter between the IP core and the network. This block is also required for the ONoC, so it is not considered in this comparison framework. The second one is the buffering stage. In order to preserve the generality of the design and support cores with different operating frequencies that access an ENoC with fixed common frequency, dual-clock FIFOs have been included at the electronic network interfaces, similar to the ONoC NI design. However, in this case all DC FIFOs have 5 slots at both initiator and target side, because round-trip latency does not require larger buffers for maximum throughput operation.

7.5. Methodology

Our NI can work with any WRONoC topology. Without lack of generality, for this evaluation we model a wavelength-routed ring inspired by [82] implemented on an optical layer vertically stacked on top of the baseline electronic layer (note that this is not the optimized version introduced in Chapter 6). To obtain accurate latency results, we implemented detailed RTL models of the optical and electronic network interfaces and NoCs using SystemC. We instantiate a 4x4 2D mesh for the electronic NoC, and a similar system connected by the optical ring. The network-wide focus, well beyond the NI, aims at relating NI quality metrics to network ones. Delay values for the optical ring have been backannotated from physical-layer analysis, and have been differentiated on a per-path basis.

For power modelling, every electronic component has been synthesized, placed and routed using a low power 40 nm industrial technology library. Power metrics have been calculated by backannotating the switching activity of block internal nets, and then importing waveforms in the *PrimeTime* tool. We have applied clock gating to achieve realistic static power values. Energy-per-bit has been computed by assuming 50% switching activity. The photonic components and values are listed in Table 7.1. Table 7.2 sums up the static power and energy-per-bit for all the electronic and optical devices, with 3 and 4-bit parallelism. For the fast developing optical technology, we consider a coherent set of both conservative and aggressive parameters [16, 15].

7.6. Initial Evaluation

Before integrating the ONoC into a real system, we test the latency of simple read and write transactions in order to get an initial understanding of the potential of this technology. The traffic the ONoC will need to support in a real system will be simply a combination of these simple transactions.

7.6.1. Latency Breakdown

Figure 7.3 presents the latency breakdown for the NI components and the ONoC, obtained from our accurate RTL-equivalent simulations. We clearly see that the latency of the network is negligible, but it requires support from a time consuming NI. Inside the NI, the DC FIFOs are the components with the largest latency.

Table 7.1: Photonic components and parameters with their values with aggressive and conservative technologies.

Parameter	**Cons. tech.**	**Aggr. tech.**
Coupler loss	0.46 dB	0.46 dB
Modulator insertion loss	4.0 dB	4.0 dB
Photodetector loss	1.0 dB	1.0 dB
Filter drop loss	1.0 dB	1.0 dB
Through ring loss	0.0001 dB/ring	0.0001 dB/ring
Propagation loss	1.5 dB/cm	1.5 dB/cm
Bending loss	0.0005 dB	0.0005 dB
Crossing loss	0.52 dB	0.18 dB
Wall-plug laser efficiency	8%	20%
Thermal tuning	20 uW/ring	20 uW/ring
Transmitter (dyn. energy)	50 fJ/bit	20 fJ/bit
Transmitter (fixed energy)	0.100 mW	0.025 mW
Receiver (dyn. energy)	25 fJ/bit	10 fJ/bit
Receiver (fixed energy)	0.150 mW	0.050 mW

Table 7.2: Static Power and Dynamic Energy of Electronic and Optical Devices.

HARDWARE COMPONENTS	**3-bit parallelism**			**4-bit parallelism**		
	count per NI	**STATIC POWER (mWatts)**	**DYNAMIC ENERGY (fJ/bit)**	**count per NI**	**STATIC POWER (mWatts)**	**DYNAMIC ENERGY (fJ/bit)**
DC_FIFO 5slots (TX)	3	0.12	10.65	3	0.12	12.72
DC_FIFO 5slots (RX)	30	0.12	8.54	30	0.12	10.2
DC_FIFO 15-17 slots	15	0.12	26.50	15	0.12	31.65
DEMUX1x3	1	0.000725	0.92	1	0.000725	0.92
DEMUX1x15	3	0.0021	25.21	3	0.0021	25.21
DEMUX1x4	15	0.00056	6.72	15	0.00056	6.72
MUX4x1 + ARB	15	0.08	0.36	15	0.11	0.49
MUX45x1 + ARB	1	0.9	5.09	1	0.9	5.09
SERIALIZER	45	0.0475	9.41	60	0.0417	2.63
DESERIALIZER	45	0.0289	7.74	60	0.0281	6.12
MESO-SYNCHRONIZER	45	0.041	8.00	45	0.0565	11.1
COUNTER 2bits	45	0.01482	1.014	45	0.01482	1.014
BRUTE FORCE SYNC	15	0.004234	1.4	15	0.00503	1.66
CLOCK DIVIDER	15	0.01172	0.6	15	0.0139	0.714
TSV	120	/	2.50	150	/	2.50
TRANSMITTER aggressive	60	0.025	20	75	0.025	20
TRANSMITTER conservative	60	0.100	50	75	0.100	50
RECEIVER aggressive	60	0.050	10	75	0.050	10
RECEIVER conservative	60	0.150	25	75	0.150	25
THERMAL TUNING/RING $20°K$	180	0.020	/	225	0.020	/
LASER POWER aggressive	/	0.0421	/	/	0.0525	/
LASER POWER conservative	/	0.308	/	/	0.385	/
E-SWITCH (3VCs)	/	17.9	193	/	17.9	193

7.6.2. Testing Simple Transactions

Figure 7.4 shows the zero-load latency of simple read and write transactions. A read involves sending a request message followed by a reply with the data, and a write simply sends the data to the desired destination. Single reads and writes have 64-byte data messages, while burst transactions carry 512 bytes. The figure includes results for the ONoC with all the combinations of flit width (32 and 64 bits) and bit

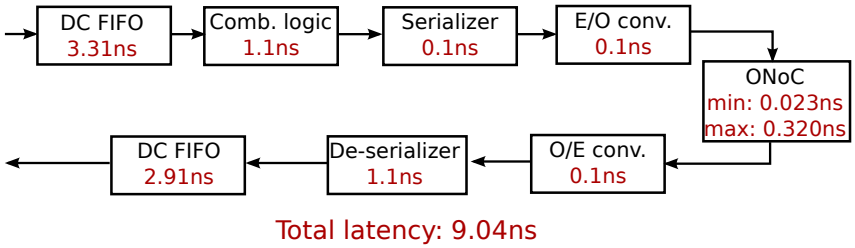


Figure 7.3: Latency breakdown of the optical NI with 3-bit parallelism and the optical ring.

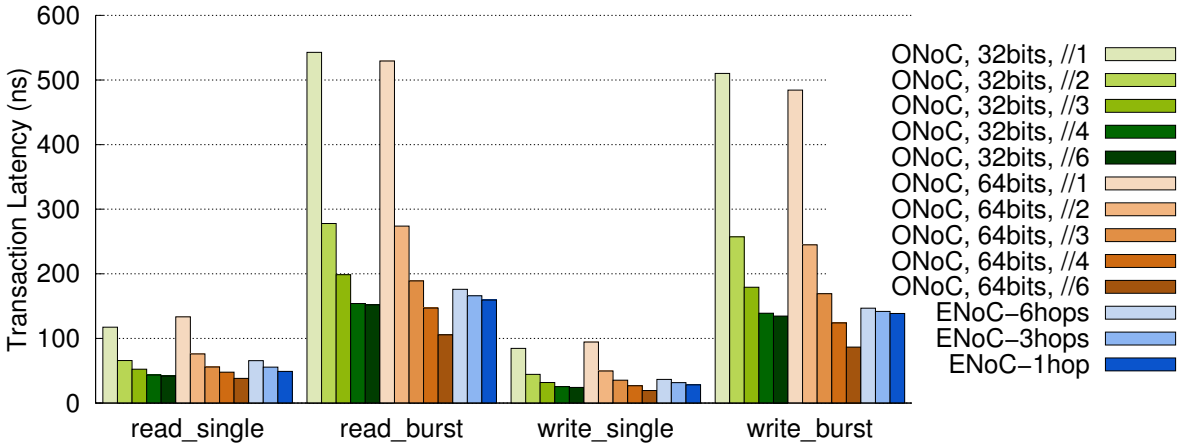


Figure 7.4: Optical Network Interface Architecture with 2 virtual channels for 3-bit parallelism

parallelism (1, 2, 3, 4, and 6), and for the ENoC with minimum, maximum, and average path lengths.

The ONoC latency decreases as we increase bit parallelism because flits are sent faster thanks to the parallelization of the serialization process. However, this improvement saturates when increasing from 4 to 6 bit parallelism with 32-bit flit width, because we are already transmitting flits faster than we produce them, at the core frequency. When we move from 32 to 64 bits/flit, each message is divided into a smaller number of flits, but it takes twice as long to serialize and send each of the flits. If we compare the results at each bit parallelism, we notice that the latency is slightly better with 32 bit parallelism for single transactions, but slightly worse for burst transactions. The frequency inside the NI is twice as high with 32 bits/flit than it is with 64 (we need to serialize half of the bits). Due to the way the DC FIFO synchronizes the front and back-ends, this means that flits go through the DC FIFOs at the transmission side faster with 32-bit flits, giving those configurations a clear advantage. However, there is another effect we must consider when analysing the burst transactions. In every flit, there is a fixed number of management bits (flit type and VC identification) that can't be used to send useful data. Therefore, 64-bit flits are more efficient in carrying data, because they have a better ratio of useful bits over management bits. This becomes relevant when we have to send larger amounts of data and overrides the positive effect of the higher NI frequency of 32-bit flits, resulting in shorter latencies for the 64-bit configurations.

If we compare the ONoC vs. the ENoC for single transactions, we notice that the latency of the ONoC with 32 bits/flit and 3-bit parallelism is comparable to the latency of one-hop communications in the ENoC and shorter than the average latency. However, with burst transactions, the bandwidth of the network becomes more relevant, and we need to move up to 4-bit parallelism for the 32-bit ONoC to be as fast as one-hop communications on the ENoC. As we will see in the following chapter, increasing the bit parallelism involves higher power consumption, so we will need to consider it to find the best design point.

Chapter 8
Case Study: Optical Networks-on-Chip for Memory-Coherent CMPs

Summary

This chapter presents the integration of a complete optical on-chip network including a realistic network interface into a real platform: a chip multiprocessor. We compare performance, static power and energy of the optical NoC with its electronic counterpart. Our focus is not just on the performance of NoC read and write transactions, but rather on their aggregation into higher-order operations relevant for the system at hand. As a result, we consider the communication patterns generated by a coherence protocol for the CMP.

8.1. Introduction

Unfortunately, the extended experimental evidence of the benefits of optical networks-on-chip has still not translated into a stabilization of roadmaps for the industrial uptake of this on-chip communication technology. This consideration is further exacerbated by the high cost of introducing it, and by the far-from-consolidating maturity of basic optical components. Besides, the projected results are overly optimistic due to optimistic technology assumptions, use of logical topology designs instead of physical ones, and overlooking static power [15]. Fundamentally, the main challenge consists of showing a compelling advantage (if any) for on-chip nanophotonic interconnection networks (ONoCs) with accurate modelling assumptions, while meeting the requirements and operating conditions of real-life user devices and workloads.

We replace the traditional electronic mesh of a realistic cache-coherent chip multiprocessor with a wavelength-routed optical ring, and compare performance and power for the two versions. We include in our simulation framework the modelling of the NI architecture (as presented in Chapter 7), in an attempt to validate whether (and to what extent) the projected benefits of optical NoCs over their electrical counterpart are still preserved with the NI in the picture. Our NI design takes into account important interdependent issues such as end-to-end flow control, need for virtual channels, buffer sizing, clock re-synchronization, and serialization ratio. The baseline electronic NoC is the optimized version introduced in Section 7.4.

8.2. Architecture of the Chip Multiprocessor

The limited amount of instruction level parallelism makes chip multiprocessors (CMPs) a better resource utilization strategy than traditional superscalar processors [105]. CMPs include multiple simple processors to efficiently exploit thread level parallelism, and can work at high frequencies. Besides, from a fabrication point of view, it is easier to replicate several simple processors, than to build one complex processor.

We focus on a homogeneous chip multiprocessor with 16 cores, similar to the Tilera architecture [145]. Each core has a private L1 cache and a bank of the shared distributed L2 cache, both connected to a common NI through a crossbar. The system has directory-based coherence managed with a MESI protocol. It is essential to set several virtual channels for the different messages classes of the protocol, as we explained in Section 7.3 By analysing the dependency chains of the protocol and deadlock-free buffer sharing opportunities, we came up with a requirement of 3 VCs for deadlock avoidance.

8.3. Customizing the optical NI

The first modification compared to the generic design presented in Chapter 7 is the number of virtual channels. As we explained in the previous section, our coherence protocol requires 3 virtual channels. Therefore, the NI needs 3 DC FIFOs in the transmission side and 3 per source in the reception side.

To size the DC FIFOs, we considered also the size of the packets that would use each of the VCs. With 32-bit flits, control packets need 2 flits, while data packets need 21 flits. In Section 7.3 we explained how we sized the reception side buffers based on round trip latency. However, for the control VCs we decided to keep small 5-slot DC FIFOs because they can already fit two complete packets and we do not expect to send many back-to-back control packets with the target cache-coherence protocol. The FIFO depth will be assessed in the experimental results.

8.4. Evaluation

This section presents the latency and energy of the optical NoC in the CMP, following the methodology explained in Section 7.5. In this case, the optical ring is not an optimized design obtained by the algorithm

Table 8.1: Messages generated by the coherence protocol.

id	Event	Sequence of messages
P1a	L1 miss	**1.** Request from L1 to L2 **2.** Data reply from L2 to L1 **3.** ACK from L1 to L2
P1b/c	L1 write miss, 1/2 sharers	**1.** Request from L1 to L2 **2.** L2 sends data reply and invalidates 1/2 sharers **3.** Sharers sends ACK to L1 req. **4.** ACK from L1 to L2
P2a	L1 needs upgrade to write	**1.** Request from L1 to L2 **2.** ACK reply from L2 to L1 **3.** ACK from L1 to L2
P2b/c	L1 needs upgrade to write, 1/2 sharers	**1.** Request from L1 to L2 **2.** ACK reply from L2 to L1 and invalidates 1/2 sharers **3.** Sharers send ACK to L1 req. **4.** ACK from L1 to L2
P3	L1 write miss, another owner	**1.** Request from L1 to L2 **2.** L2 forwards request to owner **3.** Owner sends data to L1 **4.** ACK from L1 to L2
P4	L1 read miss, another owner	**1.** Request from L1 to L2 **2.** L2 forwards request to owner **3.** Owner sends data to L1 and L2 **4.** ACK from L1 to L2
P5	L1 replacement	**1.** Writeback from L1 to L2 **2.** ACK from L2 to L1

presented in Chapter 6, and power calculations do not include the laser distribution network. Results for an ENoC working at 1.2 GHz and configured with typical parameters from [139] are also included. These results have been published in [115, 111, 116, 112, 114].

8.4.1. Transaction Latency

We simulate the most common traffic patterns generated by a MESI coherence protocol in our RTL models without any contention. The increased accuracy of our analysis stems from the fact that our packet injectors and ejectors model actual transactions of the protocol, as well as their interdependencies. Table 8.1 describes the analysed compound transactions and Figure 8.1 presents the zero-load latency results. The messages included in these patterns amount to an average 98.8% of the total network traffic, as we observed from full-system simulations of realistic parallel benchmarks from PARSEC and SPLASH2 and multiprogrammed workloads built with SPEC applications in Chapter 4 (we only exclude communication with the memory controllers). Therefore, they are a very good indicator of the network latency improvements we can expect from the optical network, including its (non-negligible) network interface overhead.

We observe that in all the patterns except the last one, the ONoCs beat or obtain similar results to the ENoC with minimal path lengths: for the 32-bit ONoC, this is achieved with 3-bit parallelism or higher; for the 64-bit ONoC, we need to use 4-bit parallelism. With higher bit parallelism, the ONoC configurations achieve a substantial advantage over the ENoC. The differences between the 32 and 64-bit configurations come from the frequency of each configuration inside the NI, as it was explained in Section 7.6.2. The tendency changes in pattern 5 because the replacement packet is using a VC designed for control to transmit data, and the smaller FIFO cannot store enough flits to support the round-trip latency. However, these messages are only 4.7% of the total network traffic.

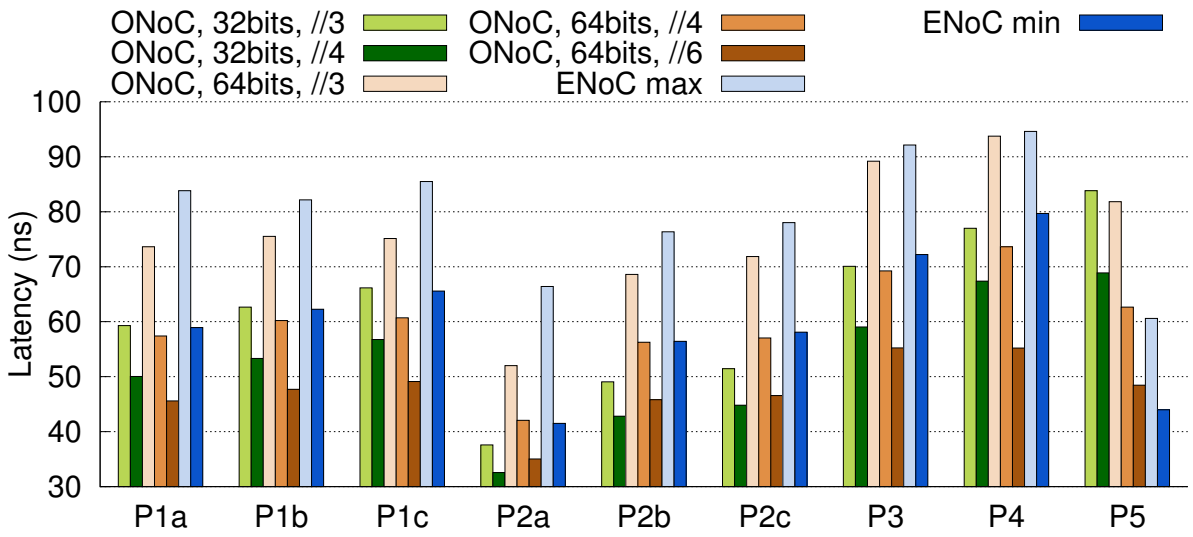


Figure 8.1: Latency of the most common communication patterns. For the ENoC, we include minimum and maximum paths.

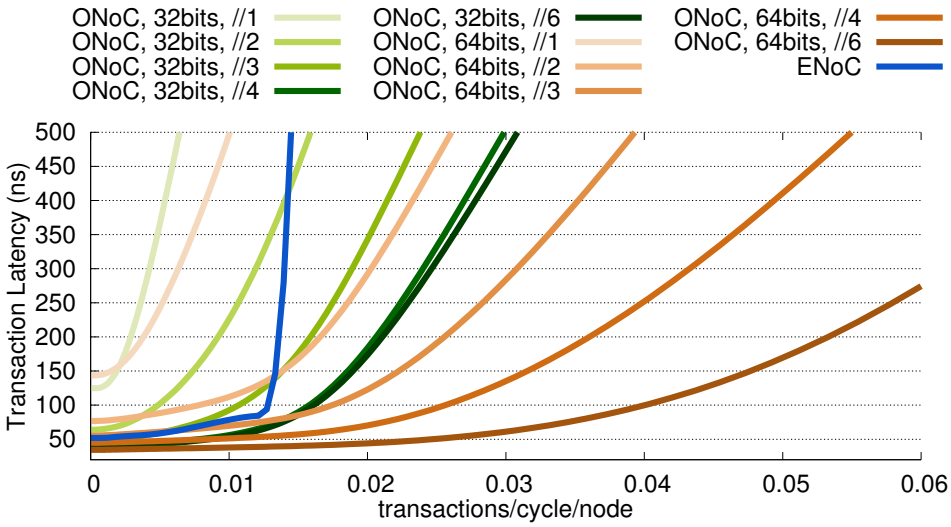
8.4.2. Uniform and Hotspot Traffic

Figure 8.2 shows the transaction latency for a request-reply pattern (the most common transaction in our chip multiprocessor) with uniform and hotspot traffic while increasing the injection rate. It also includes the ENoC latency as a reference point. Focusing first on uniform traffic, we clearly see that latency is shorter and it takes higher injection rates to saturate the network with larger bit parallelism. Comparing the 32 and 64-bit flit configurations at each bit parallelism, we see an interesting effect: latency is shorter with 32 bits/flit with low injection rates, but this trend changes when we increase the injection rate. As we explained in Section 7.6.2, when there is no contention in the network, the higher frequency inside the network interface gives the 32-bit configurations a clear advantage when going through the DC FIFOs. However, when the network gets congested, messages are queued up to enter the network interface towards the ONoC and also to leave the network interface after they've reached their destination. In this scenario, the reduced number of flits of each message with 64-bit flits shortens the waiting time and, therefore, the latency.

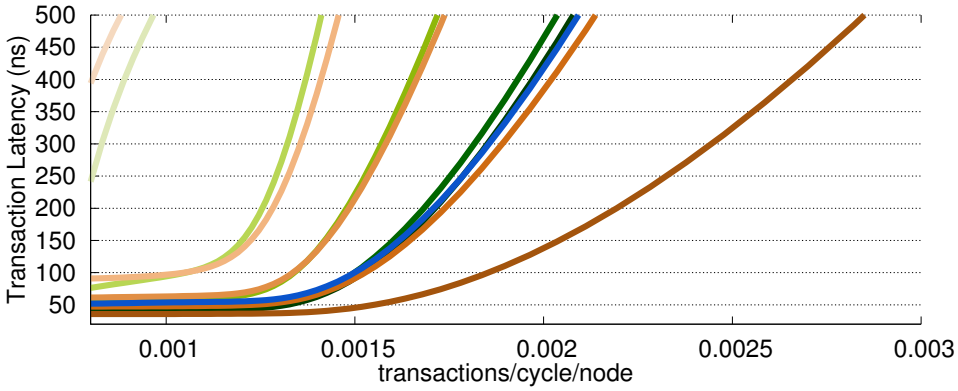
The flow control mechanism has a very relevant impact when the network is congested. Since we have one credit associated to each flit, we must send twice as many credits with 32-bit flits than with 64-bit flits. In our design, we group several credits on the same *credit flit* to optimize resources and help synchronization between the frequency domains. Based on the frequency inside the NI, more credits are grouped together with 64-flit configurations (at the same bit parallelism), reducing even more their negative impact. As we explained in Section 7.3, sending a credit involves stalling the data flits and delaying message transmission. Therefore, for all these reasons, the credit-based flow control penalizes the 32-bit configurations much more than the 64-bit ones, greatly reducing their performance with high injection rates.

The ENoC, included as a reference point, starts with lower latency than 3-bit parallelism ONoCs, but quickly saturates as we increase the injection rate and loses against almost all the ONoC configurations. This is because all the ENoC fabric gets filled with flits and they have to contend for resources at each hop, while the ONoC benefits from simultaneous contention-free all-to-all communication.

With hotspot traffic, all nodes are sending requests to the same L2 cache bank. In this case, the queue of data flits that need to leave from that congested node is the bottleneck of the system. Therefore, the most relevant metric is the bandwidth of the network, which determines the speed at which messages are sent. At 3 bit parallelism, the ONoC has a speed of 30 Gbps, which is less than the ENoC (38.2 Gbps, 32 bits per flit times 1.2 GHz). As a result, the 32-bit ONoC has worse performance than the ENoC. But when we increase to 4 bit parallelism (40 Gbps), the 32-bit ONoC achieves very similar results. As



(a) Uniform Traffic



(b) Hotspot Traffic

Figure 8.2: Transaction latency of a request-reply pattern with increasing injection rate.

we increase injection rate, the 64-bit flit configurations are again better than their 32-bit counterparts because of the reduced number of flits per message.

8.4.3. Buffer Size Exploration

In this section we analyse the effect of modifying the buffering of the optical network interface. We fix the flit width at 32 bits and bit parallelism at 3, and explore all the buffer size combinations detailed in table 8.2. Figure 8.3 shows how buffer size in the NI affects transaction latency, using a request-reply pattern.

We can extract the same conclusions from the uniform and hotspot traffic simulations. In case A, the minimum buffering has a very negative impact on performance, because data packets are stalled waiting for credits from the reception side FIFOs, which can only store 2 flits. This effect is slightly mitigated when we increase the buffer size for this VC to 5 slots in case B. Even though the DC FIFOs can achieve perfect throughput, backpressure is still preventing faster communications. We don't see any difference by increasing the size of control VCs in case C because the bottleneck is in the data VC. However, in case D, the reception side has been sized based on the round-trip latency and we achieve the maximum possible throughput. The larger buffers in cases E and F do not show any further improvements because the network is already using up all the bandwidth.

Table 8.2: Buffer sizes explored for the 3 VCs at each side of the NI. Note that the actual capacity of the DC FIFOs is one flit smaller than the number of slots.

id	Transmission side	Reception side
A	3, 3, 3	3, 3, 3
B	3, 3, 5	3, 3, 5
C	5, 5, 5	5, 5, 5
D	5, 5, 5	5, 5, 15
E	5, 5, 22	5, 5, 15
F	10, 10, 44	10, 10, 44

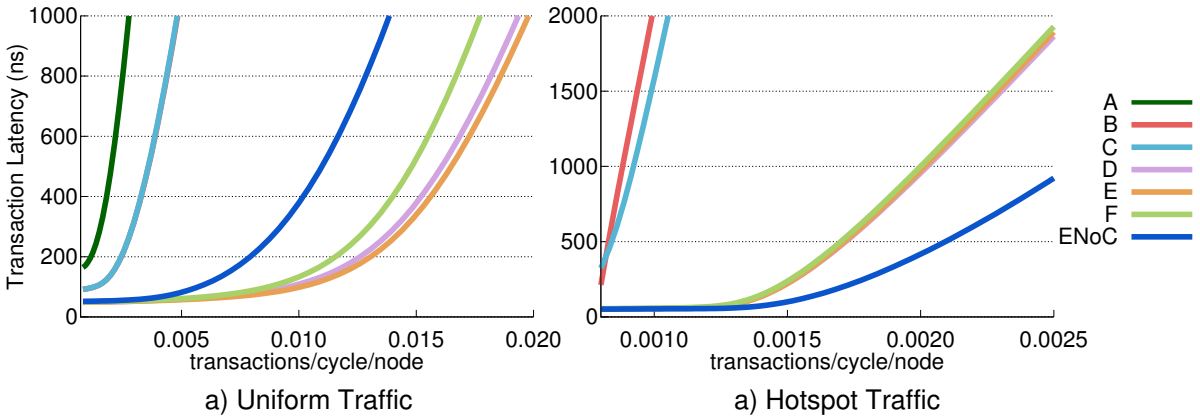


Figure 8.3: Transaction latency of a request-reply pattern with varying buffer sizes with uniform and hotspot traffic. The ONoC has 32 bits/flit and 3 bit parallelism. With uniform traffic, the line corresponding to case B (red) is hidden behind case C (light blue). With hotspot traffic, case A is not shown because the latency is much larger than for the rest of the configurations.

8.4.4. Power and Energy-per-Bit

Figure 8.4 depicts the static power and (dynamic) energy-per-bit for the ENoC vs. all the optical NoC configurations. We present a breakdown of the contributions of the NIs (electronic and optical components) and NoCs (the optical NoC is solely composed of laser power). We show the values for the two sets of optical parameters: conservative and aggressive.

Clearly, the ONoC consumes more static power than the ENoC, up to 360% more with conservative optical technology. This difference is reduced more than half with the aggressive optical technology. Focusing only on the ONoCs, we see that power increases with bit parallelism. This happens both in the NI, due to the higher frequency inside the NI and the larger amount of devices we need for parallel electro-optical and opto-electronic conversion, and in the NoC, due to the increased number of wavelengths. With a fixed bit parallelism, the difference between the ONoC with 32 bits/flit and the one with 64 bits/flit is very small because for larger flits, the power increase due to the bigger size of the devices is compensated by the slower frequency inside the NI.

We observe that the electronic switches dominate the static power in the ENoC, accounting for 95.8% of the total. However, this trend is reversed in the ONoC, with a contribution of less than 13% in all the configurations with aggressive technology. It is worth highlighting that a very large share of the static power of the electronic components in the NI comes from the DC FIFOs.

Figure 8.5 shows the energy-per-bit of the ENoC and ONoC configurations. For the ENoC, we include values for several path lengths. In this case, the ONoC has significantly lower energy-per-bit than the ENoC, which confirms the trend from previous literature. Apart from that, we still see how the main contributor for the ENoC energy is the NoC, while the NI carries all the complexity for the ONoC.

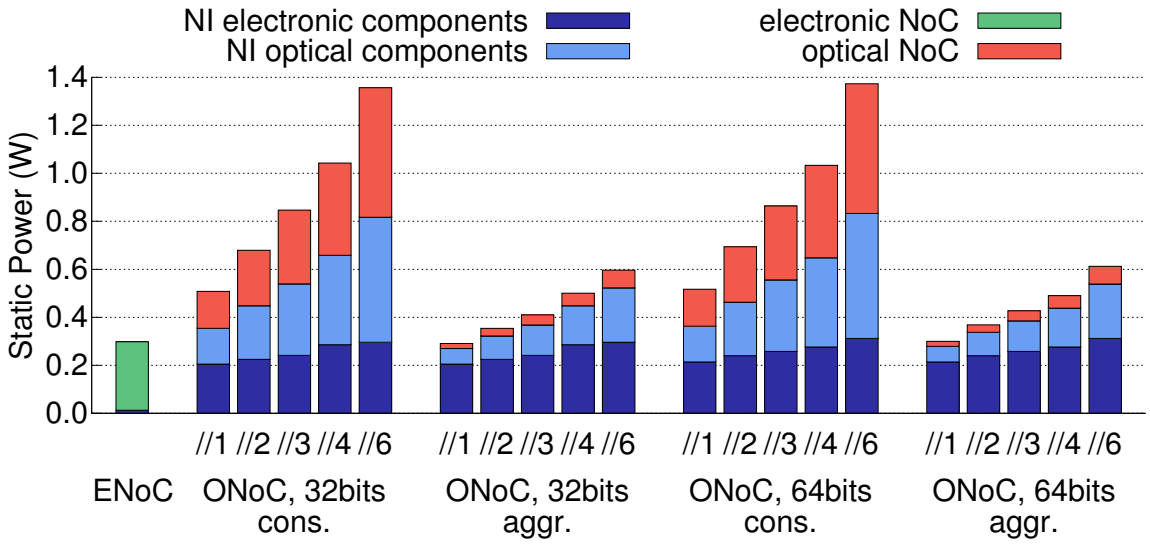


Figure 8.4: Static power of the NIs and the electronic and optical NoCs.

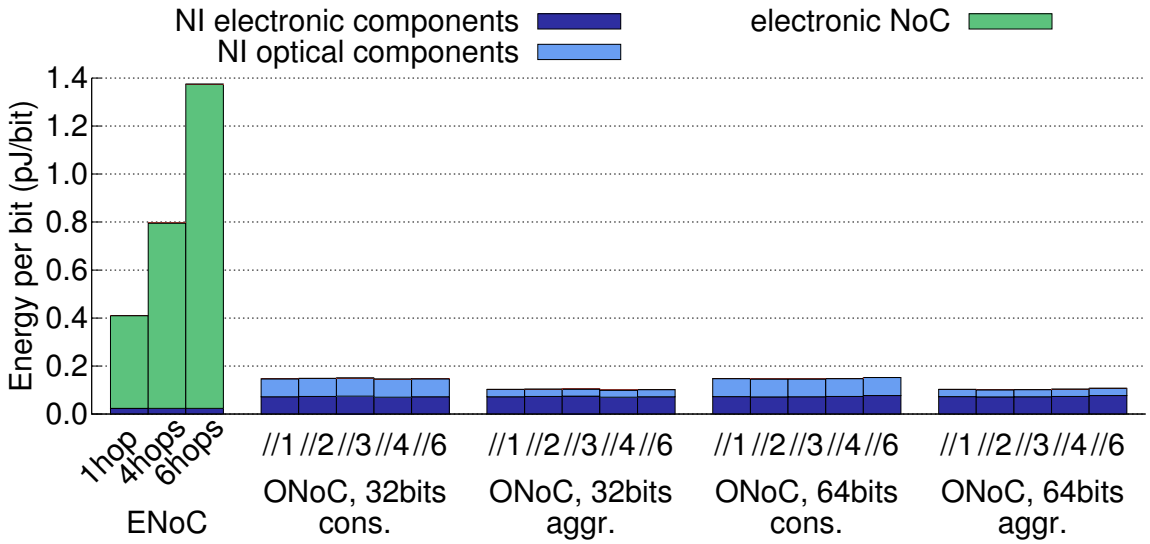
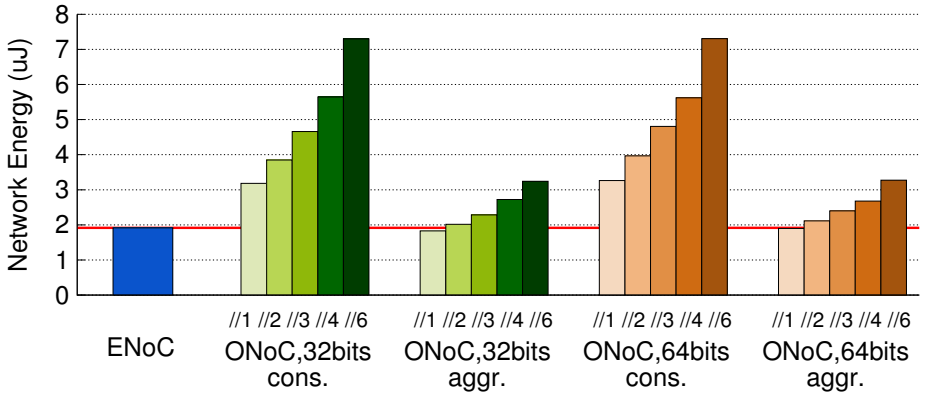


Figure 8.5: Energy-per-bit of the NIs and the electronic and optical NoCs. We present the energy to transmit a bit from a control flit, bits belonging to data flits need slightly more energy due to the different size of the reception side buffers.

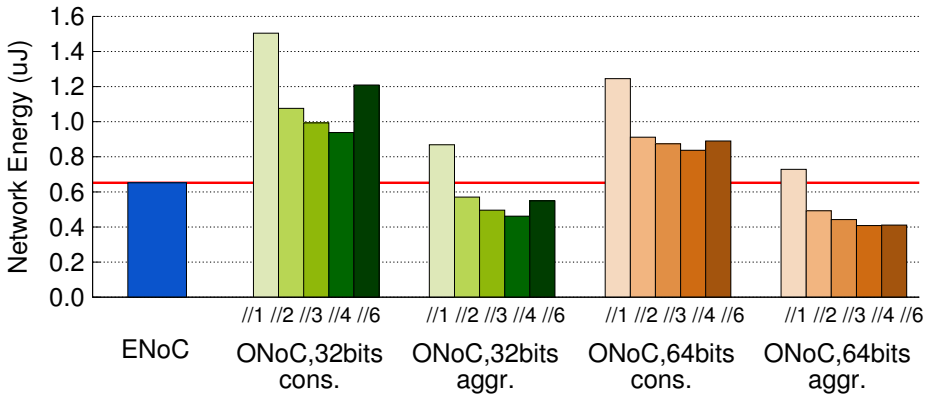
8.4.5. Network Energy

As we demonstrated in the previous section, the ONoC consumes more static power than the ENoC. However, the improved performance reduces the execution time and may result in overall energy reductions. In order to quantify this trade-off, we compute the network energy expended to execute a synthetic workload that consists on fixed number of random transactions (requests and replies) and a computation time between each one and the next. We explore two scenarios: one with light traffic (large computation time) and one with a more congested network (short computation time).

Figure 8.6 shows the results for all our configurations. We notice that, with light traffic, the ONoC does not save energy with respect to the ENoC. This is because the improvements in network performance don't have a big effect on execution time, due to the scarce use of the network. However, when we generate heavier traffic, we see that all the configurations with more than 1 bit parallelism and aggressive optical technology achieve significant energy savings. Network energy savings go from 13% to 37%, and these percentages will be even more pronounced when we consider the power of the whole system [79]. Therefore,



(a) Light traffic



(b) Heavy traffic

Figure 8.6: Network energy expended to execute a synthetic workload.

even though the ONoCs are not power efficient, their good performance makes them a very promising option to reduce system energy.

Finally, we notice that increasing the bit parallelism is worth it in terms of total energy only with heavy traffic, especially in the 32-bit configurations. The trend stops at 6 bit parallelism, because static power increases but the speedup stays the same due to the upper bound of the frequency set by the cores (as explained in Section 7.6.2).

8.5. Concluding Remarks

We apply our accurate design of NIs for WRONoCs to a realistic chip multiprocessor with nodes connected via an optical ring. This allows us to capture the effect of the NI on the most important network-quality metrics, and sets the scene for further comparative ONoC analysis. Regarding latency, the ONoC is always faster than its electronic counterpart even considering the NI, thus preserving the primary goal of a WRONoC. The behaviour under contention depends mainly on the available bandwidth of the interconnect technologies under test. For the WRONoC, such bandwidth can be modulated by tuning the bit parallelism, and adjusting buffer size to flow control requirements for maximum throughput operation. Similar tuning knobs do exist for ENoCs, namely flit width and buffer sizes. Therefore, the ultimate question is whether such tuning knobs are energy efficient in comparative terms, which depends on the sensitivity of system performance to such knobs for the application at hand. This is left for future work.

When we consider power figures, we notice that, while switches are the main contributors in ENoCs, the NI has the largest share in ONoCs. For static power, this contribution is in the same order of magnitude than that from laser sources with conservative optical technology parameters. However, by further improving the optical technology, the role of the NI becomes dominant, thus making it the main target for future optimizations. Finally, the ONoC preserves its superior dynamic energy properties over its ENoC counterpart, even in the presence of its NI.

This work shows that the NI architecture should not be overlooked for realistic ONoC assessments, and comes up with new insights not provided by earlier photonic network evaluations. The most important one is that NI optimizations perhaps have higher priority over the relentless search for ultra-low-loss optical devices.

Chapter 9
Case Study: Augmenting Manycore Programmable Accelerators with Photonic Interconnect Technology

Summary

There is today a fundamental lack of compelling cases proving the superior performance and/or energy properties yielded by devices of practical interest when re-architected around a photonically-integrated communication fabric. This work takes its steps from the consideration that manycore computing platforms are gaining momentum in the high-end embedded computing domain in the form of general-purpose programmable accelerators (GPPAs). We analyse the performance and energy implications of augmenting these devices with an optical interconnect technology by using an accurate benchmarking framework against an aggressively optimized electronic NoC. We present two different approaches to incorporate the optical network into the GPPA: replacing the global network that connects all the GPPA nodes with an optical ring, and replacing the local network for intra-partition communication. In the second case, we present the first partitioning algorithm for wavelength-routed ONoCs, which minimizes the use of wavelengths thus reducing laser power consumption.

9.1. Introduction

This work aims at extending the feasibility analysis of optical interconnect technology when integrated into industry-relevant objects. In particular, the focus is on the high-end embedded computing domain, where photonic networks have already been proven to be promising for DRAM memory access [52]. We investigate a key component to sustain the performance-per-watt metric of embedded computing platforms as a candidate for photonic integration, namely a general-purpose manycore programmable accelerator (GPPA). In fact, driven by flexibility, performance and cost constraints of demanding modern applications, heterogeneous Systems-on-Chip (SoCs) are the dominant design paradigm in the embedded computing domain. SoC architectures and heterogeneity clearly provide a wider power/performance scaling, combining host CPUs along with massively parallel general purpose programmable accelerator fabrics. The latter hold the potential of bridging the gap between the energy efficiency (GOPS/W) of hardwired hardware accelerators and the computational power delivered by throughput computing. In contrast to graphics processing units, applicability of optical interconnect technology to GPPAs is faced with a more balanced trade-off between latency and throughput requirements, and by a different usage model of the manycore device. To our knowledge, this is the first time insights and guidelines are given to exploit optical technology in emerging GPPAs. The author of this thesis contributed to this work by integrating the optical network into the already existing GPPA platform.

9.2. GPPA Motivation

In the latest heterogeneous Systems-on-Chip (SoC), and even more in future ones, the quest for processing specialization to deliver ultra-high performance acceleration at reduced energy cost does not necessarily imply hundreds of dedicated hardware accelerators [90]. There are at least a couple of reasons against that approach. On one hand, the performance of a specialized processing engine may in many cases be equally achieved by the parallel computation of programmable processing units [38]. Execution efficiency can thus be achieved without sacrificing programmability. On the other hand, the trend towards simplifying the microarchitecture design of system building blocks is becoming increasingly strong. Only a replication-driven approach ultimately pays off in terms of design productivity.

There are two main architecture families that might in principle suit the need for manycore programmable accelerators: GP-GPUs [118] are optimized for the single instruction multiple data/thread execution model (SIMD/SIMT), while GPPAs rely on the multiple instruction multiple data (MIMD) model (although they are not limited to it). MIMD programmable accelerators do not implement GPU-like data-parallel cores, with common fetch/decode phases which imply performance loss when parallel cores execute out of lock-step mode. They are rather independent RISC cores, well suited to execute both SIMD and MIMD types of parallelism. When coupled with a hierarchical organization into clusters like [89, 97, 55], such accelerators lend themselves to powerful programming abstractions such as nested parallelism [93].

One reason for the growing interest in manycore accelerators in the embedded computing domain is that there is a rapidly growing demand for a new type of interactions between the user and the device, based on understanding of the environment sensed in multiple manner (image, motion, sound, etc.) striving to create more friendly user interfaces (augmented reality, virtual reality, haptics, etc.). Despite the good degree of data parallelism, parallel threads in this class of applications usually expose a behaviour which is heavily dependent on the local data content, resulting into many truly independent parallel computations [97]. In such a situation, GP-GPUs lose efficiency due to large divergence between threads.

The ultimate evidence of the practical and strategic relevance of the MIMD many-core acceleration template comes from industry, where similar architectures are being developed, for instance by Plurality (e.g., Hypercore Architecture Line, HAP) [89], or by STMicroelectronics (Platform 2012) [97]. Future evolutions of Kalray's manycore processors [64], Intel Single-Chip Cloud Computer [36], or Tilera's processors [145] could potentially follow the same trend, depending on market opportunities and device-level design choices.

The above motivations are at the core of this work's decision to investigate the potentials of optical

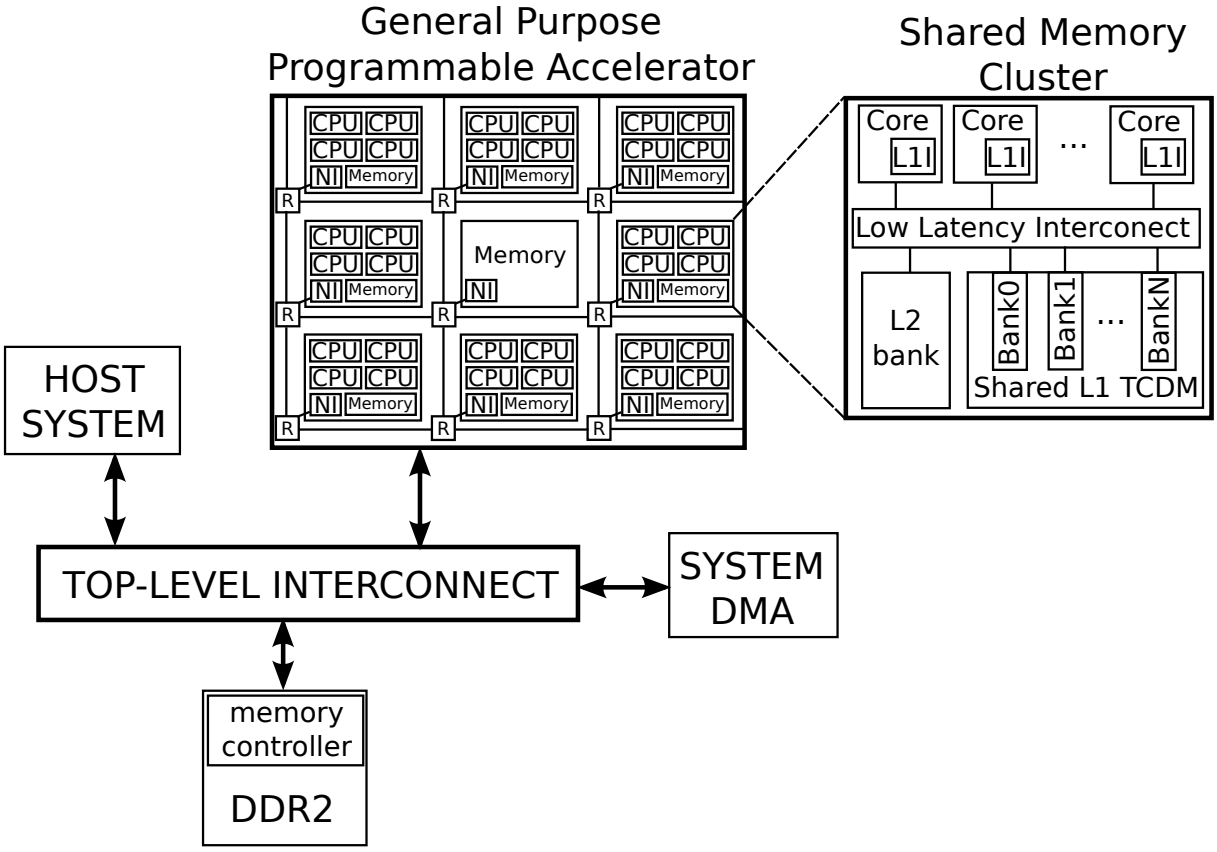


Figure 9.1: Heterogeneous (many-core accelerator-based) MPSoC architecture.

interconnect technology in the context of flexible MIMD/SIMD General-Purpose Programmable Accelerators for the high-end embedded computing domain.

9.3. Target Architecture

A common embodiment of architectural heterogeneity is a template where a powerful general-purpose processor (usually called the *host*), is coupled to a general-purpose programmable manycore accelerator (GPPA) composed of several tens of simple processors, where critical computation kernels of an application can be offloaded to improve overall performance/watt [97, 56, 54, 57]. Figure 9.1 shows a block diagram of such a system. The focus of this chapter is on GPPA manycore design, which we describe in details in the following subsections.

9.3.1. Cluster Architecture

The GPPA is a cluster-based many-core computing system. Clusters are the central building block of several recent many-cores [64, 89, 97]. These processors consider a hierarchical design, where simple processing units are grouped into small-medium sized subsystems (the *clusters*) sharing high-performance local interconnect and L1 data memory. Scaling to larger system sizes is enabled by replicating clusters and interconnecting them with a scalable medium like a network-on-chip. The simplified block diagram of the target cluster is shown in the rightmost part of Figure 9.1. It contains several simple RISC32 processor cores (typically up to 16), each featuring a private instruction cache. Processors communicate through a multi-banked, multi-ported Tightly-Coupled Data Memory (TCDM). This shared L1 TCDM is implemented as explicitly managed SRAM banks (i.e., scratchpad memory), to which processors are interconnected through a low-latency, high-bandwidth data interconnect. This is a very common design

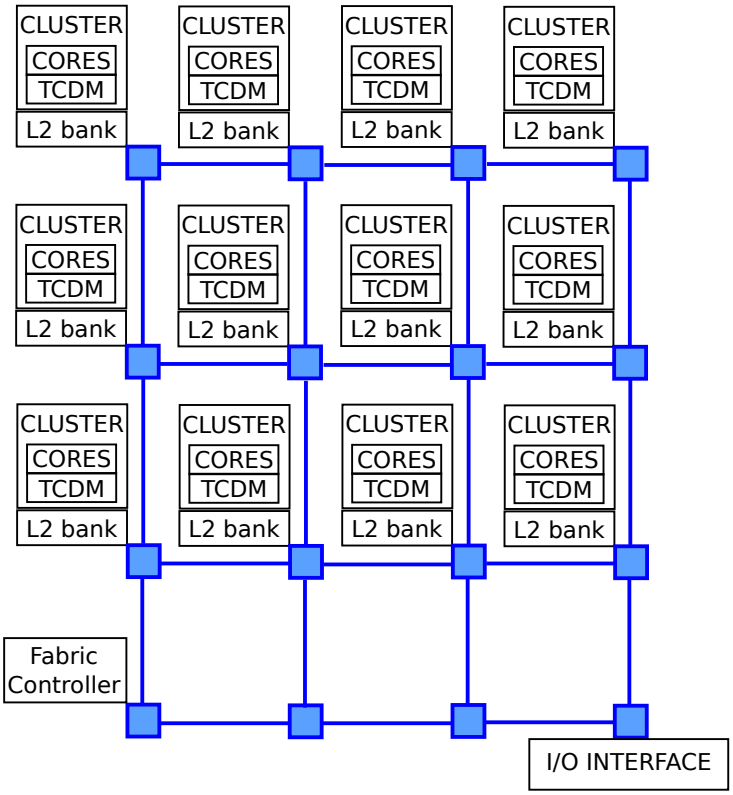


Figure 9.2: General-Purpose Programmable Accelerator Architecture.

choice for constrained embedded manycores, as the area and power overheads of hardware-managed caches (as compared to scratchpads) is very significant, and coherency protocols encounter severe scalability issues when interconnecting a large number of nodes.

Figure 9.2 depicts the global GPPA architecture. It consists of a configurable number of computing clusters (up to 12 in our setup), interconnected by a 2-D mesh network-on-chip. The topology of the NoC is a simple $n \times n$ mesh. Each of the first 12 nodes includes a computing cluster and an L2 bank. Another node hosts the "Fabric Controller", a special cluster instance with a single processor acting as a main controller for the whole many-core platform. This node interacts directly with the host system, and is in charge of the boot sequence of other clusters and their operation control. It has the fundamental role of managing NoC routing reconfiguration, setting up partitions and starting applications. Among the remaining three nodes, one switch is reserved to communications with an I\O interface (GPPA reading and writing ports), while the other two are temporarily left unused, and are available for future extension of the computation power.

Every full-cluster block is linked to a switch of the on-chip network with two network interfaces (NIs), a master and a slave, supporting OCP (Open Core Protocol). The master NI is dedicated to the core transactions, while the slave NI is used for accessing the internal cluster memory. Accesses to the L2 banks is possible through dedicated slave NIs.

9.3.2. Memory Architecture

Each cluster has an internal memory organized as private, per-core L1 instruction caches plus local L1 scratchpad data memory shared among all cores. The L2 memory is architected as a distributed shared memory, where each NoC router hosts an L2 bank. To minimize the probability of conflicts on a single L2 bank, contents are interleaved by line address. Overall, the memory system is organized as a partitioned global address space. Each processor in the system can explicitly address every memory segment: local

TCDM, remote TCDMs, L2, and main memory. Clearly, transactions that traverse the boundaries of a cluster are subject to NUMA effects: higher latency and lower bandwidth.

When the GPPA has to perform a new computation, the binary code is copied via global direct memory access (DMA) into the L2. Data is stored in main memory, where it is originally allocated by host programs. Permanently hosting entire data structures in the L1 TCDMs is not feasible, due to a limited size of 256 KB. The software must thus explicitly orchestrate data transfers from main memory to L1 or L2, to ensure that the most frequently referenced data are kept close to the processors. To enable performance and energy-efficient transfers, each cluster is equipped with a local DMA engine.

9.3.3. The Baseline ENoC Architecture

The GPPA network is built with *compound switches* that can be broken into two separate physical networks:

- A *Local Network* serves local traffic within GPPA partitions and guarantees traffic isolation across partitions [131]. Without lack of generality, we adapt from [41] proper synchronization mechanisms between communicating peers, so that it becomes possible to perform inter-cluster communication only through write transactions. Should this not be the case, then 2 VCs would be needed in the local network. This network implements overlapped static reconfigurations as a runtime reconfiguration mechanism for the routing function, thus enabling the dynamic management of partitions (setup, teardown, shape redefinition) [11].
- A *Global Network* supports global network-wide and I/O communication traffic while avoiding interference with intra-partition local traffic. Communication flows on this network are made up of both write transactions (code offload to the GPPA, data transfer from main memory into the GPPA local memory) and request/reply transactions (on an L1 instruction cache miss). Therefore, the global NoC includes 2 virtual channels (VCs) in order to avoid message-dependent deadlock. Following the design philosophy in [45], they are implemented by replicating the single VC-less switch twice. The replicated switches do not need any reconfiguration support because their routing functions are hardwired.

The ENoCs are overclocked (1 GHz) with respect to the speed of the processor cores (700 MHz). They thus require the use of decoupling dual-clock FIFOs between clusters and switching fabric, which are placed after the network interfaces. Such FIFOs also serve as a key enabler for the implementation of dynamic voltage and frequency scaling.

9.3.4. Usage Model

We present here the process that must be followed to execute code on the GPPA, which involves three steps:

1. Offloading Scenario. When a host application wants to offload a computational kernel on the GPPA, it needs to collect the code (the kernel executable) and data (e.g., pointers to data in main memory) into metadata structures that are forwarded to the fabric controller. The host processors initiate the copy of the kernel executable through the system DMA into the GPPA L2 memory. The cost of offloading computation to the GPPA should be kept as small as possible, otherwise it may completely hide all the benefits introduced by code acceleration.
2. Partitioning Scenario. To maximize the usage of the manycore accelerator we consider a scenario where multiple virtual machines are allowed to concurrently offload computation to the GPPA by creating isolated cluster partitions. The NoC disables communication between clusters belonging to different partitions.
3. Run Time Scenario. Once the offload and reconfiguration (partitioning) sequences are complete, the kernel executable is launched on the selected clusters. Upon program start all the involved cores experience cold instruction cache effects, which implies massive cache refill traffic. This is both a

latency-sensitive and bandwidth-sensitive operation (the first word of a burst read is sensitive to latency, while the rest of the burst is sensitive to bandwidth). All the cluster partitions have access to the whole L2 memory, with no affinity between clusters and their local L2 banks. While this is prone to NUMA effects, it allows better usage of L2 memory space, as no a-priori logic partitioning is done, which would lead to memory waste.

9.4. Replacing the Electronic Global Network with an Optical Ring

In this Section we explore the benefits of replacing the global electronic network of the GPPA with an optical ring. We carry out a performance characterization of system operations with the hybrid interconnect fabric, and benchmark it against a competitive electrical baseline. Our focus is not just on the performance of NoC read and write transactions, but rather on their aggregation into higher-order operations relevant for the system at hand (e.g., computation offload). As a side effect, performance of such operations is not just determined by the system interconnect, but rather by the cooperation of several components (e.g., the DRAM subsystem, DMA architecture, and memory hierarchy). This work captures such interdependency. This work has been published in [10, 114].

The ONoC architecture is designed by following a cross-layer design methodology, where the quality metrics of the selected design point include awareness of the degradation effect of place&route constraints over insertion loss, and the overhead of the upper layers of the optical network interface beyond the domain conversion circuits (e.g., buffering, flow control, virtual channels, synchronization) as presented in Chapter 7. The optical ring implemented in this architecture is not the optimized version obtained from the algorithm presented in Chapter 6, and we do not include the power of the laser distribution network.

Energy efficiency figures are provided by accounting for the execution time of real-life workloads, for a parametric set of quality metrics for the fast-evolving optical devices, and for the energy of electrical components on a 40nm low-power industrial technology library (which makes the electrical counterpart extremely competitive).

9.4.1. Customizing the Optical NI

Aware of the difficulty to make the case for a purely optical interconnect fabric, we conservatively and realistically come up with a hybrid architecture: the optical network replaces the global network of the manycore accelerator, while the local network remains electronic.

We have customized the optical NI to work with the master and slave NIs of the GPPA. As we explained in Section 9.3.3, the global network needs 2 virtual channels. The behaviour of the system NIs allows us to remove the initial demultiplexer in the transmission side of the optical NI, connecting directly the master to VC 0 and the slave to VC 1. Following the same idea, the arbitration and multiplexing at the end of the reception side have been specialized to send flits to the appropriate NI and to avoid conflicts with flits arriving from the local ENoC.

9.4.2. Evaluation

This section presents the experimental results for the code offload and the power analysis. The rest of the communication scenarios are included in the last section, which tests real applications. We explore up to 4-bit parallelism for the global ONoC and always use 32 bits/flit. The whole GPPA system with the NoC variants has been modelled and simulated with cycle accuracy in RTL-equivalent SystemC by augmenting the baseline VirtualSoC simulation environment [22].

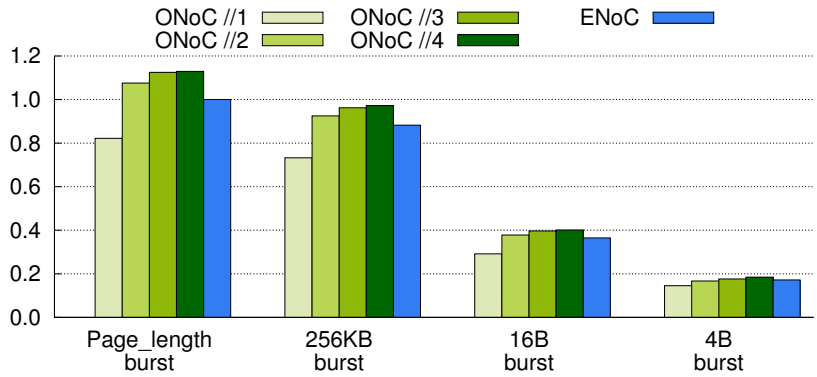


Figure 9.3: Offload bandwidth as a function of DMA burst size, normalized to the ENoC with page-length burst.

9.4.2.1. Code Offload

The system DMA reads from main memory the code to offload by means of burst transactions of parametric length (from 4 bytes to the DRAM page size), and then writes it into the GPPA L2 banks. This system operation stresses the bandwidth properties of the interconnects under test.

Figure 9.3 shows normalized code offload bandwidth. For small 4-byte bursts, there is fundamentally no difference in performance among the NoCs under test. This is because transfers are slowed down by the latency to reach and access the off-chip L3 cache [1], which makes all other contributions negligible. As the burst size increases, this overhead is amortized over multiple code words. For the largest possible burst size, the 3-bit ONoC outperforms the baseline ENoC by roughly 13% in terms of offload bandwidth. However, with only 1-bit parallelism, it suffers a degradation of 18%. Finally, ONoC performance saturates with 3 and 4 bit parallelism, because performance is limited by the frequency of the cores are caches.

9.4.2.2. Power Analysis

In this section, the power of the NI master and slave is not consider, since it is common for all the networks under test.

Figure 9.4 compares the static power of the ENoC vs. the hybrid NoC with aggressive and conservative optical technologies. With a conservative optical technology, the ENoC is clearly more power efficient than the ONoC, regardless of the bit parallelism. This is mainly due to the power overhead of the optical devices, especially laser sources. In contrast, with an aggressive technology, the ONoC differs from the ENoC by only 4.7% with 2-bit parallelism.

Figure 9.5 shows the energy-per-bit comparison between the ENoC and the hybrid NoC. The results for the hybrid NoC are independent from the path length and are more energy efficient than ENoC (up to an order of magnitude) regardless of the specific optical technology. This confirms that, at least in terms of energy-per-bit, the ONoC definitely outperforms the ENoC.

9.4.3. Application Benchmarking

In this section, we compare the execution time between ENoC and hybrid NoC-based GPPA platforms for real workloads. Our benchmarks are two common computer vision applications: colour tracking, implemented from the open source computer vision library (OpenCV) for single colour tracking, and FAST [132], which is a corner detector for image feature extraction.

Without lack of generality, both applications are mapped on the whole GPPA, thus emulating a 12-cluster partition that cooperatively processes the computational task. The same code is executed on

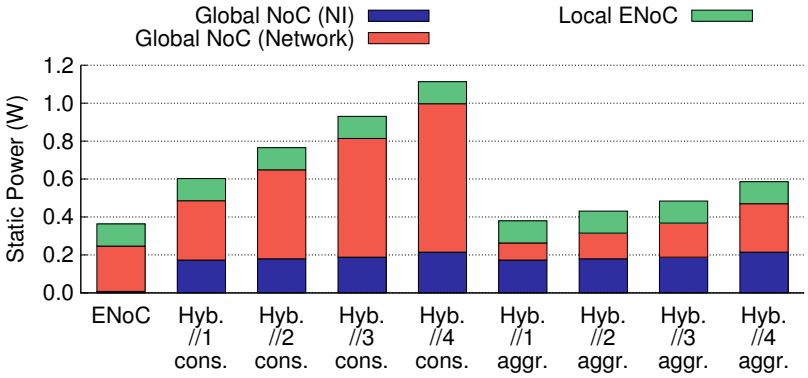


Figure 9.4: Static power for the ENoC vs. the hybrid ONoC variants.

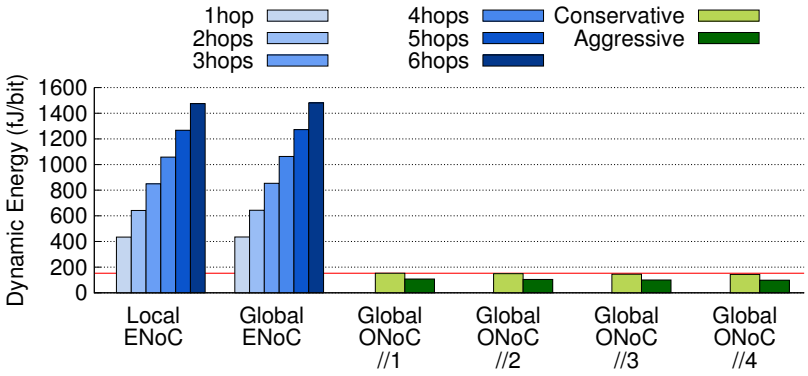


Figure 9.5: Dynamic energy for the ENoC vs. the hybrid ONoC variants under test. Results are broken down into the local and global networks, and the NIs.

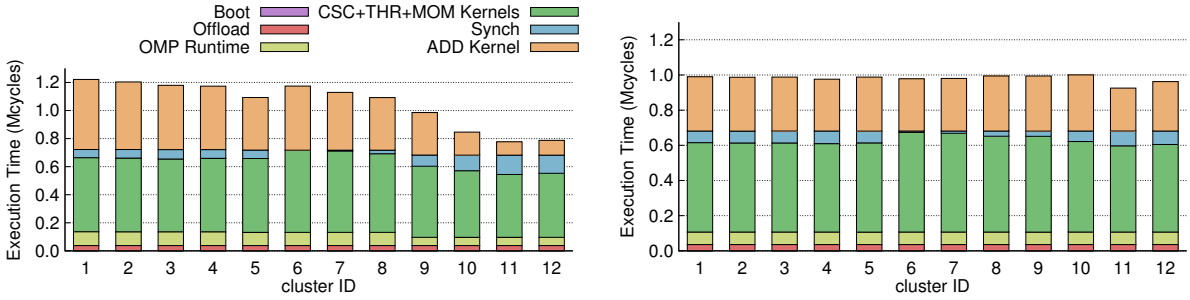
each cluster, but fed by different image portions.

The plots in Figure 9.6 compare the execution time spent on each cluster to perform the colour tracking on a single QVGA 24-bit input frame for the two platforms under test. The execution of this application is independent from the processing data, making NUMA effects more visible. Colour tracking consists of four kernels: colour space conversion (CSC), threshold (THR), moments computation (MOM), and pixel-wise addition (ADD). The first three kernels are merged in the results to achieve a better computation to communication ratio. The hybrid NoC improves the execution time by 18.1% with respect to the baseline ENoC. It also has better alignment of all the cores due the neutralization of the NUMA effects. This is more evident on the ADD kernel, which is memory dominated. Please note that the OMP contribution consists of the overhead for the OpenMP runtime environment [94], and that the offload time is considered as well.

Figure 9.7 depicts the same analysis on the FAST application. It consists of a single kernel that works using a stencil pattern of accesses. In this case, the execution flow depends on the actual pixel content, potentially leading to divergence between clusters. In this case, the hybrid NoC improves the execution time by 16.5%.

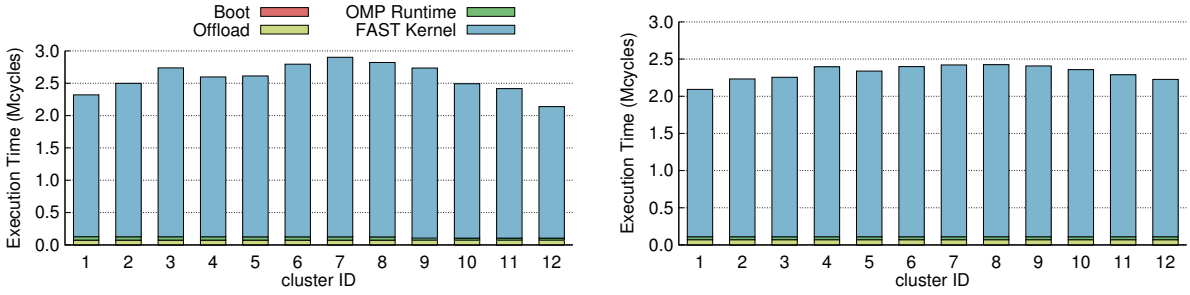
9.5. Replacing the Electronic Local Network with a Partitionable Optical NoC

Unfortunately, the advantages of wavelength-routed optical networks do not come for free, since the increase of communication actors causes the proliferation of the required laser sources. This involves



(a) ENoC **(b)** Hybrid NoC

Figure 9.6: Execution time for the colour tracking kernel in each of the 12 clusters.



(a) ENoC **(b)** Hybrid NoC

Figure 9.7: Execution time for the FAST kernel in each of the 12 clusters.

a significant static power overhead that may render the photonically-integrated system unaffordable. However, traditional benchmarking efforts fail to capture the most recent trends in the use of multicore hardware platforms. In fact, they do not consider that extracting massive task-level parallelism out of legacy software is not as trivial as integrating hundreds of identical processing elements on a modular hardware platform. Therefore, considering photonic-integration of multicore systems viewed as monolithic resources is not realistic in many domains.

In contrast, the most common usage model in those domains consists of their partitioning and concurrent exploitation by a number of applications running in parallel. The recent uptake of the virtualization paradigm by embedded devices is causing the need for such programmable accelerators to support multiple concurrent acceleration requests at the same time, the alternative being their inefficient time multiplexing. Partitioning of the computation/communication fabric to accommodate such offload requests is the most obvious way of meeting the concurrency requirement.

The key intuition behind this work is that partitioning of a photonically-integrated multicore computing platform is an opportunity for the optical fabric to make a more conscious use of its laser sources, hence relieving the static power concern of ONoCs. Especially, the wavelength-routing paradigm can take the most advantage of the partitioning usage model given its relevant use of laser sources to deliver contention-free all-to-all connectivity. In fact, wavelengths can be potentially reused across partitions, therefore permitting to power-off unused laser sources.

However, materializing power savings out of this idea is non-trivial. First, an online wavelength allocation algorithm should be designed in order to allocate the minimum number of additional wavelengths that meet the connectivity requirements of a new partition setup request. To the best of our knowledge, we present the first algorithm that smartly chooses the partition nodes to maximize wavelength reuse across partitions and reduce laser power. Second, the degree of wavelength reuse is topology-specific, since it stems from the matching between the partition setup request pattern and the truth table of

the wavelength-routed topology at hand. In this work, we assess the suitability of the most relevant wavelength-routed optical NoC topologies for use in partition-ready multicore processors. Our interest is in understanding whether the degree of wavelength reuse can offset power efficiency gaps between existing topologies. Third, when partitioning is a key requirement, one may think of fulfilling it with radical design choices, that is, by statically partitioning the network into predefined partitions. This implies a much more efficient physical design of partitioned topologies, which is offset by the lack of flexibility in the allocation of the partition size. This may cause either underutilization of resources or execution time penalties for those applications whose parallelism cannot be exploited by the available static partitions. This work has been published in [108].

9.5.1. Related Work

Partitioning isolation and reconfiguration technologies are sufficiently consolidated for ENoCs. Balboni *et. al* optimize the runtime reconfiguration of the routing function exploring the trade-offs between performance and implementation cost [11]. There are many publications in the recent literature that tackle this problem, both based on static reconfiguration [134] and dynamic reconfiguration techniques [91, 9]. An intensive research effort is currently under way in an attempt to find a suitable design point for chip implementations [124, 39].

There is no such reconfiguration technology for ONoCs. However, a few inspiring works do exist about wavelength reuse. This is done through network partitioning by assigning different message classes to disjoint network partitions, where the same wavelengths can be reused and the physical design is much simpler [129, 81]. There are also previous proposals that manage the laser power in order to reduce available bandwidth when the network is lightly loaded [29, 35, 86], but this idea has never been applied in the context of a dynamically partitioned system.

9.5.2. Customizing the ONoC and the GPPA

In the target architecture, we use wavelength-routing for the partition-capable network, although this requires some customization to work around the global connectivity it delivers. The optical network will provide intra-partition communications, and clusters will only be able to access the L2s inside their partition. In this case, we consider that the 16 nodes of the GPPA contain computational clusters and L2 banks. Communication with the off-chip memory ports is possible from every node by using four separate and cost-effective photonic buses with a different communication protocol: two buses for the requests from nodes to the two memory controllers using the multiple-writer-single-reader protocol [147], and two buses for the replies using the single-reader-multiple-writer protocol [70]. In this case, it is not cost-effective to use a laser-greedy WRONoC, as the alternative choice allows implementing all the required communication paths with only two wavelengths. The arbitration of the wavelengths for off-chip DRAM access is justified by the fact that accesses are bursty and sporadic, since they are aimed at uploading or downloading processing data onto/from the GPPA. If more wavelengths are required in order to increase memory access bandwidth, this can be easily delivered by spatial-division-multiplexing (i.e., by increasing the number or waveguides) rather than by increasing the number of wavelengths. In any case, should we need more wavelengths for memory access, the work in [28] suggests not to overload optical power waveguides with too many splitters. Therefore, it is reasonable to conceive dedicated laser sources for the off-chip memory network and dedicated sources for the partition-capable network. The latter is the explicit target of our optimization.

In this work, we test several well-known WRONoC topologies for inter-node communications in the partition-capable photonic NoC: the λ-router [103], the GWOR [143], and several ring variants inspired by [82]. However, we take a radically different perspective to their comparative analysis: their suitability for laser source reuse in the context of a partition-enabled multicore architecture.

9.5.3. Dynamic Partitioning

WRONoC topologies are designed with enough wavelengths to guarantee all-to-all communication. However, the GPPA must allocate isolated partitions to service several requests concurrently. This means that, at any given moment, many of the communication paths that are implemented in the chip will not be used. If we choose the nodes that compose each partition so that intra-partition communications reuse wavelengths as much as possible, we will have several unused wavelengths and will be able to power them off.

Our wavelength-reuse methodology is based on a distinctive property of optical NoCs, experimentally verified with real workloads in [10]: an optical transport medium is capable of smoothing out non-uniform memory access (NUMA) effects. That is, access latencies to distributed L2 memory banks are almost position-independent on a 2D mesh of processing elements. The practical implication is that, while with an electronic NoC partitions should group cores that are physically located close to each other, with ONoCs the notion of locality does not make sense. Hence, efficient partitions may be set up by grouping cores that are physically placed far apart from each other. This is a relevant degree of freedom that our methodology exploits to come with partition configurations that optimize the degree of wavelength-reuse.

9.5.3.1. Basic Idea

Let us consider Figure 9.8, where the truth table of an 8x8 gwor topology is illustrated. This topology does not deliver self-communication, which is then assumed to be implemented via an electronic shortcut. The topology delivers connectivity to a 3x3 array fabric of computation clusters, where one core serves as the fabric controller, hence cannot take part to any partition. Let us assume that a background partition is instantiated, including clusters 0 and 7, which use $\lambda 7$ as their communication wavelength (from 0 to 7 and vice versa). At this point in time, $\lambda 7$ is the only powered-on laser source. Let us then assume that a new partition has to be activated, consisting of two computation clusters. Figure 9.8 illustrates two options. In the first one, clusters 2 and 5 are activated based on a smart selection policy where they keep making use of the same $\lambda 7$ for their inter-cluster communication. Instead, should the runtime manager select clusters 1 and 2, then laser sources $\lambda 1$ and $\lambda 6$ would need to be activated to deliver intra-partition communication. In this sub-optimal case, 3 laser sources would be on at the same time after the second partition is set up. Clearly, it is possible to come up with a partition allocation algorithm that can meet connectivity requirements while making a conscious use of laser sources.

9.5.3.2. Greedy Algorithm

We propose a greedy algorithm to allocate partitions of any number of nodes in real time. Before executing the algorithm to service a new request, we have a set of already allocated nodes and wavelengths for the existing partitions. The set of already allocated wavelengths always includes the two that are used for communication with the memory controllers. The greedy algorithm follows several steps to generate the new partition:

1. Randomly choose a free node to start the partition.
2. Until we have the desired number of nodes in the new partition, we keep adding nodes following these steps:
   a) Try to reuse the allocated wavelengths to add a new node to the partition.
      - First, we find the free nodes we can reach from the nodes in our partition using already allocated wavelengths.
      - Out of those free nodes, we select only the ones that require minimum number of extra wavelengths for full connectivity inside the partition.
      - If there are several free nodes that are equally good, we choose randomly among them.

   b) If there are no nodes reachable from the partition with the allocated wavelengths and we still have just one node in the new partition, we try to find one that satisfies the *symmetric property*: the same wavelength is used to communicate the two nodes in the two directions.

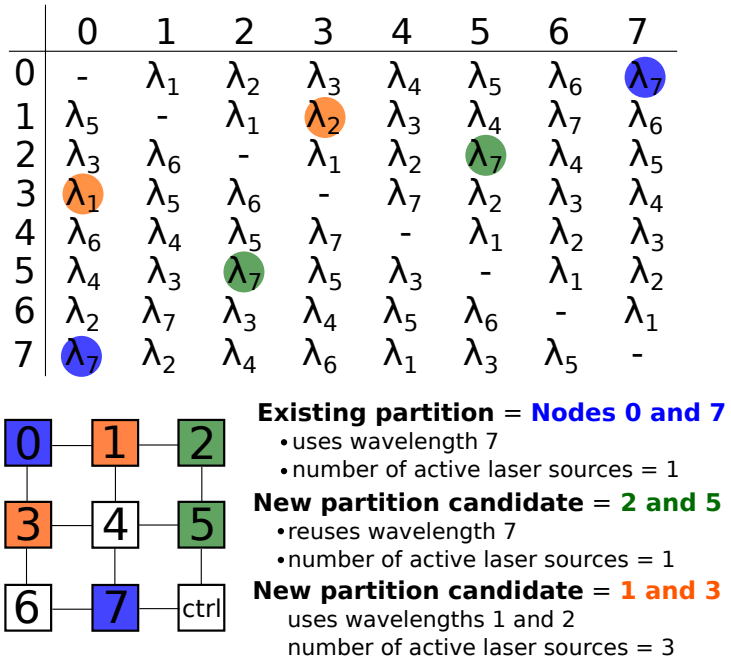


Figure 9.8: Truth table of the 8x8 gwor and basic example to set up partitions with and without wavelength reuse.

c) If the previous points failed, we simply choose a free node randomly.

After choosing the next node, we add to the allocated wavelength list all the wavelengths needed for full connectivity in the new partition.

The algorithm takes a locally optimal decision at each step, and never backtracks. The complexity of the algorithm is $O(n^2)$, which makes it perfectly within reach of online execution. The actual execution time depends on the chosen processor and its internal parallelism. We will only choose to apply the algorithm if its overhead is compensated by the execution time of the request. In our experimental setup, we consider the time to run the algorithm negligible, and demonstrate in Section 9.5.6.4 that its application would be cost-effective up to a 45% overhead with respect to the request execution time.

9.5.3.3. Exhaustive Search Algorithm

As a high performance alternative, we also introduce an exhaustive search algorithm that finds the best possible partition for every new request. This algorithm always finds a partition that minimizes the number of allocated wavelengths. Its use on a real system is infeasible due to the high complexity and execution time, but we include it as a comparison point. The algorithm checks all the possible combinations of free nodes to build the requested partition, and then chooses the one that results on a system with minimal number of wavelengths. If there are several options that are equally good, it randomly chooses one of them.

So far, the two algorithms do their best to service the current request. However, the decision for the current partition may affect future partitions. In the exhaustive search, we include two optimizations to choose the best option among all the ones with equal number of wavelengths and improve long-term results:

- Maximize wavelength reuse. We prioritize wavelengths that are already being used in several partitions. This way, we will still have large wavelength-reuse values after we remove a partition.
- Minimize wasted wavelengths. We characterize a "wasted wavelength" as an allocated wavelength that is used to communicate nodes inside a partition with nodes outside the partition. These communication paths will never be used, reducing the opportunities to reuse this wavelength in new isolated partitions.

Note that these optimizations cannot be applied to our greedy algorithm, where nodes are added one by one. If we applied it when adding a node, we would reduce the reusing opportunities to add the next ones.

9.5.4. Static Partitioning

In the previous section, we started from a fully connected optical network and dynamically set partitions on top of it. We now explore a different option: partitions statically built on the chip at fabrication time. This option lacks the flexibility of the dynamic partitioning to accommodate requests of any size, but gives us the opportunity to design very power efficient partitions that reuse a minimum number of wavelengths. In practice, it means having several smaller and independent ONoCs instead of a single big one.

We must carefully decide the number of partitions to build and their size, because it will not be possible to modify them later on. We analyse the request trace and extract the most common partition size and the most useful partition mix. We decide to test two different static configurations: one with homogeneous static partitioning (4 partitions of 4 nodes each, 4 being the weighted average partition size), and one with a mix of static partitions (4 partitions of 2, 4, 4, and 6 nodes, respectively, because this is the mix that would best fit all the partition mixes we observe over time). All the small networks are built with optical rings, and the same wavelengths are reused as much as possible across them.

This radical design choice is fully compatible with modern programming models. In fact, the notion of cluster-based manycore accelerators is now central in two very representative examples: OpenCL [68] and OpenMP [106]. Both of them ensure portability among different accelerator targets by allowing the runtime system to map the user request to a smaller number of physical resources. With the latest version, OpenMP 4.0 [153] is going further in the direction of integrating the notion of computation clusters in the programming interface, and guarantees that a smaller number of available clusters than those possibly requested at the application level does not constitute a problem.

9.5.5. Methodology

We set up a simulation platform that processes request traces and generates execution time and wavelength-usage results for all the configurations. The first step, common for the dynamic and static partitioning schemes, is to randomly generate several request traces (each one from a different host computer) that will all simultaneously target the GPPA. This traces store the number of nodes (randomly chosen between 2 and the total number of nodes divided 2) and the execution time required to process each request, as well as the computation time in the host until the next request.

In the dynamic partitioning configurations, each request in the traces will be processed following several steps:

1. If there are enough free nodes to accommodate the new partition, we run the algorithm to choose the nodes that minimize the number of allocated wavelengths.

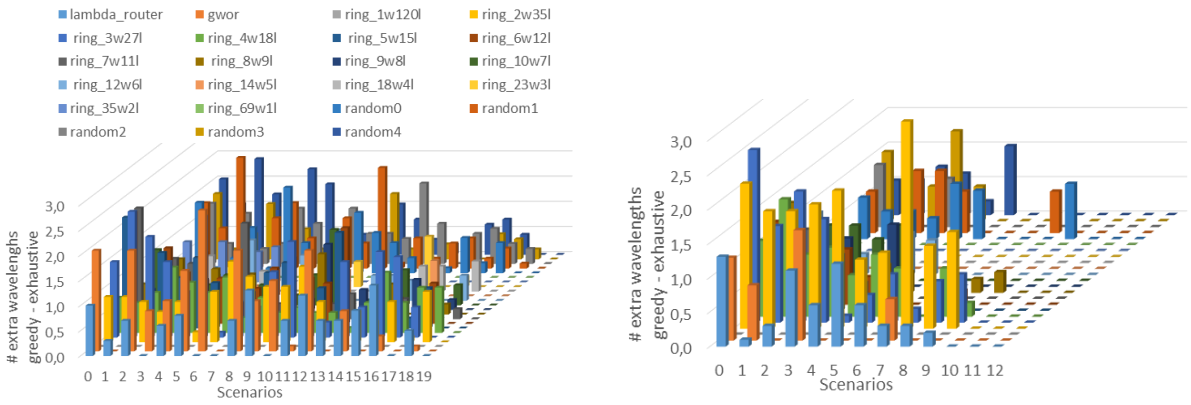
2. If there are not enough free nodes (but there are at least 2), we assign them all to service the request and extend the execution time. We distribute the aggregate execution time for all the nodes and apply a 10% penalty for each missing node.

3. If there are no free nodes (or there is just one free node), we deny the request. The computation will be run at the host computer, again extending the execution time and penalizing for the lack of parallelism. Extending the execution time (both in this and in the previous case) will delay the whole trace from that host computer.

In the static partitioning configurations, the trace processing will be slightly different:

1. If there is a free static partition that fits the request, it is assigned.

2. If there is not, we look for a bigger partition, in which some of the nodes will be left unused.



(a) New partition of 4 nodes

(b) New partition of 8 nodes

Figure 9.9: Number of allocated wavelengths for our greedy algorithm over the exhaustive search algorithm for all the considered topologies in 20 random initial scenarios. The scenarios are ordered from the highest number of free nodes (scenario 0, 16 free nodes) to the lowest (scenario 19, 4 free nodes)

3. If both options failed, we look for a smaller partition and extend the execution time applying the penalization.

4. If there are no free partitions, we deny the request.

As we can see, the static partitioning configurations are less flexible and will result in longer execution times.

9.5.6. Results

This section presents the results for wavelength usage and laser power savings, pointing out the trade-offs between the dynamic and the static partitioning strategies.

9.5.6.1. Characterization of the Algorithm

We first focus on the dynamic partitioning strategies, and determine how good our greedy algorithm is at reusing wavelengths in comparison with the exhaustive search algorithm, which is much more complex. We analyse the number of extra allocated wavelengths to create a single new partition in the λ-router, gwor, several ring designs with varying number of wavelengths and waveguides, and several random communication matrixes that do not correspond to real topologies (but are anyway useful to test the algorithm). To obtain meaningful results, we analyse the allocation of a new partition from 20 different initial scenarios. To create each of the initial scenarios, we set a small random trace and run it on every topology with the greedy algorithm. That way, we get an equivalent starting point for every topology.

Figure 9.9 shows the number of allocated wavelengths for our greedy algorithm over the exhaustive search algorithm, for new partitions of 4 and 8 nodes. We notice that when there are already many allocated nodes, and, therefore, many allocated wavelengths (towards the right-hand side of the graphs), it is easier for the greedy algorithm to find a partition that needs as few extra wavelengths as the exhaustive search. This is especially true when we create a bigger partition, because there are fewer degrees of freedom, so our algorithm is more likely to find an optimum set of nodes. To create the 4-node partitions, the greedy algorithm needs to add an average of 2.7 wavelengths across all scenarios, compared to 2.2 for the exhaustive search. For the 8-node partitions, the average number of added wavelengths is 7.9 and 7.6, for the greedy and exhaustive algorithms, respectively. Our greedy algorithm never needs to add more than 3 extra wavelengths over the exhaustive search algorithm, and rarely more than 2, which is an outstanding result for such a low complexity algorithm.

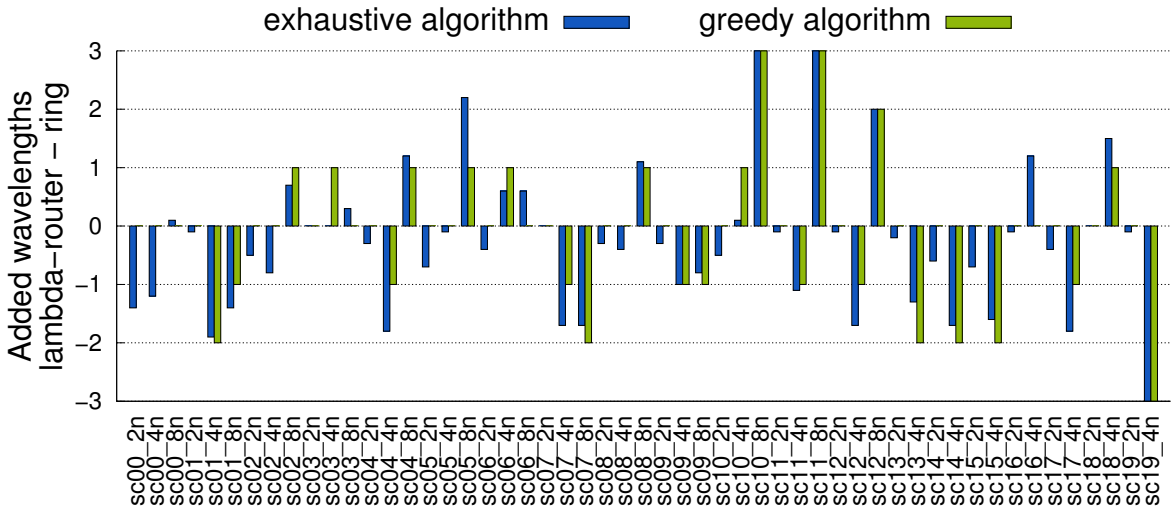


Figure 9.10: Comparison of the λ-router with the ring in 20 random initial scenarios and new partitions of 2, 4, and 6 nodes. The bars represent the number of allocated wavelengths in the λ-router over the ones allocated in the ring to set partitions of different sizes in the 20 scenarios, with the greedy and the exhaustive algorithms. Note that a larger value for the exhaustive algorithm does not mean that it allocates more wavelengths, it simply means there is a larger difference between the topologies. The absolute number of allocated wavelengths is always smaller for the exhaustive algorithm.

9.5.6.2. Partitioning Comparison of Different Topologies

Following the same initial-scenario methodology as in the previous section, we now perform pairwise comparisons to demonstrate that the greedy algorithm does not favour a topology over another, but rather it is the inherent characteristics of each topology that make it behave better or worse in each scenario. We compare two topologies with the greedy and the exhaustive search algorithms, and prove that, at each testing point, the same topology performs better than the other regardless of the algorithm.

Figure 9.10 shows the results of the comparison of the λ-router with the 15-wavelength ring. The positive values correspond to the cases where the λ-router needs more extra wavelengths, and a larger absolute value means a larger difference between the two topologies. We notice that in every case, the greedy and exhaustive bars have the same polarity. We also compared the λ-router with the gwor and the randomly generated truth tables for all-to-all communication with 16 wavelengths, seeing that this was true in 99.7% of the scenarios. This points out that in each scenario one topology is more difficult to handle than the other due to its features, not to the algorithm.

9.5.6.3. Logical-Level Wavelength-on Time

We now run the complete traces as explained in Section 9.5.5 and calculate the aggregated wavelength-on time, that is, the sum of the number of cycles each wavelength is used, considering that when a wavelength is not used in any partition, the corresponding laser source can be switched off. We must remember that two wavelengths are always kept on in order to guarantee communication with the memory controllers.

For the dynamic partitioning configurations we have tried to match the number of wavelengths across all topologies: 16 for the λ-router, and 15 for the gwor and ring. This way, we can fairly compare how each topology reacts to partitioning requests at a logical level. In this case, we introduce also the two statically partitioned configurations, as explained in Section 9.5.4. The inflexibility of the static configurations leads to a longer execution time compared with the dynamic partitioning ones, in particular, 19% extra cycles for the homogeneous partitioning and 14% extra for the mix.

Figure 9.11 depicts the aggregated wavelength-on time for the different topologies and partitioning strategies. For the dynamic partitioning we notice that our algorithm is able to cut the wavelength-on time almost in half from the always-on baseline, and is only slightly worse than the exhaustive search.

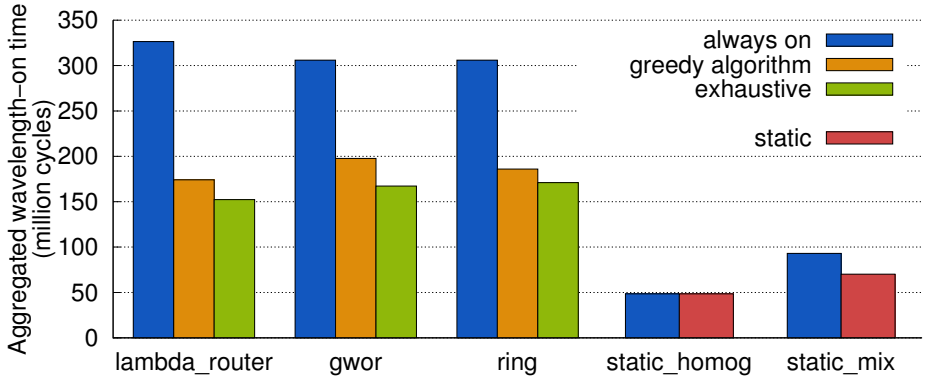


Figure 9.11: Aggregated wavelength-on time for different topologies and partitioning strategies.

Out of the three topologies, the λ-router is the one that achieves the best results, even though it starts with one wavelength more than the others. The static partitioning configurations result on a much better wavelength-reuse, and the extended execution time does not reflect on longer usage time for the laser sources. In this case, switching off the unused lasers does not result in a large improvement, but the implementation of several small static partitions that reuse the same wavelengths is already an optimized starting point. Out of the two static configurations, the one with homogeneous partitions obtains the best results, as it only needs two wavelengths.

9.5.6.4. Energy Analysis

To realistically calculate the laser power for the topologies, we take into account the place&route constraints of the 3D architecture. We assume an 8mmx8mm die size and consider the optical parameters from [130].

We compute the maximum insertion loss of the optical network and calculate the minimum power required to reliably detect the optical message at the destination side. We set the GPPA frequency at 1GHz. The physical design of the optical ring is manually generated, while the λ-router is automatically generated [21]. The gwor is left out of the comparison due to the complexity of its physical design and the clear supremacy of the ring over filtered-based topologies [130]. We assume that the laser stabilization time is included in the partition set-up time, along with the execution time of the greedy algorithm.

Figure 9.12 shows the energy spent by the laser sources to run the traces on the λ-router, the ring, and the static configurations. We clearly see that the λ-router cannot compete with the ring, even though it was the topology that achieved the best wavelength-reuse in the previous section. Again, the exhaustive search gives only slightly better results than the greedy algorithm. The static partitioning consumes significantly less laser power, and the best choice among all the configurations is the statically partitioned ONoC with homogeneous partitions. In that case, the 2 implemented wavelengths must be always on in order to support the communication with the memory controllers. We must remember, however, that this benefit comes at the cost of a rigid chip architecture that involves worse performance.

In our experiments, we consider that the time to set up the partitions is negligible. This allows us to compare the greedy and exhaustive search algorithms under the same trace, which would otherwise be impossible. The outstanding energy savings we achieve give us a large margin for the execution time of the algorithm before its use stops being cost-effective. For example, with the ring we can afford an overhead of 45% over the execution time of each request before we lose the energy savings. This can certainly accommodate the greedy algorithm, but not the inefficient exhaustive search.

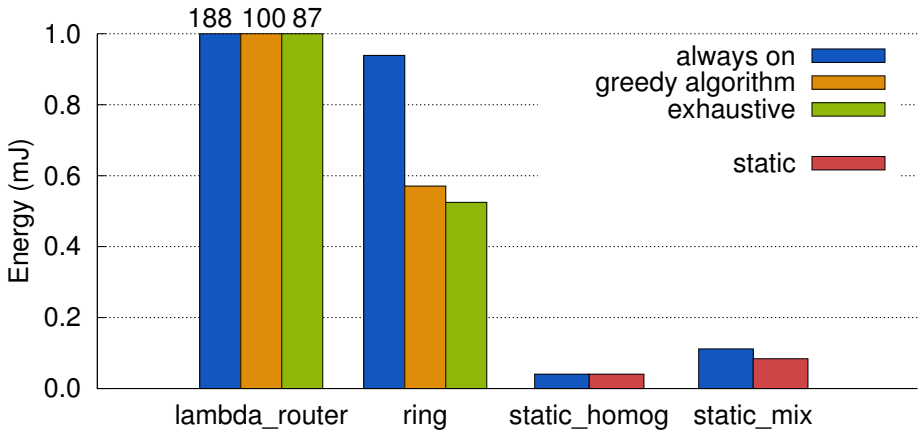


Figure 9.12: Laser source energy for different topologies and partitioning strategies.

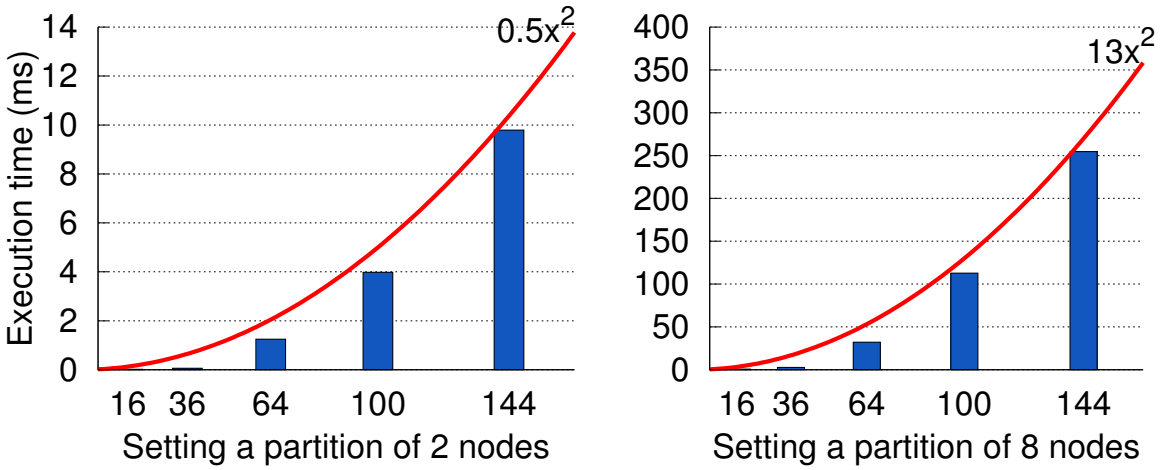


Figure 9.13: Execution time of the greedy algorithm to allocate a new partition of 2 and 8 nodes with increasing number of nodes in a ring topology. As a reference, we also plot a quadratic curve in each graph.

9.5.7. Scalability of the algorithm

In this section we demonstrate that the execution time of the greedy algorithm scales quadratically with the number of nodes in the system, confirming the complexity of $O(n^2)$. Figure 9.13 shows the execution time to build a partition of 2 and 8 nodes with an increasing number of nodes in a ring topology, on top of an ARMv7 processor simulated on gem5. The observed trend corroborates the polynomial complexity and confirms the suitability of the algorithm for its integration on larger systems. The exhaustive search algorithm was also tested under the same scenarios, resulting in exorbitant execution times.

9.6. Concluding Remarks

The work proposes the first assessment of optical interconnect technology in the context of GPPA devices for the high-end embedded computing domain following two different approaches. First, the system is re-architected around an optical interconnect fabric, under a realistic hybrid integration strategy. When put at work with realistic workloads, the photonically-integrated GPPA turns out to be extremely effective in speeding up application execution by at least 15%. This translates into a static power overhead of 2.5x for the ONoC, which is however expected to go down to 1.3x with future silicon photonic technology. In contrast, the ONoC is more energy efficient than the ENoC (up to an order of magnitude) regardless of the specific optical technology, thus confirming that in terms of energy-per-bit, the ONoC is definitely

hard to beat. Overall, the above quality metrics paint a promising picture for augmenting GPPAs with optical devices, while clearly pointing to the most important candidate for optimization: static energy reduction through technology evolution as well as gating techniques.

Second, we present the first work that integrates an optical network into a virtualized environment. Partitioning actually yields laser power savings, thus addressing the concerns detected in our first approach, and we present two versions with different trade-off points to extract these benefits. On one hand, wavelengths are reused across partitions by running an online greedy algorithm for wavelength allocation and partition configuration. On the other hand, we present statically partitioned ONoCs and demonstrate their superior physical properties, which ultimately lead to hard-to-beat total energy figures. This approach is compatible with the flexibility of modern programming models, which can adapt to the parallelism the hardware platform exposes even if it is not the optimal one for the application at hand. However, if we are not ready to accept the drawback of building such a rigid chip architecture, we can opt for the first option and still achieve significant power savings.

Part IV

Conclusions

This final part concludes the dissertation. In includes the conclusions obtained from the contributions of this thesis, ideas for future work and the list of the thesis publications.

Chapter 10
Conclusions and future work

Summary

This last chapter sums up the conclusions of this work, introduces some ideas for future research, and lists the publications where the contributions of this thesis have been published.

10.1. Conclusions

Chip multiprocessors (CMPs) are composed of multiple nodes connected via an interconnection network, which contributes with a substantial share to chip area, energy consumption, and system performance. This network-on-chip (NoC) has traditionally been implemented with metal wires and has been the focus of many research publications. In this work we start by analysing these electronic NoCs from a comprehensive perspective: we consider the interconnection network and the cache hierarchy simultaneously, which helps identify improvement opportunities in the design of CMPs. We model in detail the processors, memory hierarchy, and electronic network using full-system simulation and executing both parallel and multiprogrammed realistic workloads. We perform a qualitative and quantitative analysis of three well-known network topologies: mesh, torus, and ring, and their concentrated versions for CMPs with 16 single and multi-threaded cores and 64 single-threaded cores.

With this detailed analysis, we demonstrate that performance is highly affected by the choice of the interconnect, especially in 64-core systems, where the ring performance drops by 72% with respect to the concentrated mesh for parallel workloads. The ring topologies perform worse due to the increased hop count, which translates into higher network latency and turns out to be the most relevant aspect of the NoC. As a result, the concentrated mesh topology offers the best performance with low energy consumption and area for all workloads, even with multithreaded cores, which generate a heavier traffic load. We also determine that the placement and the number of memory controllers has a negligible effect on system performance with the realistic applications we have tested, because they have limited memory access.

From this detailed network evaluation where we identified the most relevant metrics to be optimized, we come up with a smart network design that greatly cuts down the energy expended at the interconnect, reduces the area, and improves performance. Our work was inspired by the observation that most of the traffic follows a request-reply pattern, which helps anticipate the path most replies will follow. We propose a mechanism called Reactive Circuits based on reserving network resources and dynamically building the circuit for the reply while the request travels through the network. Guaranteeing complete circuits for data messages enables us to predict when they will reach their destination, and elegantly eliminate the need for their acknowledgement, as well as removing unnecessary buffers, thus reducing network power by 20%.

In the second section of the thesis, we consider the emerging silicon photonics technology to build optical networks-on-chip (ONoCs) with increased bandwidth and reduced latency and energy-per-bit. We focus on wavelength-routed optical NoCs, which can implement simultaneous all-to-all communications, and dedicate our efforts both to designing the optical network and to comparing it with its electronic counterpart when integrated into industry-relevant devices.

First, we present a tool to generate ring communication matrices while optimizing for power efficiency. We automatically calculate power with physical constraint awareness including the contribution of the laser distribution network. Our algorithm is able to generate ring designs with fewer waveguides and/or wavelengths than any other existing proposal for any number of nodes. We demonstrate that an even number of waveguides allows for more balanced designs with reduced power. We also find out that the best design point is the ring with only two waveguides, pointing out that adding extra wavelengths is more cost-effective than adding extra waveguides. With a given number of waveguides, reducing the number of wavelengths does not necessarily mean saving power, because it enforces the use of non-minimal paths, which increases insertion loss. There is a large margin between results with the ideal and realistic laser distribution networks, indicating that power not only depends on an efficient communication matrix, but also on a good laser distribution network design, which should be the focus of further optimizations.

Then, we design the first complete network interface architecture for optical NoCs, and use the complete ONoC design to communicate the nodes of a CMP. This work sets the scene for further comparative analysis of optical versus electronic NoCs. Regarding latency, the ONoC is always faster than its electronic counterpart even considering the NI, thus preserving the primary goal of a wavelength-routed ONoC. Considering power, switches are the main contributors in ENoCs, while the NI has the largest share in ONoCs. Finally, the ONoC preserves its superior dynamic energy properties over its ENoC counterpart,

even in the presence of its NI. This work shows that the NI architecture should not be overlooked for realistic ONoC assessments, and comes up with new insights not provided by earlier photonic network evaluations. The most important one is that NI optimizations perhaps have higher priority over the relentless search for ultra-low-loss optical devices.

With the objective of testing the optical network with other well-known devices, we also undertake the first assessment of optical interconnect technology in the context of general-purpose programmable accelerators (GPPA) devices for the high-end embedded computing domain, following two different approaches. First, the system is re-architected around an optical interconnect fabric under a realistic hybrid integration strategy: we integrate a global optical network and keep the local electronic network. When put at work with realistic workloads, the photonically-integrated GPPA turns out to be extremely effective in speeding up application execution. Overall, our results paint a promising picture for augmenting GPPAs with optical devices, while clearly pointing to the most important candidate for optimization: static energy reduction through technology evolution as well as gating techniques. Second, we present the first work that integrates an optical network into a virtualized environment. This usage model is gaining momentum in manycore processors as a way to enable the concurrent execution of many programs in the same platform. This environment is of significant benefit for optical networks and partitioning actually yields laser power savings, thus addressing the concerns detected in our first approach. We present two versions with different trade-off points to extract these benefits: on one hand, wavelengths are reused across partitions by running an online greedy algorithm for wavelength allocation and partition configuration; on the other hand, we present statically partitioned ONoCs and demonstrate their superior physical properties, which ultimately lead to outstanding total energy figures.

In conclusion, optical networks-on-chip offer very promising results, but still need a considerable research effort. We have to come up with optimized designs custom-tailored for real platforms, and define all the details beyond simple optical waveguides required to materialize the use of this new technology in silicon chips.

10.2. Future Work

We propose several topics to continue the research effort of this work:

- The network interface architecture designed for optical networks targets wavelength-routed ONoCs. As an interesting extension, detailed network interfaces should be designed for arbitrated optical networks (single-reader multiple-writer, multiple-reader single-writer, or multiple-reader multiple-writer), and for space-routed networks.
- When integrating the optical network into a GPPA, we have followed two different approaches: replacing the global network while keeping an electronic local network, and replacing the local network with an ONoC capable of setting partitions. The next logical step is engineering an all-optical communication system for the GPPA.
- There are several technologies that could also be used to successfully implement efficient networks on chip such as carbon nanotubes, graphene nanoribbons, and wireless. A detailed and unbiased comparison of all the technologies would be very beneficial and point future research in the correct direction.

10.3. Publications

This Section presents the comprehensive list of all the publications of the thesis, which were also referenced in the appropriate sections.

About the analysis of electronic network-on-chip topologies:

- Marta Ortín, Alexandra Ferrerón, Jorge Albericio, Darío Suárez, María Villarroya, Cruz Izu, and Víctor Viñals. Characterization and cost-efficient selection of NoC topologies for general purpose

CMPs. In proceedings of the 2013 workshop on Interconnection Network Architecture: On-Chip, Multi-Chip (INA-OCMC), January 2013. In collaboration with the University of Adelaide. [110]

- Marta Ortín, Darío Suárez, María Villarroya, Cruz Izu, and Víctor Viñals. Analysis of Network-on-Chip Topologies for Cost-Efficient Chip Multiprocessors. Journal of Microprocessors and Microsystems, February 2016. In collaboration with the University of Adelaide. [117]

About the construction of reactive circuits in electronic NoCs:

- Marta Ortín, Darío Suárez, María Villarroya, Cruz Izu and Víctor Viñals. Reserva de circuitos para tráfico reactivo en CMPs homogéneos. In proceedings of the XXIV Jornadas de Paralelismo, September 2013. In collaboration with the University of Adelaide. [113]
- Marta Ortín, Darío Suárez, María Villarroya, Cruz Izu and Víctor Viñals. Dynamic construction of circuits for reactive traffic in homogeneous CMPs. In proceedings of the Design, Automation and Test in Europe Conference, March 2014. In collaboration with the University of Adelaide. [109]

About the network interface design for optical networks and its application to chip multiprocessors:

- Marta Ortín, Luca Ramini, Víctor Viñals, and Davide Bertozzi. Capturing the Sensitivity of Optical Network Quality Metrics to its Network Interface Parameters. Invited paper to the I Workshop on Exploiting Silicon Photonics for Energy-Efficient Heterogeneous Parallel Architectures (SiPhotonics), January 2014. In collaboration with the University of Ferrara. [115]
- Marta Ortín, Luca Ramini, Hervé Tatenguem Fankem, Víctor Viñals, and Davide Bertozzi. A Complete Electronic Network Interface Architecture for Global Contention-free Communication over Emerging Optical Networks-on-chip. In proceedings of the 24th Edition of the Great Lakes Symposium on VLSI, May 2014. In collaboration with the University of Ferrara. [111]
- Marta Ortín, Luca Ramini, Víctor Viñals, and Davide Bertozzi. Capturing the sensitivity of optical network quality metrics to its network interface parameters. Journal of Concurrency and Computation: Practice and Experience, 2014. In collaboration with the University of Ferrara. [116]
- Marta Ortín, Marco Balboni, Luca Ramini, Víctor Viñals, and Davide Bertozzi. Optical Networks-on-Chip: Time for Accurate Crossbenchmarking. Invited talk at the CMOS Emerging Technologies Research conference, July 2014. In collaboration with the University of Ferrara. This talk also included the use of optical networks in general purpose programmable accelerators. [114]
- Marta Ortín, Luca Ramini, Hervé Tatenguem Fankem, Víctor Viñals, and Davide Bertozzi. Arquitectura completa de una interfaz de red electrónica para redes ópticas en chip. In proceedings of the XXV Jornadas de Paralelismo, September 2014. In collaboration with the University of Ferrara. [112]
- Luca Ramini, Hervé Tatenguem Fankem, Alberto Ghiribaldi, Paolo Grani, Marta Ortín, Anja Boos, and Sandro Bartolini. Towards compelling cases for the viability of silicon-nanophotonic technology in future manycore systems. In proceedings of the Eighth International Symposium on Networks-on-Chip (NoCS), September 2014. In collaboration with the University of Ferrara, the University of Siena, and the University of Munich. [128]

About optical networks applied to general purpose programmable accelerators:

- Marco Balboni, Marta Ortín, Alessandro Capotondi, Hervé Fankem Tatenguem, Alberto Ghiribaldi, Luca Ramini, Víctor Viñals, Andrea Marongiu, and Davide Bertozzi. Augmenting Manycore Programmable Accelerators with Photonic Interconnect Technology for the High-End Embedded Computing Domain. In proceedings of the Eighth International Symposium on Networks-on-Chip (NoCS), September 2014. In collaboration with the University of Ferrara, the University of Bologna, and ETH Zurich. [10]
- Marta Ortín, Luca Ramini, Marco Balboni, Lorenzo Zuolo, Nonato Maddalena, Víctor Viñals, and Davide Bertozzi. Partitioning Strategies of Wavelength-Routed Optical Networks-on-Chip for Laser Power Minimization. In proceedings of the II Workshop on Exploiting Silicon Photonics for Energy-Efficient Heterogeneous Parallel Architectures (SiPhotonics), January 2015. In collaboration with the University of Ferrara. [108]

Conclusiones

Los multiprocesadores en chip (CMPs, del inglés *chip multiprocessors*) están compuestos por varios nodos conectados a través de una red de interconexión, que contribuye en gran medida al área, consumo energético y rendimiento del chip. Esta red en chip se ha implementado tradicionalmente con conexiones metálicas y ha sido objeto de muchas publicaciones científicas. En este trabajo, comenzamos analizando estas redes electrónicas desde una perspectiva muy amplia: consideramos la red de interconexión y la jerarquía de memoria simultáneamente, lo que ayuda a identificar nuevas oportunidades de mejora en el diseño de CMPs. Modelamos detalladamente los procesadores, jerarquía de memoria y la red de interconexión usando simulación de sistema completo y ejecutando cargas de trabajo realistas, tanto paralelas como multiprogramadas. Llevamos a cabo un análisis cualitativo y cuantitativo de tres topologías muy conocidas: malla, toro y anillo, y sus versiones concentradas, para CMPs con 16 cores (de 1 y 4 hilos) y 64 cores (de 1 hilo).

Con este detallado análisis, demostramos que la red tiene un alto impacto en el rendimiento, especialmente en sistemas de 64 cores, en los que el anillo aumenta el tiempo de ejecución en un 72 % respecto a la malla concentrada para cargas paralelas. Los anillos tienen peor rendimiento debido al alto número de saltos necesarios para ir de origen a destino, lo que se traduce en una latencia de red más alta y resulta ser el aspecto más relevante de la red. Como resultado, la malla concentrada ofrece el mejor rendimiento con bajo consumo energético y área para todas las cargas de trabajo, incluso con cores de varios hilos, que generan mayor cantidad de tráfico. También determinamos que la posición y el número de controladores de memoria tienen un efecto despreciable con las aplicaciones realistas que hemos simulado, debido a que su acceso a memoria es muy limitado.

A partir de esta evaluación detallada de la red en la que identificamos los aspectos más relevantes que deben ser optimizados, proponemos un diseño de red que disminuye en gran medida la energía consumida por la red, reduce el área y mejora el rendimiento. Observamos que la mayor parte del tráfico sigue un patrón de petición-respuesta, lo que ayuda a anticipar el camino que la mayor parte de las respuestas seguirán. Basándonos en eso, proponemos un mecanismo llamado Circuitos Reactivos que reserva los recursos de la red y construye dinámicamente un circuito para la respuesta mientras la petición viaja por la red. Garantizar un circuito para los mensajes nos permite predecir cuándo llegarán a su destino, y por lo tanto suprimir los *acknowledgements*, además de eliminar *buffers* innecesarios reduciendo la potencia de la red en un 20 %.

En la segunda sección de la tesis, consideramos una tecnología emergente, la fotónica en silicio, para construir redes ópticas en chip con mayor ancho de banda y latencia y energía-por-bit reducidos. Nos centramos en las redes ópticas enrutadas mediante la selección de longitud de onda (*wavelength-routed)*, que implementan comunicaciones simultáneas entre todos los nodos. Nos dedicamos tanto a diseñar la red óptica como a compararla con su equivalente electrónico en sistemas relevantes para la industria.

Primero, presentamos una herramienta para generar las matrices de comunicación de un anillo óptico, al mismo tiempo que optimiza su energía. Calculamos la potencia de la red automáticamente teniendo en cuenta las restricciones físicas y la contribución de la red de distribución del láser. Nuestro algoritmo es capaz de generar diseños para el anillo con menos guías y/o longitudes de onda que ninguna otra propuesta existente para cualquier número de nodos. Demostramos que un número par de guias de onda lleva a diseños más balanceados con consumo energético reducido. También descubrimos que los mejores diseños son los anillos con únicamente dos guías de onda, destacando que aumentar el número de longitudes de onda es más efectivo que añadir cables ópticos. Dado un número fijo de cables ópticos, reducir el número de longitudes de onda no implica necesariamente un ahorro energético, porque fuerza a utilizar caminos no mínimos, lo que incrementa las pérdidas por inserción. Hay un amplio margen entre los resultados con las redes de distribución del láser ideal y realista, lo que indica que la potencia consumida depende no sólo de una matriz de comunicación eficiente, sino también de una buena red de distribución del láser, que debería ser objeto de futuras optimizaciones.

Después, diseñamos la primera interfaz de red completa para redes en chip ópticas, y usamos la red

óptica completa para comunicar los nodos en un CMP. Este trabajo sienta las bases para futuros análisis comparativos de redes ópticas y electrónicas. Respecto a la latencia, la red óptica es siempre más rápida que su equivalente electrónico, incluso considerando la interfaz, preservando así el principal objetivo de las redes ópticas *wavelength-routed*. En cuanto a potencia, detectamos que los routers suponen la principal contribución en las redes electrónicas, mientras que la interfaz constituye el mayor porcentaje en las ópticas. Finalmente, la red óptica sigue manteniendo un menor consumo en energía dinámica, incluso incluyendo la interfaz. Este trabajo demuestra que la interfaz de red no debería ser obviada si se desea llevar a cabo un análisis realista de la red óptica, y proporciona nuevas conclusiones para la investigación en redes de este tipo. La más importante es que la optimización de la interfaz de red debería ser prioritaria frente a la búsqueda de componentes ópticos de muy bajo consumo.

Con el objetivo de probar la red óptica en otras plataformas conocidas, también realizamos el primer análisis de una red de interconexión óptica integrada en un acelerador programable de propósito general para procesadores empotrados de alto nivel, siguiendo dos enfoques diferentes. Primero, seguimos una estrategia de integración híbrida muy realista: integramos una red global óptica y mantenemos la red local electrónica. Usando aplicaciones realistas, demostramos que el acelerador con red híbrida resulta muy efectivo para mejorar el rendimiento. En general, nuestros resultados son optimistas, aunque claramente señalan la energía estática como principal candidato para futuras optimizaciones. A continuación, presentamos el primer trabajo que integra una red óptica en un entorno virtualizado. Este modelo de uso está adquiriendo popularidad en procesadores de varios cores para permitir la ejecución concurrente de varias aplicaciones en la misma plataforma. Este entorno es especialmente beneficioso para las redes electrónicas, ya que su particionado puede resultar en una reducción energética, atendiendo así los problemas detectados en el primer enfoque. Presentamos dos versiones con diferentes *trade-offs* para extraer estos beneficios: por un lado, ejecutamos un algoritmo para configurar particiones que procura reutilizar las longitudes de onda; por otro lado, usamos redes ópticas con particiones estáticas y demostramos su superioridad en cuando a consumo energético.

Como conclusión, las redes ópticas en chip ofrecen resultados muy prometedores, pero todavía necesitan un esfuerzo de investigación considerable. Debemos proponer diseños optimizados y personalizados para plataformas reales, así como definir todos los detalles necesarios para materializar esta nueva tecnología en chips de silicio.

Bibliography

[1] ELPIDA 512M bits Mobile RAM, EDS51321DBH-TS.

[2] A. Abousamra, A.K. Jones, and R. Melhem. Codesign of NoC and cache organization for reducing access latency in chip multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 23(6):1038–1046, 2012.

[3] Ahmed Abousamra, Alex K. Jones, and Rami Melhem. Proactive circuit allocation in multiplane NoCs. In *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, pages 35:1–35:10, New York, NY, USA, 2013. ACM.

[4] A.K. Abousamra, R.G. Melhem, and A.K. Jones. Deja-vu switching for multiplane NoCs. In *Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS)*, pages 11 –18, may 2012.

[5] Dennis Abts, Natalie D. Enright Jerger, John Kim, Dan Gibson, and Mikko H. Lipasti. Achieving predictable performance through better memory controller placement in many-core CMPs. In *Proceedings of the 36th annual international symposium on Computer architecture*, ISCA '09, pages 451–461, New York, NY, USA, 2009. ACM.

[6] N. Agarwal, T. Krishna, Li-Shiuan Peh, and N.K. Jha. GARNET: A detailed on-chip network model inside a full-system simulator. In *IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS*, pages 33 –42, april 2009.

[7] Niket Agarwal, Li-Shiuan Peh, and Niraj K. Jha. In-network coherence filtering: snoopy coherence without broadcasts. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 232–243, New York, NY, USA, 2009. ACM.

[8] Y. Arakawa, T. Nakamura, Y. Urino, and T. Fujita. Silicon photonics for next generation system integration platform. *Communications Magazine, IEEE*, 51(3):72–77, March 2013.

[9] D. Avresky and N. Natchev. Dynamic reconfiguration in computer clusters with irregular topologies in the presence of multiple node and link failures. *Computers, IEEE Transactions on*, 54(5):603–615, May 2005.

[10] M. Balboni, M. Ortin, A. Capotondi, H. Fankem Tatenguem, A. Ghiribaldi, L. Ramini, V. Viñals, A. Marongiu, and Bertozzi. Augmenting manycore programmable accelerators with photonic interconnect technology for the high-end embedded computing domain. In *International Symposium on Networks-on-Chip (NOCS)*, 2014.

[11] Marco Balboni, Francisco Triviño, José Flich, and Davide. Bertozzi. Optimizing the overhead for network-on-chip routing reconfiguration in massively parallel multi-core platforms. In *International System-on-Chip Symposium*, 2013.

[12] James Balfour and William J. Dally. Design tradeoffs for tiled CMP on-chip networks. In *Proceedings of the 20th annual international conference on Supercomputing*, ICS '06, pages 187–198, New York, NY, USA, 2006. ACM.

[13] Luiz André Barroso, Kourosh Gharachorloo, Robert McNamara, Andreas Nowatzyk, Shaz Qadeer, Barton Sano, Scott Smith, Robert Stets, and Ben Verghese. Piranha: a scalable architecture based on single-chip multiprocessing. In *Proceedings of the 27th annual international symposium on Computer architecture*, ISCA '00, pages 282–293, New York, NY, USA, 2000. ACM.

[14] C. Batten, A. Joshi, J. Orcutt, A. Khilo, B. Moss, C. Holzwarth, M. Popovic, Hanqing Li, Henry I. Smith, J. Hoyt, F. Kartner, R. Ram, V. Stojanovic, and K. Asanovic. Building manycore processor-to-DRAM networks with monolithic silicon photonics. In *16th IEEE Symposium on High Performance Interconnects, HOTI*, pages 21–30, Aug 2008.

[15] C. Batten, A. Joshi, V. Stojanovic, and K. Asanovic. Designing chip-level nanophotonic interconnection networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, pages 137–153, 2012.

[16] Scott Beamer, Chen Sun, Yong-Jin Kwon, Ajay Joshi, Christopher Batten, Vladimir Stojanovic, and Krste Asanovic. Re-architecting DRAM memory systems with monolithically integrated silicon photonics. In *international symposium on Computer architecture, ISCA*, pages 129–140. ACM, 2010.

[17] Keren Bergman, Luca P. Carloni, Aleksandr Biberman, Johnnie Chan, and Gilbert Hendry. *Photonic Network-on-Chip Design*. Springer, 2014.

[18] George B. P. Bezerra, Stephanie Forrest, and Payman Zarkesh-Ha. Reducing energy and increasing performance with traffic optimization in many-core systems. In *Proceedings of the System Level Interconnect Prediction Workshop*, SLIP '11, pages 3:1–3:7, Piscataway, NJ, USA, 2011. IEEE Press.

[19] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The PARSEC benchmark suite: characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, PACT '08, pages 72–81, New York, NY, USA, 2008. ACM.

[20] M. Biere, L. Gheorghe, G. Nicolescu, I. O'Connor, and G. Wainer. Towards the high-level design of optical networks-on-chip. Formalization of opto-electrical interfaces. In *International Conference on Electronics, Circuits and Systems.*, pages 427–430, 2007.

[21] Anja Boos, Luca Ramini, Ulf Schlichtmann, and Davide Bertozzi. Proton: An automatic place-and-route tool for optical networks-on-chip. In *Proceedings of the International Conference on Computer-Aided Design*, ICCAD '13, pages 138–145, Piscataway, NJ, USA, 2013. IEEE Press.

[22] D. Bortolotti, C. Pinto, A. Marongiu, M. Ruggiero, and L. Benini. VirtualSoc: A full-system simulation environment for massively parallel heterogeneous system-on-chip. In *International Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW)*, pages 2182–2187, May 2013.

[23] M. Briere, B. Girodias, Y. Bouchebaba, G. Nicolescu, F. Mieyeville, F. Gaffiot, and I. O'Connor. System level assessment of an optical NoC in an MPSoC platform. In *Design, Automation Test in Europe Conference Exhibition*, pages 1–6, 2007.

[24] Everton Carara, Fernando Moraes, and Ney Calazans. Router architecture for high-performance NoCs. In *Proceedings of the 20th annual conference on Integrated circuits and systems design*, SBCCI '07, pages 111–116, New York, NY, USA, 2007. ACM.

[25] J. Chan, G. Hendry, A. Biberman, and K. Bergman. Architectural design exploration of chip-scale photonic interconnection networks using physical-layer analysis. In *Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference, 2010 Conference on (OFC/NFOEC)*, pages 1–3, 2010.

[26] J. Chan, G. Hendry, A. Biberman, and K. Bergman. Architectural exploration of chip-scale photonic interconnection network designs using physical-layer analysis. *Journal of Lightwave Technology*, pages 1305–1315, 2010.

[27] Johnnie Chan, Gilbert Hendry, Aleksandr Biberman, Keren Bergman, and Luca P. Carloni. PhoenixSim: A simulator for physical-layer analysis of chip-scale photonic interconnection networks. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '10, pages 691–696, 3001 Leuven, Belgium, Belgium, 2010. European Design and Automation Association.

[28] C. Chen, T. Zhang, P. Contu, J. Klamkin, A. Coscun, and A. Joshi. Sharing and placement of on-chip laser sources in silicon photonic NoCs. In *International Symposium on Networks-on-Chip (NOCS)*, 2014.

[29] Chao Chen and A Joshi. Runtime management of laser power in silicon-photonic multibus NoC architecture. *IEEE Journal of Selected Topics in Quantum Electronics*, 19(2):3700713–3700713, March 2013.

[30] Mark J. Cianchetti, Joseph C. Kerekes, and David H. Albonesi. Phastlane: A rapid transit optical routing network. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 441–450, New York, NY, USA, 2009. ACM.

[31] P. Conway, N. Kalyanasundharam, G. Donley, K. Lepak, and B. Hughes. Cache hierarchy and memory subsystem of the AMD opteron processor. *Micro, IEEE*, 30(2):16–29, 2010.

[32] William Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[33] William J. Dally and Charles L. Seitz. The torus routing chip. *Distributed Computing*, 1:187–196, 1986. 10.1007/BF01660031.

[34] W.J. Dally. Virtual-channel flow control. In *17th Annual International Symposium on Computer Architecture*, pages 60–68, May 1990.

[35] Yigit Demir and Nikos Hardavellas. Ecolaser: An adaptive laser control for energy-efficient on-chip photonic interconnects. In *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*, ISLPED '14, pages 3–8, New York, NY, USA, 2014. ACM.

[36] S. Dighe, S. Gupta, V. De, S. Vangal, N. Borkar, S. Borkar, and K. Roy. A 45nm 48-core IA processor with variation-aware scheduling and optimal core mapping. In *Symposium on VLSI Circuits (VLSIC)*, pages 250–251, June 2011.

[37] J. Duato, P. Lopez, F. Silla, and S. Yalamanchili. A high performance router architecture for interconnection networks. In *Proceedings of the International Conference on Parallel Processing*, volume 1, pages 61–68 vol.1, 1996.

[38] K. Fan, M. Kudlur, G. Dasika, and S. Mahlke. Bridging the computation gap between programmable processors and hardwired accelerators. In *IEEE 15th International Symposium on High Performance Computer Architecture*, pages 313–322, Feb 2009.

[39] Chaochao Feng, Zhonghai Lu, Axel Jantsch, Jinwen Li, and Minxuan Zhang. A reconfigurable fault-tolerant deflection routing algorithm based on reinforcement learning for network-on-chip. In *Proceedings of the Third International Workshop on Network on Chip Architectures*, NoCArc '10, pages 11–16, New York, NY, USA, 2010. ACM.

[40] R. Fernandez-Pascual, J.M. Garcia, M.E. Acacio, and J. Duato. A fault-tolerant directory-based cache coherence protocol for CMP architectures. In *IEEE International Conference on Dependable Systems and Networks With FTCS and DCC*, pages 267–276, 2008.

[41] P. Francesco, P. Antonio, and P. Marchal. Flexible hardware/software support for message passing on a distributed shared memory architecture. In *Design, Automation and Test in Europe, 2005. Proceedings*, pages 736–741 Vol. 2, March 2005.

[42] R. Gabor, Shlomo Weiss, and A. Mendelson. Fairness and throughput in switch on event multithreading. In *39th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-39*, pages 149–160, 2006.

[43] C. Galland et al. A CMOS-compatible silicon photonic platform for high-speed integrated optoelectronics. *Proc. Integrated Photonics: Materials, Devices, and Applications*, 2013.

[44] P.T. Gaughan and S. Yalamanchili. A family of fault-tolerant routing protocols for direct multiprocessor networks. *IEEE Transactions on Parallel and Distributed Systems*, 6:482–497, 1995.

[45] F. Gilabert, M.E. Gomez, S. Medardoni, and D. Bertozzi. Improved utilization of NoC channel bandwidth by switch replication for cost-effective multi-processor systems-on-chip. In *International Symposium on Networks-on-Chip (NOCS)*, pages 165–172, 2010.

[46] F. Gilabert, S. Medardoni, D. Bertozzi, L. Benini, M.E. Gomez, P. Lopez, and J. Duato. Exploring high-dimensional topologies for NoC design through an integrated analysis and synthesis framework. In *Second ACM/IEEE International Symposium on Networks-on-Chip, NoCs*, pages 107–116, April 2008.

[47] Darryl Gove. CPU2006 working set size. *SIGARCH Comput. Archit. News*, 35(1):90–96, March 2007.

[48] Paolo Grani and Sandro Bartolini. Design options for optical ring interconnect in future client devices. *Journal on Emerging Technologies in Computing Systems*, 10(4):30:1–30:25, June 2014.

[49] Tom R. Halfhill. Power8 hits the merchant market. Memory bandwidth helps IBM server processor ace big benchmarks. *Microprocessor report*, March 2014.

[50] Andreas Hansson, Kees Goossens, and Andrei Rădulescu. Avoiding message-dependent deadlock in network-based systems on chip. *VLSI Design*, 2007.

[51] G. Hendry, J. Chan, S. Kamil, L. Oliker, J. Shalf, L.P. Carloni, and K. Bergman. Silicon nanophotonic network-on-chip using TDM arbitration. In *Annual Symposium on High Performance Interconnects (HOTI)*, pages 88–95, 2010.

[52] G. Hendry, E. Robinson, V. Gleyzer, J. Chan, L.P. Carloni, N. Bliss, and K. Bergman. Circuit-switched memory access in photonic interconnection networks for high-performance embedded computing. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–12, 2010.

[53] J. Howard, S. Dighe, S.R. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V.K. De, and R. Van Der Wijngaart. A 48-core IA-32 processor in 45 nm CMOS using on-die message-passing and DVFS for performance and power scaling. *IEEE Journal of Solid-State Circuits*, 46(1):173 –183, jan. 2011.

[54] Adapteva Inc. Adapteva reference manual. `http://www.parallella.org/wp-content/uploads/2013/01/parallella_gen1_reference.pdf` (Last access November 2015).

[55] NVIDIA Inc. Cuda compute architecture: Fermi. (Last access November 2015).

[56] Texas Instruments Inc. Multicore DSP+ARM keystone II system-on-chip (SoC). `http://www.ti.com/lit/ds/symlink/66ak2h12.pdf` (Last access November 2015).

[57] Xilinx Inc. Zynq-7000 all programmable SoC overview. `http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf` (Last access November 2015).

[58] Intel. Intel Xeon Phi. 2014. `http://www.intel.es/content/dam/www/public/us/en/documents/datasheets/xeon-phi-coprocessor-datasheet.pdf` (Last access November 2015).

[59] Intel. Intel Xeon Phi, Knights Landing. 2014. `https://software.intel.com/sites/default/files/managed/e9/b5/Knights-Corner-is-your-path-to-Knights-Landing.pdf` (Last access November 2015).

[60] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* Wiley, 1991.

[61] Natalie D. Enright Jerger, Li-Shiuan Peh, and Mikko H. Lipasti. Circuit-switched coherence. In *Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '08, pages 193–202, Washington, DC, USA, 2008. IEEE Computer Society.

[62] A. Joshi, C. Batten, Yong-Jin Kwon, S. Beamer, I. Shamim, K. Asanovic, and V. Stojanovic. Silicon-photonic clos networks for global on-chip communication. In *International Symposium on Networks-on-Chip*, pages 124–133, 2009.

[63] Sheng Li; Ke Chen; Jay B. Brockman; Norman P. Jouppi. Performance impacts of non-blocking caches in out-of-order processors. Technical report, HP Laboratories, 2011.

[64] Kalray. Mppa 256 - programmable manycore processor. `www.kalray.eu/products/mppa-manycore/mppa-256/` (Last access November 2015).

[65] David Kanter. 14nm Xeon D secures the data center. *Microprocessor report*, March 2015.

[66] Rishi Kapoor, George Porter, Malveeka Tewari, Geoffrey M. Voelker, and Amin Vahdat. Chronos: Predictable low latency for data center applications. In *Procs of the Third ACM Symp. on Cloud Comp.*, pages 9:1–9:14, New York, NY, USA, 2012. ACM.

[67] Pawan Kapur and Krishna C. Saraswat. Optical interconnects for future high performance integrated circuits. *Physica E: Low-dimensional Systems and Nanostructures*, pages 620 – 627, 2003.

[68] Khronos OpenCL. `http://www.khronos.org/opencl/`. (Last access November 2015).

[69] J. Kim. Low-cost router microarchitecture for on-chip networks. In *42nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-42*, pages 255–266, 2009.

[70] Nevin Kirman, Meyrem Kirman, Rajeev K. Dokania, Jose F. Martinez, Alyssa B. Apsel, Matthew A. Watkins, and David H. Albonesi. Leveraging optical technology in future bus-based chip multiprocessors. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, pages 492–503, Washington, DC, USA, 2006. IEEE Computer Society.

[71] Donald Kline, Jr., Kai Wang, Rami Melhem, and Alex K. Jones. Mscs: Multi-hop segmented circuit switching. In *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, GLSVLSI '15, pages 179–184, New York, NY, USA, 2015. ACM.

[72] Michihiro Koibuchi, Hiroki Matsutani, Hideharu Amano, D. Frank Hsu, and Henri Casanova. A case for random shortcut topologies for HPC interconnects. In *Proceedings of the 39th International Symposium on Computer Architecture*, ISCA '12, pages 177–188, Piscataway, NJ, USA, 2012. IEEE Press.

[73] Pranay Koka, Michael O. McCracken, Herb Schwetman, Xuezhe Zheng, Ron Ho, and Ashok V. Krishnamoorthy. Silicon-photonic network architectures for scalable, power-efficient multi-chip systems. In *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ISCA '10, pages 117–128, New York, NY, USA, 2010. ACM.

[74] S. Koohi, M. Abdollahi, and S. Hessabi. All-optical wavelength-routed NoC based on a novel hierarchical topology. In *International Symposium on Networks on Chip (NoCS)*, pages 97–104, 2011.

[75] Tushar Krishna, Li-Shiuan Peh, Bradford M. Beckmann, and Steven K. Reinhardt. Towards the ideal on-chip fabric for 1-to-many and many-to-1 communication. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44 '11, pages 71–82, New York, NY, USA, 2011. ACM.

[76] A Kumar, P. Kundu, A.P. Singhx, Li-Shiuan Peh, and N.K. Jha. A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS. In *25th International Conference on Computer Design, ICCD*, pages 63–70, 2007.

[77] Prabhat Kumar, Yan Pan, John Kim, Gokhan Memik, and Alok Choudhary. Exploring concentration and channel slicing in on-chip network router. In *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '09, pages 276–285, Washington, DC, USA, 2009. IEEE Computer Society.

[78] Rakesh Kumar, Victor Zyuban, and Dean M. Tullsen. Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling. In *Proceedings of the 32nd annual international symposium on Computer Architecture*, ISCA '05, pages 408–419, Washington, DC, USA, 2005. IEEE Computer Society.

[79] G. Kurian, Chen Sun, C.-H.O. Chen, J.E. Miller, J. Michel, Lan Wei, D.A. Antoniadis, Li-Shiuan Peh, L. Kimerling, V. Stojanovic, and A. Agarwal. Cross-layer energy and performance evaluation of a nanophotonic manycore processor system using real application workloads. In *Int. Parallel Distributed Processing Symposium (IPDPS)*, pages 1117–1130, 2012.

[80] George Kurian, Jason E. Miller, James Psota, Jonathan Eastep, Jifeng Liu, Jurgen Michel, Lionel C. Kimerling, and Anant Agarwal. Atac: A 1000-core cache-coherent processor with on-chip optical network. In *Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques*, PACT '10, pages 477–488, New York, NY, USA, 2010. ACM.

[81] S. Le Beux, J. Trajkovic, I O'Connor, G. Nicolescu, G. Bois, and P. Paulin. Multi-optical network-on-chip for large scale mpsoc. *Embedded Systems Letters, IEEE*, 2(3):77–80, Sept 2010.

[82] S. Le Beux, J. Trajkovic, I. O'Connor, G. Nicolescu, G. Bois, and P. Paulin. Optical ring network-on-chip (ORNoC): Architecture and design methodology. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, 2011.

[83] Chun-Yi Lee and N.K. Jha. Variable-pipeline-stage router. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(9):1669–1682, Sept 2013.

[84] Jinho Lee, Junwhan Ahn, Kiyoung Choi, and Kyungsu Kang. THOR: Orchestrated thermal management of cores and networks in 3D many-core architectures. In *20th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 773–778, Jan 2015.

[85] J. Leu and V. Stojanovic. Injection-locked clock receiver for monolithic optical link in 45nm SOI. In *IEEE Asian Solid State Circuits Conference (A-SSCC)*, pages 149–152, 2011.

[86] Zhongqi Li and Tao Li. Espn: A case for energy-star photonic on-chip network. In *Proceedings of the 2013 International Symposium on Low Power Electronics and Design*, ISLPED '13, pages 377–382, Piscataway, NJ, USA, 2013. IEEE Press.

[87] Shaoteng Liu, A. Jantsch, and Zhonghai Lu. Parallel probe based dynamic connection setup in TDM NoCs. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 1–6, March 2014.

[88] Mario Lodde, Toni Roca, and José Flich. Heterogeneous network design for effective support of invalidation-based coherency protocols. In *Proceedings of the 2012 Interconnection Network Architecture: On-Chip, Multi-Chip Workshop*, INA-OCMC '12, pages 1–4, New York, NY, USA, 2012. ACM.

[89] Plurality Ltd. The hypercore processor (tech. report), 2010. `www.plurality.com/` (Last access November 2015.

[90] Michael J. Lyons, Mark Hempstead, Gu-Yeon Wei, and David Brooks. The accelerator store: A shared memory framework for accelerator-based systems. *ACM Transactions on Architecture and Code Optimization (TACO)*, 8(4):48:1–48:22, January 2012.

[91] O. Lysne, J.M. Montanana, J. Flich, J. Duato, T.M. Pinkston, and T. Skeie. An efficient and deadlock-free network reconfiguration protocol. *Computers, IEEE Transactions on*, 57(6):762–779, June 2008.

[92] P.S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A full system simulation platform. *Computer*, 35(2):50 –58, feb 2002.

[93] A. Marongiu, P. Burgio, and L. Benini. Fast and lightweight support for nested parallelism on cluster-based embedded many-cores. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 105–110, March 2012.

[94] Andrea Marongiu, Alessandro Capotondi, Giuseppe Tagliavini, and Luca Benini. Improving the programmability of sthorm-based heterogeneous systems with offload-enabled openmp. In *Proceedings of the First International Workshop on Many-core Embedded Systems*, MES '13, pages 1–8, New York, NY, USA, 2013. ACM.

[95] Milo M. K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. *SIGARCH Computer Architecture News*, 33:92–99, November 2005.

[96] Abbas Mazloumi and Mehdi Modarressi. A hybrid packet/circuit-switched router to accelerate memory access in NoC-based chip multiprocessors. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE, Grenoble, France, March 9-13*, pages 908–911, 2015.

[97] Diego Melpignano, Luca Benini, Eric Flamand, Bruno Jego, Thierry Lepley, Germain Haugou, Fabien Clermidy, and Denis Dutoit. Platform 2012, a many-core computing accelerator for embedded SoCs: Performance evaluation of visual analytics applications. In *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, pages 1137–1142, New York, NY, USA, 2012. ACM.

[98] Jörg Mische and Theo Ungerer. Low power flitwise routing in an unidirectional torus with minimal buffering. In *Proceedings of the Fifth International Workshop on Network on Chip Architectures*, NoCArc '12, pages 63–68, New York, NY, USA, 2012. ACM.

[99] Asit K. Mishra, N. Vijaykrishnan, and Chita R. Das. A case for heterogeneous on-chip interconnects for CMPs. In *Proceedings of the 38th annual international symposium on Computer architecture*, ISCA '11, pages 389–400, New York, NY, USA, 2011. ACM.

[100] Randy Morris, Avinash Karanth Kodi, and Ahmed Louri. Dynamic reconfiguration of 3D photonic networks-on-chip for maximizing performance and improving fault tolerance. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-45, pages 282–293, Washington, DC, USA, 2012. IEEE Computer Society.

[101] Robert Mullins, Andrew West, and Simon Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the 31st annual international symposium on Computer architecture*, ISCA '04, pages 188–, Washington, DC, USA, 2004. IEEE Computer Society.

[102] Robert Mullins, Andrew West, and Simon Moore. The design and implementation of a low-latency on-chip network. In *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, ASP-DAC '06, pages 164–169, Piscataway, NJ, USA, 2006. IEEE Press.

[103] I. O'Connor, M. Brière, E. Drouard, A. Kazmierczak, F. Tissafi-Drissi, D. Navarro, F. Mieyeville, J. Dambre, D. Stroobandt, J.-M. Fedeli, Z. Lisik, and F. Gaffiot. Towards reconfigurable optical networks-on-chip. *RECO SoC*, pages 121–128, 2005.

[104] Ian O'Connor, Dries Van Thourhout, and Alberto Scandurra. Wavelength division multiplexed photonic layer on CMOS. In *Proceedings of the 2012 Interconnection Network Architecture: On-Chip, Multi-Chip Workshop*, INA-OCMC '12, pages 33–36, New York, NY, USA, 2012. ACM.

[105] Kunle Olukotun, Basem A. Nayfeh, Lance Hammond, Ken Wilson, and Kunyung Chang. The case for a single-chip multiprocessor. In *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS VII, pages 2–11, New York, NY, USA, 1996. ACM.

[106] Open Multi Processing. `http://www.openmp.org`. (Last access November 2015).

[107] Oracle. Sparc M7-16 Server. 2015. `http://www.oracle.com/us/products/servers-storage/sparc-m7-16-ds-2687045.pdf` (Last access November 2015).

[108] M. Ortin, L. Ramini, M. Balboni, L. Zuolo, N. Maddalena, V. Viñals, and D. Bertozzi. Partitioning strategies of wavelength-routed optical networks-on-chip for laser power minimization. In *Workshop on Exploiting Silicon Photonics for Energy-Efficient High Performance Computing (SiPhotonics)*, pages 17–24, Jan 2015.

[109] M. Ortin, D. Suarez, M. Villarroya, C. Izu, and V. Vinals. Dynamic construction of circuits for reactive traffic in homogeneous CMPs. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1–4, March 2014.

[110] Marta Ortín, Alexandra Ferrerón, Jorge Albericio, Darío Suárez, María Villarroya-Gaudó, Cruz Izu, and Víctor Viñals. Characterization and cost-efficient selection of NoC topologies for general purpose CMPs. In *Proceedings of the 2013 Interconnection Network Architecture: On-Chip, Multi-Chip*, IMA-OCMC '13, pages 21–24, New York, NY, USA, 2013. ACM.

[111] Marta Ortín-Obón, Luca Ramini, Herve Tatenguem Fankem, Víctor Viñals, and Davide Bertozzi. A complete electronic network interface architecture for global contention-free communication over emerging optical networks-on-chip. In *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, GLSVLSI '14, pages 267–272, New York, NY, USA, 2014. ACM.

[112] M. Ortín, L. Ramini, H. Fanken-Tatenguem, V. Viñals, and D. Bertozzi. Arquitectura completa de una interfaz de red electrónica para redes ópticas en chip. In *XXV Jornadas de Paralelismo*, September 2014.

[113] M. Ortín, D. Suárez, M. Villarroya-Gaudó, C. Izu, and V. Viñals. Reserva de circuitos para tráfico reactivo en cmps homogéneos. In *XXIV Jornadas de Paralelismo*, September 2013.

[114] Marta Ortín, Marco Balboni, Luca Ramini, Víctor Viñals, and Davide Bertozzi. Optical networks-on-chip: Time for accurate crossbenchmarking. In *CMOS Emerging Technologies Research*, July 2014. Invited presentation.

[115] Marta Ortín, Luca Ramini, Víctor Viñals, and Davide Bertozzi. Capturing the sensitivity of optical network quality metrics to its network interface parameters. In *Invited paper to the I Workshop on Exploiting Silicon Photonics for Energy-Efficient Heterogeneous Parallel Architectures (SiPhotonics)*, January 2014.

[116] Marta Ortín-Obón, Luca Ramini, Víctor Viñals, and Davide Bertozzi. Capturing the sensitivity of optical network quality metrics to its network interface parameters. *Concurrency and Computation: Practice and Experience*, 26(15):2504–2517, 2014.

[117] Marta Ortín-Obón, Darío Suárez-Gracia, María Villarroya-Gaudó, Cruz Izu, and Víctor Viñals-Yúfera. Analysis of network-on-chip topologies for cost-efficient chip multiprocessors. *Microprocessors and Microsystems*, pages –, 2016.

[118] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Tim Purcell. A survey of general-purpose computation on graphics hardware. COMPUTER GRAPHICS FORUM 26(1):80, 2007.

[119] Francesca Palumbo, Danilo Pani, Andrea Congiu, and Luigi Raffo. Concurrent hybrid switching for massively parallel systems-on-chip: the cyber architecture. In *Proceedings of the 9th conference on Computing Frontiers*, CF '12, pages 173–182, New York, NY, USA, 2012. ACM.

[120] Yan Pan, J. Kim, and G. Memik. Flexishare: Channel sharing for an energy-efficient nanophotonic crossbar. In *International Symposium on High Performance Computer Architecture*, pages 1–12, 2010.

[121] Yan Pan, Prabhat Kumar, John Kim, Gokhan Memik, Yu Zhang, and Alok Choudhary. Firefly: Illuminating future network-on-chip with nanophotonics. In *Proceedings of the International Symposium on Computer Architecture*, ISCA '09, pages 429–440, New York, NY, USA, 2009. ACM.

[122] Li-Shiuan Peh and William J. Dally. A delay model and speculative architecture for pipelined routers. In *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, HPCA '01, pages 255–, Washington, DC, USA, 2001. IEEE Computer Society.

[123] Li-Shiuan Peh and W.J. Dally. Flit-reservation flow control. In *High-Performance Computer Architecture, 2000. HPCA-6. Proceedings. Sixth International Symposium on*, pages 73–84, 2000.

[124] V. Puente, J.-A Gregorio, F. Vallejo, and R. Beivide. Immunet: a cheap and robust fault-tolerant packet routing mechanism. In *Proceedings of the 31st Annual International Symposium on Computer Architecture*, pages 198–209, June 2004.

[125] Arun Raghavan, Colin Blundell, and Milo M. K. Martin. Token tenure and PATCH: A predictive/adaptive token-counting hybrid. *ACM Transactions on Architecture and Code Optimization*, 7(2):6:1–6:31, October 2010.

[126] Amir-Mohammad Rahmani, Mohammad-Hashem Haghbayan, Anil Kanduri, Awet Yemane Weldezion, Pasi Liljeberg, Juha Plosila, Axel Jantsch, and Hannu Tenhunen. Dynamic power management for many-core platforms in the dark silicon era: A multi-objective control approach. In *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 219–224, July 2015.

[127] Luca Ramini and Davide Bertozzi. Power efficiency of wavelength-routed optical NoC topologies for global connectivity of 3D multi-core processors. In *Proceedings of the Fifth International Workshop on Network on Chip Architectures*, NoCArc '12, pages 25–30, New York, NY, USA, 2012. ACM.

[128] Luca Ramini, Hervé Tatenguem Fankem, Alberto Ghiribaldi, Paolo Grani, Marta Ortín-Obón, Anja Boos, and Sandro Bartolini. Towards compelling cases for the viability of silicon-nanophotonic technology in future manycore systems. In *Eighth IEEE/ACM International Symposium on Networks-on-Chip, NoCS 2014, Ferrara, Italy, September 17-19, 2014*, pages 170–171, 2014.

[129] Luca Ramini, Paolo Grani, Sandro Bartolini, and Davide Bertozzi. Contrasting wavelength-routed optical noc topologies for power-efficient 3d-stacked multicore processors using physical-layer analysis. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '13, pages 1589–1594, San Jose, CA, USA, 2013. EDA Consortium.

[130] Luca Ramini, Paolo Grani, Hervé Tatenguem Fankem, Alberto Ghiribaldi, Sandro Bartolini, and Davide Bertozzi. Assessing the energy break-even point between an optical NoC architecture and an aggressive electronic baseline. In *Proceedings of the Conference on Design, Automation & Test in Europe*, DATE '14, pages 308:1–308:6, 3001 Leuven, Belgium, Belgium, 2014. European Design and Automation Association.

[131] S. Rodrigo, J. Flich, A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, and J. Duato. Addressing manufacturing challenges with cost-efficient fault tolerant routing. In *International Symposium on Networks-on-Chip (NOCS)*, pages 25–32, May 2010.

[132] Edward Rosten, R. Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119, Jan 2010.

[133] Daniel Sanchez, George Michelogiannakis, and Christos Kozyrakis. An analysis of on-chip interconnection networks for large-scale chip multiprocessors. *ACM Transactions on Architecture and Code Optimization*, 7(1):4:1–4:28, May 2010.

[134] M.D. Schroeder, AD. Birrell, M. Burrows, H. Murray, R.M. Needham, T.L. Rodeheffer, E.H. Satterthwaite, and C.P. Thacker. Autonet: a high-speed, self-configuring local area network using point-to-point links. *IEEE Journal on Selected Areas in Communications*, 9(8):1318–1335, Oct 1991.

[135] Ciprian Seiculescu, Stavros Volos, Naser Khosro Pour, Babak Falsafi, and Giovanni De Micheli. CCNoC: On-Chip Interconnects for Cache-Coherent Manycore Server Chips. In *Proceedings of the Workshop on Energy-Efficient Design (WEED 2011)*, 2011.

[136] A. Shacham, K. Bergman, and L.P. Carloni. On the design of a photonic network-on-chip. In *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, pages 53–64, May 2007.

[137] Standard Performance Evaluation Corporation (SPEC). SPEC CPU2006. `http://www.spec.org/cpu2006/` (Last access November 2015).

[138] W.J. Starke, J. Stuecheli, D.M. Daly, J.S. Dodson, F. Auernhammer, P.M. Sagmeister, G.L. Guthrie, C.F. Marino, M. Siegel, and B. Blaner. The cache and memory subsystems of the IBM POWER8 processor. *IBM Journal of Research and Development*, 59(1):3:1–3:13, Jan 2015.

[139] S. Stergiou, F. Angiolini, Salvatore Carta, L. Raffo, D. Bertozzi, and G. De Micheli. xpipes lite: a synthesis oriented design library for networks on chips. In *Design, Automation and Test in Europe.*, pages 1188–1193 Vol. 2, 2005.

[140] A. Strano, D. Ludovici, and D. Bertozzi. A library of dual-clock FIFOs for cost-effective and flexible MPSoC design. In *International Conferenece on Embedded Computer Systems (SAMOS)*, pages 20–27, 2010.

[141] Chen Sun, Chia-Hsin Owen Chen, George Kurian, Lan Wei, Jason Miller, Anant Agarwal, Li-Shiuan Peh, and Vladimir Stojanovic. DSENT - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling. In *Proceedings of the 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, NOCS '12, pages 201–210, Washington, DC, USA, 2012. IEEE Computer Society.

[142] Chen Sun, Yu-Hsin Chen, and V. Stojanovic. Designing processor-memory interfaces with monolithically integrated silicon-photonics. In *Conference on Lasers and Electro-Optics Pacific Rim (CLEO-PR)*, pages 1–2, June 2013.

[143] Xianfang Tan, Mei Yang, Lei Zhang, Yingtao Jiang, and Jianyi Yang. On a scalable, non-blocking optical router for photonic networks-on-chip designs. In *Symposium on Photonics and Optoelectronics (SOPO)*, pages 1–4, 2011.

[144] Tilera. TILEPro64. 2008. `http://www.tilera.com/products/processors/TILEPro_Family` (Last access November 2015).

[145] TileraCorporation. Tile-Gx8016 specification. `http://www.tilera.com/sites/default/files/productbriefs/Tile-Gx-8016-SB011-03.pdf` (Last access November 2015).

[146] D. Vantrease, N. Binkert, R. Schreiber, and M.H. Lipasti. Light speed arbitration and flow control for nanophotonic interconnects. In *42nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-42*, pages 304–315, 2009.

[147] Dana Vantrease, Robert Schreiber, Matteo Monchiero, Moray McLaren, Norman P. Jouppi, Marco Fiorentino, Al Davis, Nathan Binkert, Raymond G. Beausoleil, and Jung Ho Ahn. Corona: System implications of emerging nanophotonic technology. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ISCA '08, pages 153–164, Washington, DC, USA, 2008. IEEE Computer Society.

[148] J.C. Villanueva, J. Flich, J. Duato, H. Eberle, N. Gura, and W. Olesinski. A performance evaluation of 2D-mesh, ring, and crossbar interconnects for chip multi-processors. In *Network on Chip Architectures, 2009. NoCArc 2009. 2nd International Workshop on*, pages 51–56, Dec 2009.

[149] Isask'har Walter, Israel Cidon, Ran Ginosar, and Avinoam Kolodny. Access regulation to hot-modules in wormhole NoCs. In *Proceedings of the First International Symposium on Networks-on-Chip*, NOCS '07, pages 137–148, Washington, DC, USA, 2007. IEEE Computer Society.

[150] David Wentzlaff, Patrick Griffin, Henry Hoffmann, Liewei Bao, Bruce Edwards, Carl Ramey, Matthew Mattina, Chyi-Chang Miao, John F. Brown III, and Anant Agarwal. On-chip interconnection architecture of the tile processor. *IEEE Micro*, 27(5):15–31, September 2007.

[151] Neil Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley Publishing Company, USA, 4th edition, 2010.

[152] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The SPLASH-2 programs: characterization and methodological considerations. In *Proceedings of the 22nd annual international symposium on Computer architecture*, ISCA '95, pages 24–36, New York, NY, USA, 1995. ACM.

[153] www.OpenMP.org. OpenMP Application Program Interface v.4.0. s`http://www.openmp.org/mp-documents/OpenMP_4.0_RC2.pdf`. (Last access November 2015).

[154] Young Jin Yoon, Nicola Concer, Michele Petracca, and Luca Carloni. Virtual channels vs. multiple physical networks: a comparative analysis. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 162–165, New York, NY, USA, 2010. ACM.

[155] Michael Zhang and Krste Asanovic. Victim replication: Maximizing capacity while hiding wire delay in tiled chip multiprocessors. In *Proceedings of the 32nd annual international symposium on Computer Architecture*, ISCA '05, pages 336–345, Washington, DC, USA, 2005. IEEE Computer Society.

[156] Pingqiang Zhou, Jieming Yin, Antonia Zhai, and Sachin S. Sapatnekar. NoC frequency scaling with flexible-pipeline routers. In *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*, ISLPED '11, pages 403–408, Piscataway, NJ, USA, 2011. IEEE Press.